# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Целью изучения дисциплины «Проектирование программ в интеллектуальных системах» является рассмотрение технологий создания и дальнейшего развития надежных, гибких и эффективных программных систем (организация работ, анализ предметной области, спецификация требований, разработка технического задания, проектирование, моделирование, конструирование). В рамках данного курса уделяется отдельное внимание языкам программирования, использующимся при построении интеллектуальных систем, в частности графовым языкам программирования.

Целью выполнения курсовой работы по данной дисциплине является закрепление навыков

- разработки программных проектов и фрагментов проектов с применением современных методик и инструментальных средств;
- применения методов из области теории графов для решения прикладных задач в различных предметных областях;
- использования инструментальных средств разработки интеллектуальных систем;
- создания программ, ориентированных на обработку различных видов знаний, хранимых в памяти интеллектуальных систем.

Для достижения цели необходимо решить следующие задачи:

1. Изучить соответствующие разделы теории, необходимые для решения выбранной теоретико-графовой задачи;
2. Разработать алгоритм решения поставленной задачи в общем виде, без привязки к какому-либо конкретному варианту представления графа в памяти компьютера;
3. Реализовать разработанный алгоритм на выбранном языке программирования на основе предлагаемой библиотеки для работы с семантической памятью;
4. Реализовать разработанный алгоритм на графовом языке программирования SCP;

Методические указания включают в себя:

1. Общее описание типовых конструкций, хранимых и обрабатываемых в рамках семантической памяти;
2. Описание языка SCP с конкретными примерами применения операторов;
3. Примеры реализованных подпрограмм на языке SCP.

## ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

***Язык SCP*** – это графовый язык процедурного программирования, предназначенный для эффективной обработки однородных семантических сетей с теоретико-множественной интерпретацией, закодированных с помощью *SC-кода*. ***Язык SCP*** является языком параллельного асинхронного программирования. Подробно язык SCP и его особенности описаны в [1], [2], [3].

Языком представления данных для *текстов языка SCP* (*scp-программ*) является *SC-код* и, соответственно, любые варианты его представления. ***Язык SCP*** сам построен на основе *SC-кода*, вследствие чего *scp-программы* сами по себе могут входить в состав данных *scp-программ*, в т.ч. по отношению к самим себе. Таким образом, ***язык SCP*** предоставляет возможность построения реконфигурируемых программ. Однако для обеспечения возможности реконфигурирования программы непосредственно в процессе ее интерпретации необходимо на уровне *интерпретатора языка SCP (scp-интерпретатора)* обеспечить уникальность каждой исполняемой копии исходной программы.

***Язык SCP*** рассматривается как ассемблер для графодинамического компьютера, ориентированного на хранение и обработку семантических сетей.

Каждая ***scp-программа*** с теоретико-множественной точки зрения является кортежем, который включает следующие компоненты:

–*sc-узел*, обозначающий множество параметров ***scp-программы***. Является элементом конкретной *scp-программы* под атрибутом *параметры'*. Множество параметров является ориентированным, знаки конкретных *scp-параметров* упорядочиваются при помощи ролевых отношений *1'*, *2'*, *3'* и т.д., а также обязательно указывается тип *scp-параметра — in-параметр'* или *out-параметр'*.

–*sc-узел*, обозначающий множество операторов ***scp-программы*** (*scp-операторов*). Является элементом конкретной *scp-программы* под атрибутом операторы′. Все операторы в рамках данного множества равноправны, за исключением операторов, которые должны быть выполнены в первую очередь. Данные операторы помечаются атрибутом *начальный оператор'*. Знаками всех *scp-операторов*, входящих в данное множество, являются *sc-переменные*, равно как *sc-переменными* являются и все *sc-коннекторы*, связывающие знаки *scp-операторов* ***scp-программы*** с ключевыми *sc-узлами scp-интерпретатора* и операндами. На все *sc-переменные*, участвующие в формировании множества операторов *scp-программы* неявно накладывается квантор существования.

Для записи ***scp-программ*** могут использоваться любые варианты представления *SC-кода,* такие как *SCg-код*, *SCs-код*, *SC-код*.

По сути каждая ***scp-программа*** представляет собой описание последовательности операций, которые необходимо выполнить над семантической сетью, с указанием параметров для выполнения данных операций.

В зависимости от конкретной структуры множества параметров ***scp-программы***, могут быть выделен класс *scp-процедур*.

***scp-процедурой*** будем называть *scp-программу*, множество параметров которой содержит хотя бы один элемент, в независимости от типа данного параметра.

В рамках множества ***scp-процедур***, можно выделить подмножество *агентных scp-процедур* или *агентных scp-программ*.

***агентные scp-программы*** представляют собой частный случай *scp-программ* вообще, однако заслуживают отдельного рассмотрения, поскольку используются наиболее часто. *scp-программы* данного класса представляют собой реализации программ агентов обработки знаний, и имеют жестко фиксированную структуру множества параметров. Множество параметров в данном случае состоит из двух in-параметров. Значение первого параметра является знаком бинарной ориентированной пары, являющееся вторым компонентом связки отношения *первичное условие инициирования** для абстрактного *sc-агента*, в множество *программ sc-агента** которого входит рассматриваемая ***агентная scp-программа.***[4]

Значением второго параметра является *sc-элемент*, с которым непосредственно связано *sc-событие*, в результате возникновения которого был инициирован соответствующий *sc-агент*, т.е., например, сгенерированная либо удаляемая *sc-дуга* или *sc-ребро*.

Ролевое отношение ***параметры'*** указывает множество параметров в рамках *scp-программы*.

Ролевое отношение ***операторы'*** указывает множество операторов в рамках *scp-программы.*

Ролевое отношение ***начальный оператор'*** указывает в рамках множества операторов *scp-программы* операторы, которые должны быть выполнены в первую очередь.

Параметры типа ***in-параметр'*** хоть и соответствуют переменным *scp-программы*, однако не могут менять значение в процессе ее интерпретации. Фиксированное значение переменной устанавливается при создании уникальной копии *scp-программы* (*scp-процесса*) для ее интерпретации, и,

таким образом, соответствующая *scp-переменная'* на момент начала ее интерпретации становится *scp-константой'* в рамках каждого *scp-оператора*, в котором встречалась данная *scp-переменная'*. Использование *in-параметров* можно рассматривать по аналогии с использованием варианта механизма передачи по значению в традиционных языках программирования, с тем условием, что значение локальной переменной в рамках дочерней программы не может быть изменено.

Параметры типа ***out-параметр'*** соответствуют *переменным scp-программы'* и обладают всеми теми же соответствующими свойствами. Чаще всего предполагается, что значение данного параметра необходимо родительской *scp-программе*, содержащей оператор вызова текущей *scp-программы*. При этом на момент начала интерпретации в качестве параметра дочернему процессу передается непосредственно узел, обозначающий переменную (а точнее, ее уникальную копию в рамках процесса) родительского процесса. Указанная переменная может при необходимости иметь значение, либо не иметь. После завершения и во время интерпретации дочернего процесса родительский процесс по-прежнему может работать с переменной, переданной в качестве *out-параметра',* при необходимости просматривая или изменяя ее значение. Использование out-параметра можно рассматривать по аналогии с использованием механизма передачи по ссылке в традиционных языках программирования.

# 1 ТИПОЛОГИЯ КОНСТРУКЦИЙ, ХРАНИМЫХ В СЕМАНТИЧЕСКОЙ ПАМЯТИ

***sc-конструкция*** – это минимальная единица *sc-графа*, обработка которой описывается каким-либо классом *scp-операторов*. Каждый элемент ***sc-конструкции*** имеет свою строго фиксированную позицию и может иметь определенный тип. Множество ***sc-конструкций*** разбивается на следующие классы:

- ***sc-конструкция стандартного вида;***
- ***sc-конструкция нестандартного вида.***

В свою очередь множество ***sc-конструкций стандартного вида*** разбивается на следующие подклассы:

- ***одноэлементная sc-конструкция;***
- ***трехэлементная sc-конструкция;***
- ***пятиэлементная sc-конструкция.***

Каждая ***одноэлементная sc-конструкция*** состоит из одного *sc-элемента* произвольного типа.

Пример ***одноэлементной sc-конструкции:***


Рис. 1.1 Одноэлементная sc-конструкция

Каждая ***трехэлементная sc-конструкция*** состоит из трех sc-элементов. Второй элемент всегда является *sc-коннектором*, остальные элементы могут быть произвольного типа.

Пример ***трехэлементной sc-конструкции:***


Рис. 1.2 Трехэлементная sc-конструкция

Каждая ***пятиэлементная sc-конструкция*** состоит из пяти *sc-элементов*. Второй и четвертый элементы обязательно являются sc-коннекторами остальные элементы могут быть произвольного типа.

Пример ***пятиэлементной sc-конструкции:***

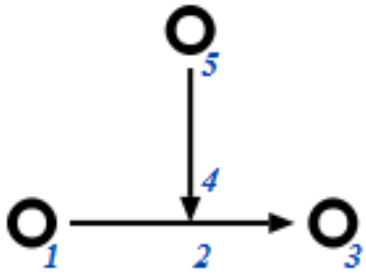



Рис. 1.3 Пятиэлементная sc-конструкция

Каждая ***sc-конструкция нестандартного вида*** состоит из произвольного количества *sc-элементов* произвольного типа.

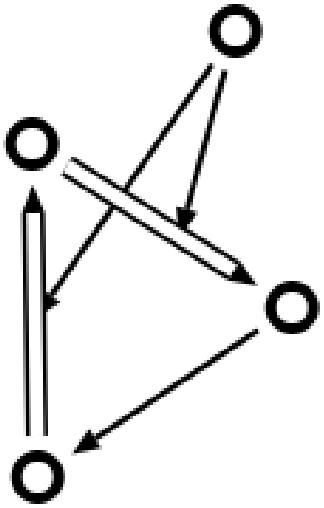


Рис. 1.4 Нестандартная sc-конструкция

# 2 ОТНОШЕНИЯ, ИСПОЛЬЗУЕМЫЕ ПРИ ПОСТРОЕНИИ SCP-ПРОГРАММ

## 2.1 Отношения, уточняющие роль операнда в рамках scp-оператора

Ролевое отношение ***тип scp-операнда'*** используется для указания того, как должен рассматриваться текущий операнд в процессе выполнения *scp-оператора* - как переменная, либо как константа.

Данное ролевое отношение разбивается на следующие подмножества:

- ***scp-константа';***
- ***scp-переменная'.***

***scp-константы'*** в рамках *scp-программы* явно участвуют в *scp-операторах* в качестве элементов и напрямую обрабатываются при интерпретации *scp-программы*. Константами в рамках *scp-программы* могут быть *sc-элементы* любого типа, как *sc-константы*, так и *sc-переменные*. Константа в рамках *scp-программы* остается неизменной в течение всего срока интерпретации, в связи с чем нет необходимости хранить связку отношения *значение** для констант. Константа *scp-программы* может быть рассмотрена как переменная, значение которой совпадает с самой переменной в каждый момент времени, и изменено быть не может. Таким образом, далее будем считать, что *scp-константа'* и ее значение это одно и то же. Каждый *in-параметр'* при интерпретации каждой конкретной копии *scp-программы* становится *scp-константой'* в рамках всех ее операторов, хотя в исходном теле данной программы в каждом из этих операторов он является *scp-переменной'*.

***scp-переменные'*** в рамках *scp-программы* не обрабатываются явно при интерпретации, обрабатываются значения переменных. Каждая переменная *scp-программы* может иметь одно значение в каждый момент времени, что указывается связкой соответствующего отношения *значение**. Переменной *scp-программы* может быть только *sc-переменная*. Значение каждой *scp-переменной'*может меняться в ходе интерпретации *scp-программы*. При этом *scp-интерпретатор* при обработке *scp-оператора* работает непосредственно со значениями *scp-переменных'*, а не самими *scp-переменными'* (которые также являются узлами той же семантической сети).

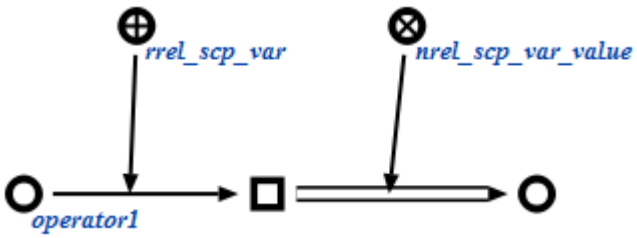



Рис. 2.1 Scp-переменная

Связки отношения ***значение**** связывают *sc-элементы*, выступающие в качестве *scp-переменных'* в рамках *scp-операторов*, с их значениями.

Ролевое отношение ***тип значения scp-операнда'*** используется для указания факта наличия значения у операнда на момент начала выполнения *scp-оператора*.

Данное ролевое отношение разбивается на следующие подмножества:

- ***scp-операнд с заданным значением';***
- ***scp-операнд со свободным значением'.***

Таблица 1 поясняет возможные комбинации ролевых отношений ***тип scp-операнда'*** и ***тип значения scp-операнда'.***

Таблица 1 – комбинации ролевых отношений

| | ***scp-операнд с заданным значением'*** | ***scp-операнд со свободным значением'*** |
|---|---|---|
| ***scp-константа'*** | Разрешено, может быть опущено | Запрещено |
| ***scp-переменная'*** | Разрешено, значение останется неизменным | Разрешено, значение переменной будет изменено либо потеряно |

Значение операндов, помеченных ролевым отношением ***scp-операнд с заданным значением'***, считается заданным в рамках текущего *scp-оператора*. Данное значение учитывается при выполнении *scp-оператора* и остается неизменным после окончания выполнения *scp-оператора*. Каждая *scp-константа'* по умолчанию рассматривается как ***scp-операнд с заданным значением'***, в связи с чем явное использование данного ролевого отношения в таком случае является избыточным**.** В таком случае в качестве значения рассматривается непосредственно сам операнд. В случае, если отношением ***scp-операнд с заданным значением'*** помечена *scp-переменная'*, то осуществляется попытка поиска значения для данной *scp-переменная'* (связки отношения *значение**). Если попытка оказалась безуспешной, то *scp-интерпретатор* заявляет о наличии возникновении ошибки времени выполнения.

Следует отметить, что любой ***scp-операнд с заданным значением'***, независимо от конкретного типа *scp-оператора*, может быть *scp-переменной'*.

Значение операндов, помеченных ролевым отношением ***scp-операнд со свободным значением'***, считается свободным (на заданным заранее) в рамках текущего *scp-оператора.* В начале выполнения *scp-оператора* связка отношения *значение** для *scp-переменной'*, помеченной данным ролевым

отношением, всегда удаляется. В результате выполнения данного оператора может быть либо сгенерирована новая связка отношения *значение** для данной *scp-переменной'*, либо не сгенерирована, тогда *scp-переменная'* будет считаться не имеющей значения. Ни одна *scp-константа'* не может быть помечена как ***scp-операнд со свободным значением'***, поскольку константа не может изменять свое значение в ходе интерпретации *scp-программы*.

Ролевое отношение ***тип sc-элемента'*** используется для уточнения типа *sc-элемента*, выступающего в роли значения некоторого операнда. ***тип sc-элемента'*** имеет смысл указывать только для операндов, помеченных как ***scp-операнд со свободным значением'***, тогда данное уточнение типа *sc-элемента* будет использовано для сужения области поиска либо уточнения параметров генерации каких-либо конструкций. Значением ***scp-операндов с заданным значением'*** является конкретный, известный на момент начала выполнения *scp-оператора sc-элемент* с конкретным типом, не зависящим от указания ***типа sc-элемента'***, в связи с чем использование ролевого отношения ***тип sc-элемента'*** в данном случае является некорректным.

Допускается использование комбинаций семантически непротиворечащих друг другу подмножеств указанного отношения. Например, допускается комбинация ***константный sc-элемент'*** и ***sc-дуга общего вида'***, но не допускается комбинация ***sc-узел'*** и ***sc-дуга'***.

Данное ролевое отношение разбивается на следующие подмножества:

- ***константный sc-элемент';***
- ***переменный sc-элемент';***
- ***sc-узел';***
- ***sc-дуга';***
- ***sc-ребро';***
- ***sc-ссылка'.***

Ролевое отношение ***sc-дуга'*** разбивается на следующие подмножества:

- ***sc-дуга общего вида';***
- ***sc-дуга принадлежности';***

Ролевое отношение ***sc-дуга принадлежности'*** разбивается на следующие подмножества:

- ***позитивная sc-дуга принадлежности';***
- ***негативная sc-дуга принадлежности';***
- ***нечеткая sc-дуга принадлежности';***
- ***временная sc-дуга принадлежности';***
- ***постоянная sc-дуга принадлежности';***

Для удобства программирования выделяется также ролевое отношение ***sc-дуга основного вида'*** являющееся объединением отношений *константный sc-элемент'*, *позитивная sc-дуга принадлежности'* и *постоянная sc-дуга принадлежности'*.

Ролевое отношение ***формируемое множество'*** используется для указания того факта, что в результате выполнения *scp-оператора* должно быть сформировано либо дополнено некоторое множество *sc-элементов*, являющееся значением одного из операндов данного *scp-оператора*. При этом если данным операнд помечен как *scp-операнд со свободным значением'*, то множество будет сформировано с нуля (сгенерирован новый sc-элемент, обозначающий данное множество), в противном случае уже существующее множество может быть дополнено. Использование данного ролевого отношения предполагает, что при его отсутствии множество бы не формировалось, а значением указанного операнда стал бы произвольный *sc-элемент* из данного множества. В данной версии *языка SCP* ролевое отношение ***формируемое множество'*** без уточнения порядкового номера используется только в *scp-операторах обработки произвольных конструкций*. Для явного указания номера операнда, которому соответствует ***формируемое множество'*** используются подмножества данного ролевого отношения, аналогичные ролевым отношениям, задающим порядок элементов в ориентированном множестве (*1'*, *2'*, *3'* и т.д.), например *формируемое множество 1'*, *формируемое множество 2'* и т.д. В данной версии *языка SCP* указанные ролевые отношения используются только в *scp-операторах поиска конструкций с формированием множеств*.

Ролевое отношение ***удаляемый sc-элемент'*** используется для указания тех операндов, значение которых должно быть удалено в процессе выполнения *scp-операторов удаления* (в данном случае удаляется не только связка отношения *значение**, но и непосредственно сам *sc-элемент*, являющийся значением). Данным ролевым отношением может быть помечен как *scp-операнд с заданным значением'*, так и *scp-операнд со свободным значением'*. При этом удаляемым *sc-элементом* может быть как *scp-константа'*, так и *scp-переменная'*.

### 2.2 Отношения, указывающие порядок выполнения scp-операторов

Отношение ***следующий оператор**** используется для указания *scp-оператора*, который должен выполниться следующим после выполнения текущего *scp-оператора*, в случае, если выполнение текущего оператора завершилось без ошибок, и нет никаких дополнительных условий успешности выполнения, зависящих от конкретного класса оператора. Переход в таком случае можно считать безусловным. Данное ролевое отношение используется в

большинстве scp-операторов, за исключением *scp-операторов ассоциативного поиска sc-конструкций* и *scp-операторов проверки условий.* Допускается использование нескольких выходящих пар данного отношения для одного и того же оператора в случае, если после завершения выполнения текущего оператора предполагается выполнение нескольких операторов параллельно. Допускается одновременное использование одной или нескольких связок отношения ***следующий оператор**** вместе с его подмножествами для одного оператора, в случае, когда некоторые операторы должны быть выполнены независимо от успешности выполнения каких-либо дополнительных условий, а некоторые операторы, в свою очередь, должны быть выполнены в зависимости от успешности выполнения каких-либо дополнительных условий.

Данное отношение разбивается на следующие подмножества:

- ***следующий оператор при успешном выполнении****
- ***следующий оператор при безуспешном выполнении****

Отношение ***следующий оператор при успешном выполнении**** используется для указания *scp-оператора*, который должен выполниться следующим после выполнения текущего *scp-оператора*, в случае, если выполнение текущего оператора завершилось без ошибок, а также выполнено некоторое дополнительное условие, зависящее от конкретного класса оператора (например, попытка поиска заданной конструкции оказалась успешной). Допускается использование нескольких выходящих пар данного отношения для одного и того же оператора в случае, если после завершения выполнения текущего оператора предполагается выполнение нескольких операторов параллельно.

Отношение ***следующий оператор при безуспешном выполнении**** используется для указания *scp-оператора*, который должен выполниться следующим после выполнения текущего *scp-оператора*, в случае, если выполнение текущего оператора завершилось без ошибок, однако не выполнено некоторое дополнительное условие, зависящее от конкретного класса оператора (например попытка поиска заданной конструкции оказалась безуспешной при заданных параметрах поиска). Допускается использование нескольких выходящих пар данного отношения для одного и того же оператора в случае, если после завершения выполнения текущего оператора предполагается выполнение нескольких операторов параллельно.

# 3 ТИПОЛОГИЯ ОПЕРАТОРОВ SCP-ПРОГРАММ

***scp-операторы*** описывают операции, которые необходимо выполнить над *sc-памятью*. Каждый *scp-оператор* является кортежем, элементы которого будем называть операндами. Порядок операндов указывается при помощи соответствующих ролевых отношений (*1', 2', 3'* и так далее). Операнд, помеченный ролевым отношением *1'*, будем называть первым операндом и т.д., помеченный ролевым отношением *2'* – вторым операндом, и т.д. Также при помощи соответствующих ролевых отношений указывается дополнительная информация о каждом операнде, необходимая для корректного выполнения ***scp-оператора***. В общем случае операндом может быть любой *sc-элемент*, в том числе, знак какой-либо *scp-программы*, в том числе самой программы, содержащей данный оператор.

Каждый ***scp-оператор*** должен иметь один и более операнд, а также указание того ***scp-оператора***, который должен быть выполнен следующим. Исключение их данного правила составляет *scp-оператор завершения выполнения программы*, который не содержит ни одного операнда и после выполнения которого никакие *scp-операторы* в рамках данной программы выполняться не могут.

Для того, чтобы обеспечить возможность распараллеливания scp-программы на уровне scp-операторов вводятся два дополнительных класса scp-операторов:

- ***scp-оператор, выполняемый после завершения хотя бы одного из предыдущих;***
- ***scp-оператор, выполняемый после завершения всех предыдущих;***

Операторы класса ***scp-оператор, выполняемый после завершения хотя бы одного из предыдущих*** могут быть выполнены в том случае, когда завершилось выполнение хотя бы одного из *scp-операторов* той же *scp-программы*, являющихся первым компонентом связок отношения ***следующий оператор**** (или его подмножеств), в которых данный *scp-оператор* является вторым компонентом. Использование операторов данного класса возможно в случаях, когда часть *scp-операторов* могут выполняться независимо от оставшейся части *scp-программы* и их выполнение не влияет на дальнейшее выполнение *scp-программы*. По умолчанию, если не указано иное считается, что *scp-оператор* принадлежит данному классу.

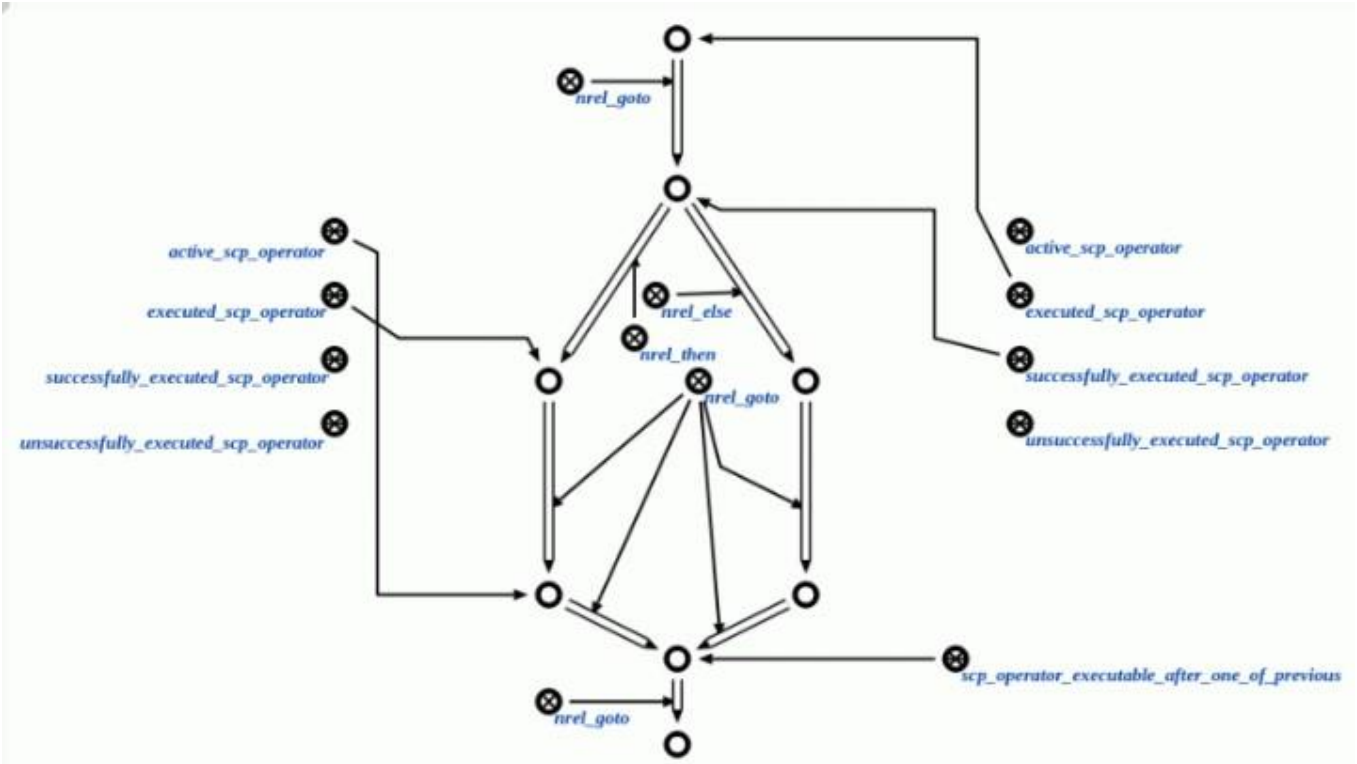



Рис. 3.1 Оператор класса scp-оператор, выполняемый после завершения хотя бы одного из предыдущих, ч.1

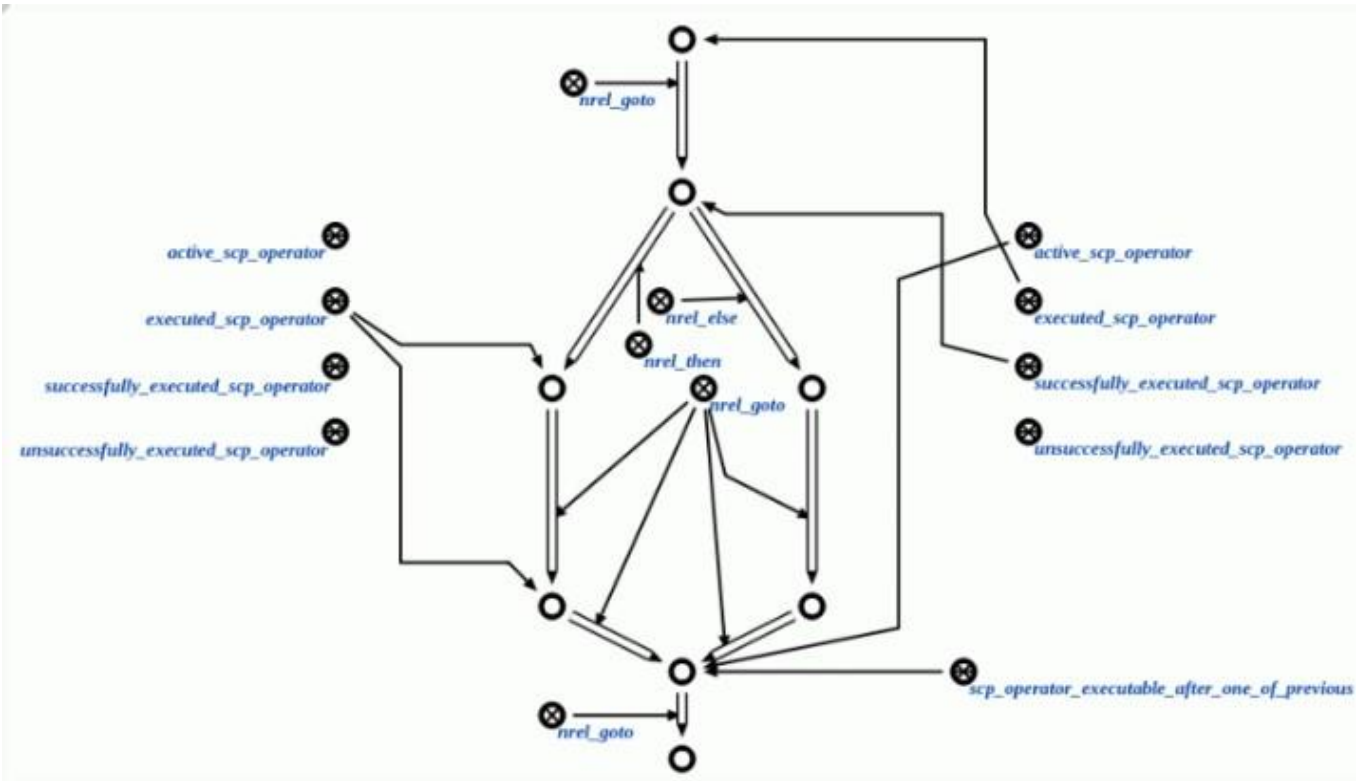



Рис. 3.2 Оператор класса scp-оператор, выполняемый после завершения хотя бы одного из предыдущих, ч.2

Операторы класса ***scp-оператор, выполняемый после завершения всех предыдущих*** могут быть выполнены только в том случае, когда завершилось выполнение всех *scp-операторов* той же *scp-программы*, являющихся первым компонентом связок отношения ***следующий оператор**** (или его подмножеств), в которых данный *scp-оператор* является вторым компонентом.

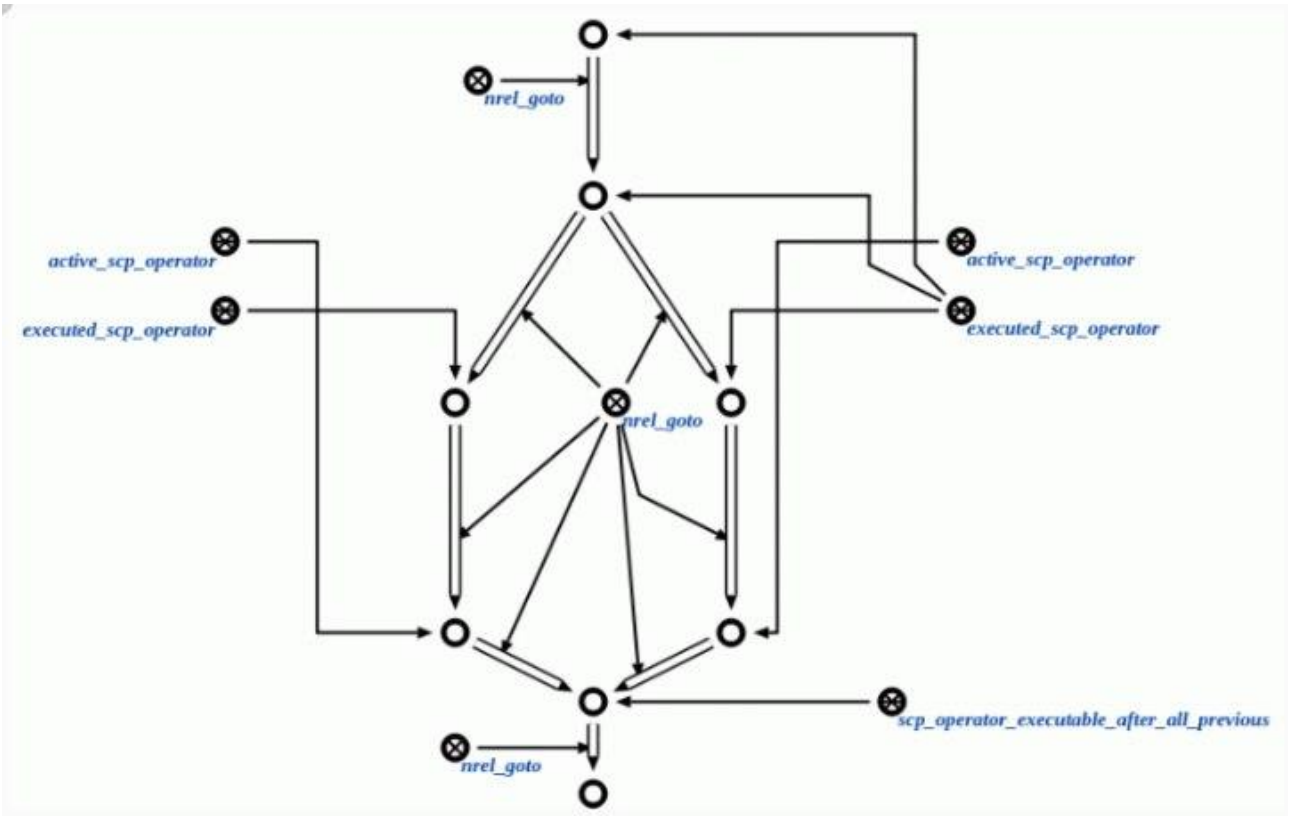



Рис. 3.3 Операторы класса scp-оператор, выполняемый после завершения всех предыдущих, ч.1

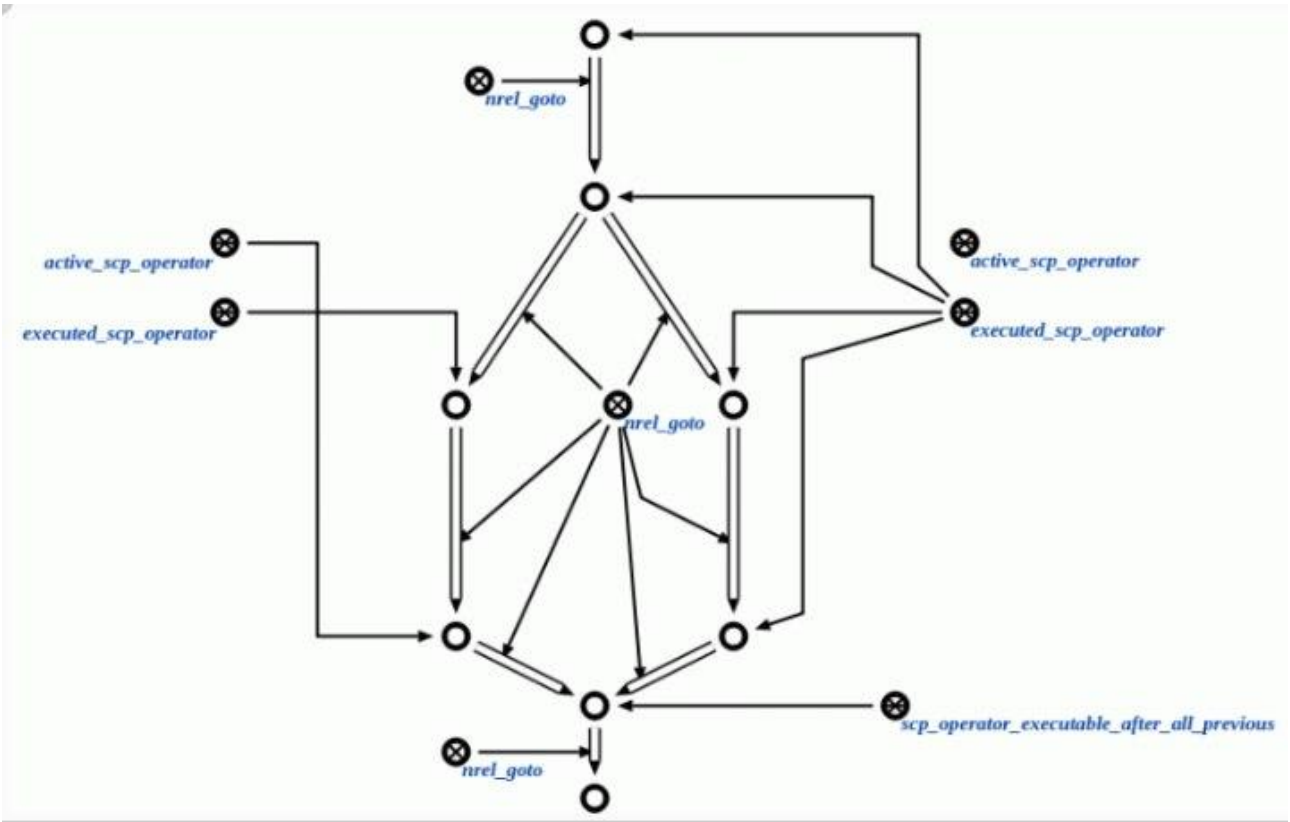



Рис. 3.4 Операторы класса scp-оператор, выполняемый после завершения всех предыдущих, ч.2

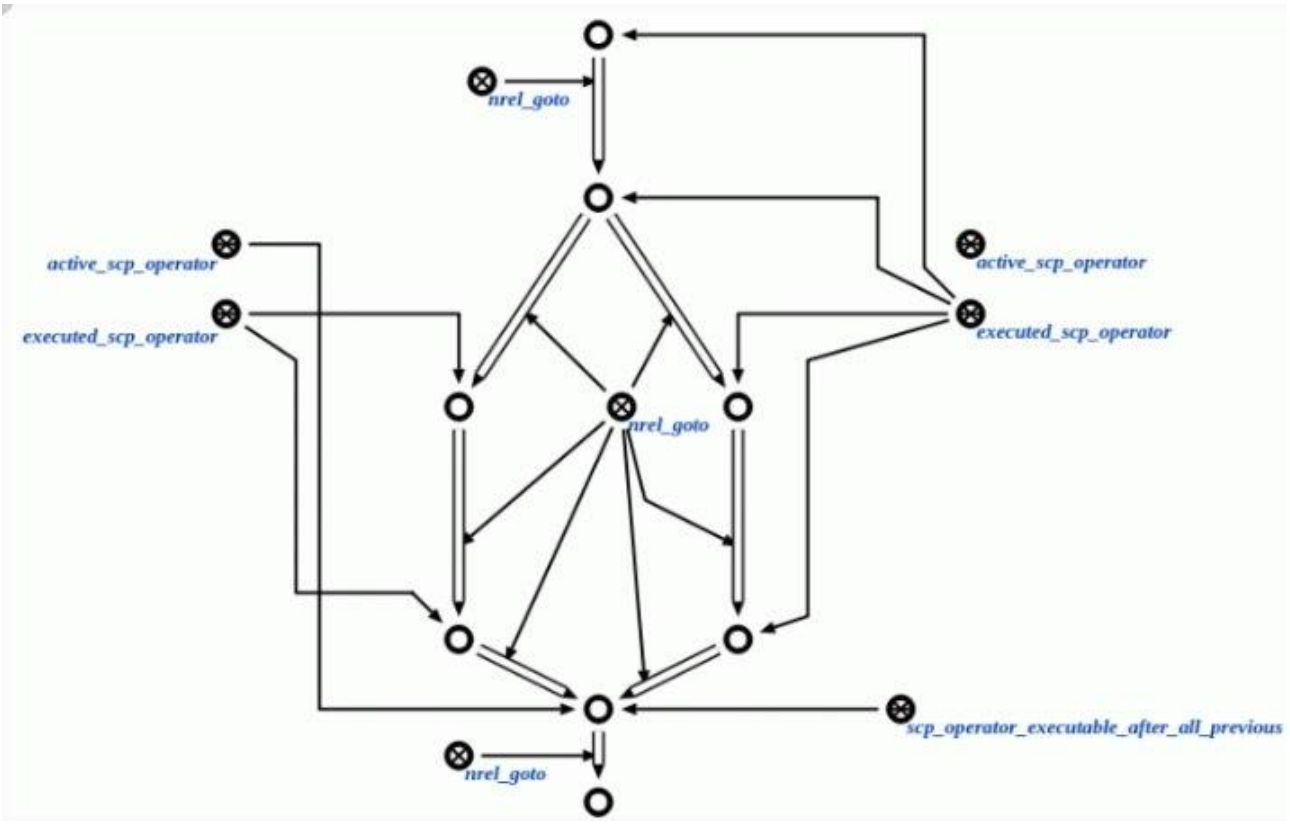



Рис. 3.5 Операторы класса scp-оператор, выполняемый после завершения всех предыдущих, ч.3

В соответствии с типом описываемых операций ***scp-операторы*** разбиваются на следующие классы:

- ***scp-оператор генерации sc-конструкций;***
- ***scp-оператор ассоциативного поиска sc-конструкций;***
- ***scp-оператор удаления sc-конструкций;***
- ***scp-оператор проверки условий;***
- ***scp-оператор обработки содержимых sc-ссылок;***
- ***scp-оператор управления scp-процессами;***
- ***scp-оператор управления событиями;***
- ***scp-оператор управления значениями операндов;***
- ***scp-оператор вывода информации в консоль;***
- ***scp-оператор синхронизации выполнения scp-программы***

### 3.1 Scp-операторы генерации sc-конструкций

Данный класс *scp-операторов* разбивается на следующие подклассы:

- ***scp-оператор генерации одноэлементной sc-конструкции;***
- ***scp-оператор генерации трехэлементной sc-конструкции;***
- ***scp-оператор генерации пятиэлементной sc-конструкции;***
- ***scp-оператор генерации sc-конструкции по произвольному образцу.***

Операторы класса ***scp-оператор генерации sc-конструкций*** описывают действия генерации каких-либо *sc-конструкций*. В результате выполнения операторов данного класса в памяти системы появляются новые сгенерированные *sc-элементы*. Операторы класса не предполагают ветвления программы в зависимости от выполнения дополнительных условий, поэтому указание следующих операторов осуществляется только при помощи отношения *следующий оператор**, без подмножеств.

Операторы класса ***scp-оператор генерации одноэлементной sc-конструкции*** описывают действие генерации одного *sc-элемента* указанного типа. Каждый оператор данного класса содержит один операнд, обязательно помеченный как *scp-операнд со свободным значением'*. Так же может быть уточнен тип генерируемого sc-элемента при помощи подмножеств ролевого отношения тип *sc-элемента'*. В результате выполнения оператора будет сгенерирован новый *sc-элемент* указанного типа.

Пример записи оператора на языке SCs:

```
scp_operator_example_genEl (*
        <- genEl;;
        -> rrel_1: rrel_assign: rrel_const: rrel_node: rrel_scp_var: _elem1;;
```

```
        => nrel_goto: scp_operator_example_goto;;
*);;
```

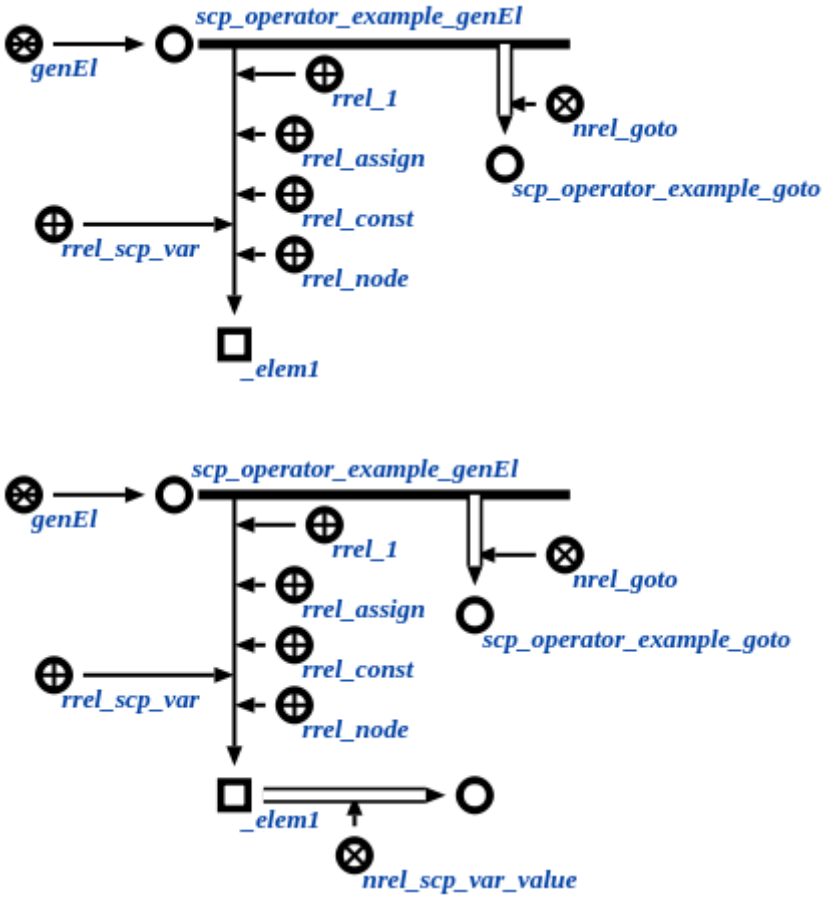



Рис. 3.6 Оператор генерации одноэлементной конструкции (Пример 1)

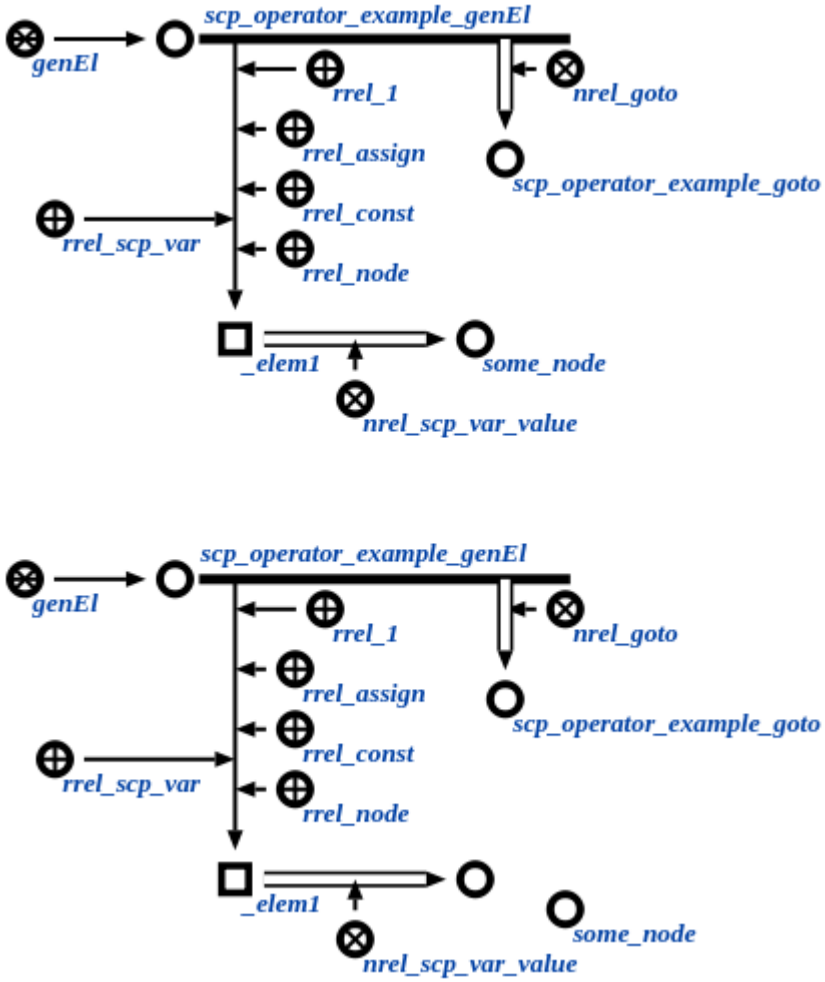



Рис. 3.7 Оператор генерации одноэлементной конструкции (Пример 2)

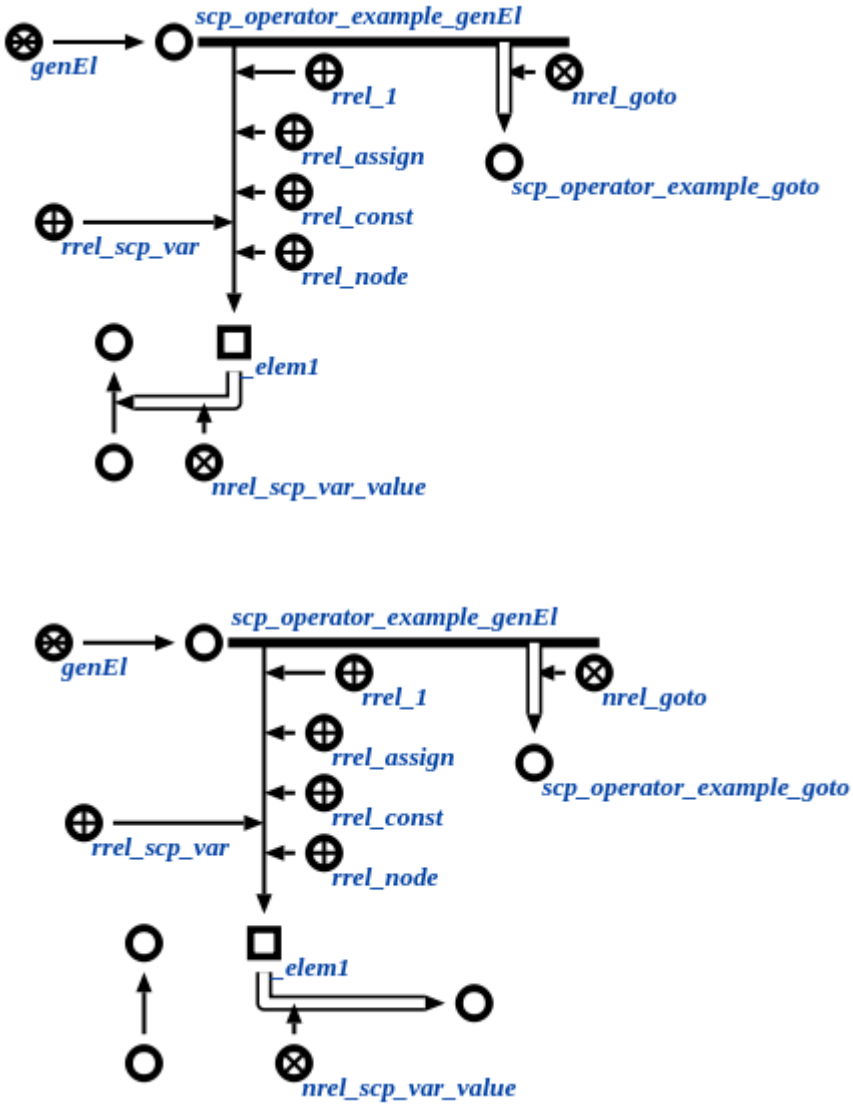



Рис. 3.8 Оператор генерации одноэлементной конструкции (Пример 3)

Операторы класса ***scp-оператор генерации трехэлементной sc-конструкции*** описывают действие генерации всех либо некоторых *sc-элементов* указанного типа, составляющих *трехэлементную sc-конструкцию*. Каждый оператор данного класса содержит три операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*, и соответствует первому элементу *трехэлементной sc-конструкции*.

Второй операнд может иметь произвольный *тип значения scp-операнда'*, и соответствует второму элементу *трехэлементной sc-конструкции*.

В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то генерируется *sc-коннектор* указанного типа. В случае, если данный операнд помечен как *scp-операнд с заданным значением'*, то *sc-коннектор* не генерируется, а будут изменены оба или один из инцидентных

ему *sc-элемента*. То есть можно сказать, что будут «переставлены» начало и/или конец данного *sc-коннектора*. При этом физический адрес *sc-коннектора* может измениться, но конфигурация окружающей его семантической сети (входящие и выходящие *sc-коннекторы*, если такие были) будет полностью соответствовать той, что была до выполнения оператора.

Третий операнд может иметь произвольный *тип значения scp-операнда'*, и соответствует третьему элементу *трехэлементной sc-конструкции*.

В результате выполнения оператора для всех операндов, помеченных как *scp-операнд со свободным значением'*, будет сгенерировано значение, т.е. новый *sc-элемент* соответствующего типа, с учетом уточнений типа. Тип генерируемого *sc-элемента* может быть уточнен при помощи подмножеств ролевого отношения *тип sc-элемента'*.

Пример записи оператора на языке SCs:

```
scp_operator_example_genElStr3 (*
        <- genElStr3;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _set;;
        -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var: _arc1;;
        -> rrel_3: rrel_assign: rrel_scp_var: _element;;

        => nrel_goto: scp_operator_example_goto;;
*);;
```

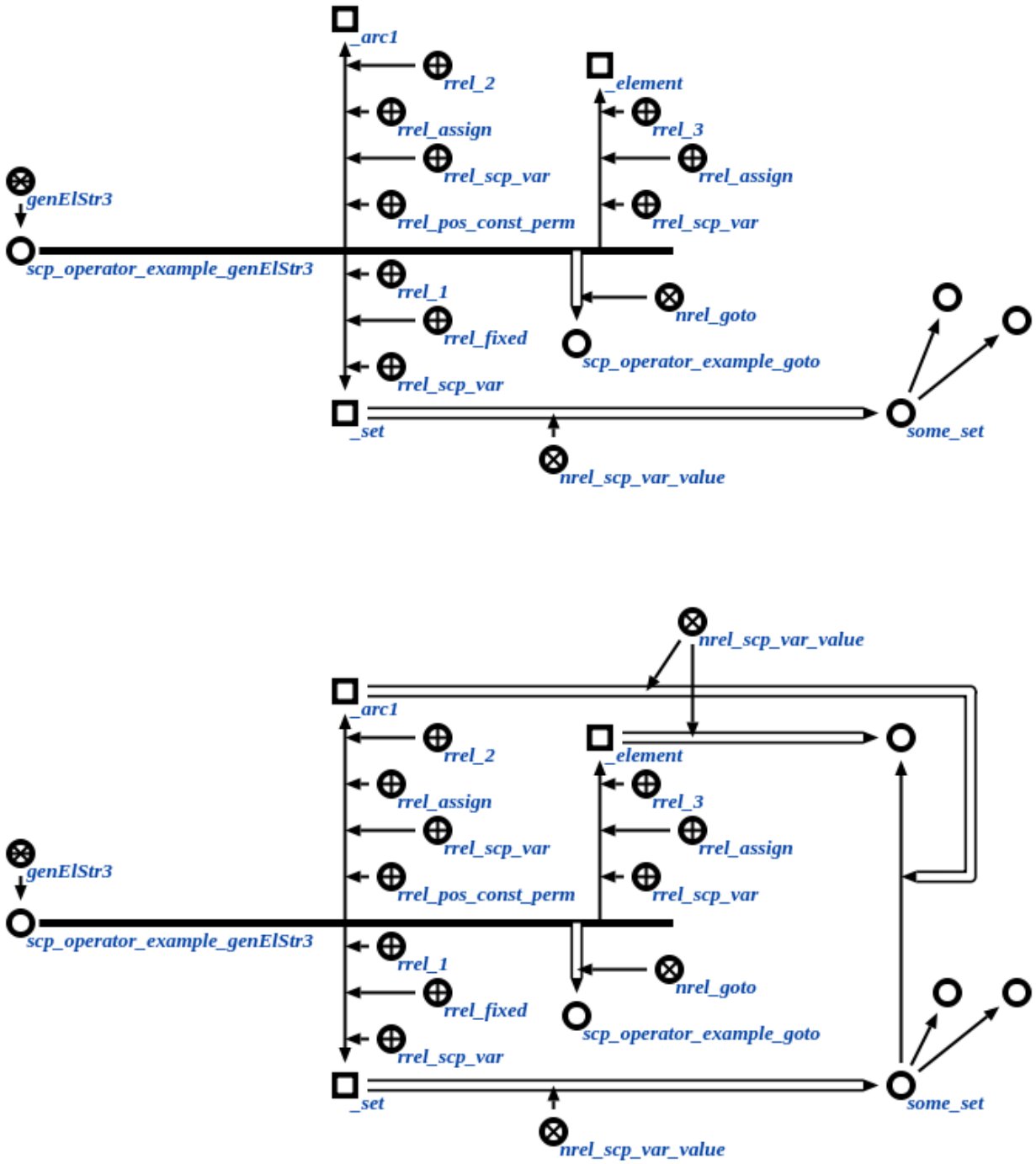



Рис. 3.9 Оператор генерации трехэлементной конструкции (Пример 1)

```
scp_operator_example_genElStr3 (*
        <- genElStr3;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _set;;
        -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var: _arc1;;
        -> rrel_3: rrel_fixed: rrel_scp_var: _element;;

        => nrel_goto: scp_operator_example_goto;;
*);;
```

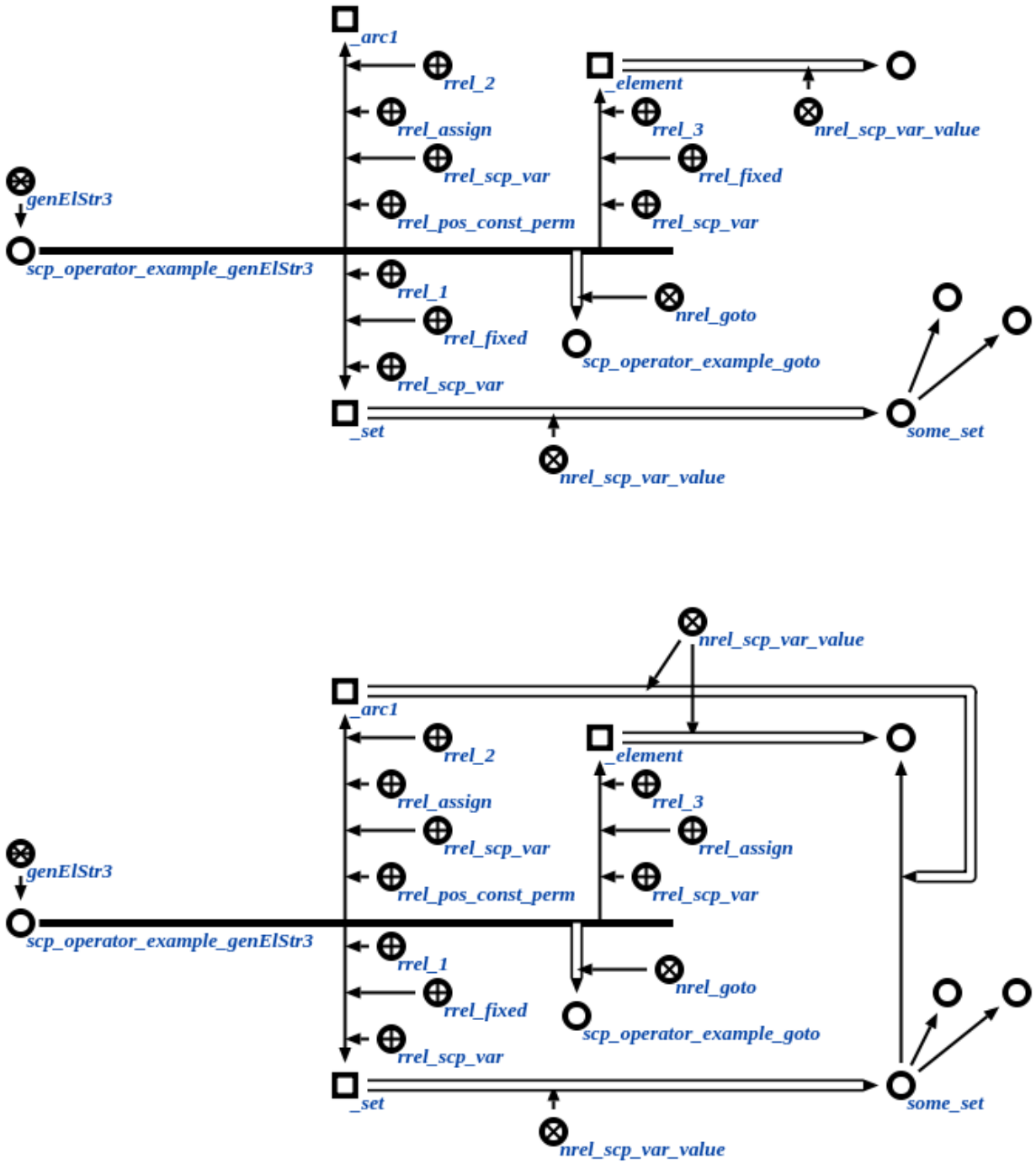



Рис. 3.10 Оператор генерации трехэлементной конструкции (Пример 2)

```
scp_operator_example_genElStr3 (*
        <- genElStr3;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _set;;
        -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var: _arc1;;
        -> rrel_3: rrel_fixed: rrel_scp_var: _element;;

        => nrel_goto: scp_operator_example_goto;;
*);;
```

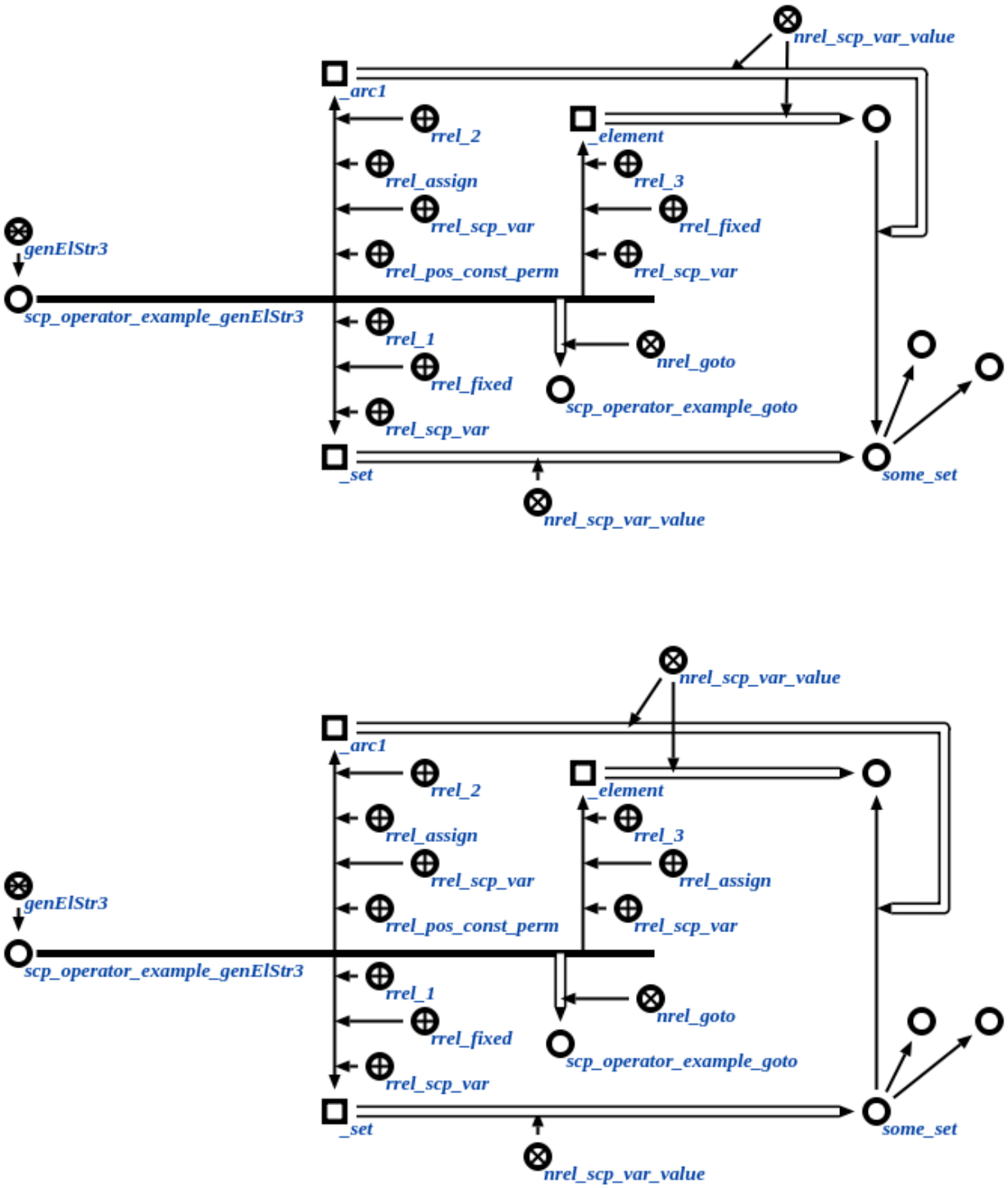



Рис. 3.11 Оператор генерации трехэлементной конструкции (Пример 3)

Операторы класса ***scp-оператор генерации пятиэлементной sc-конструкции*** описывают действие генерации всех либо некоторых *sc-элементов* указанного типа, составляющих *пятиэлементную sc-конструкцию*. Каждый оператор данного класса содержит пять операндов.

Первый операнд может иметь произвольный *тип значения scp-операнда'*, и соответствует первому элементу *пятиэлементной sc-конструкции*.

Второй операнд может иметь произвольный *тип значения scp-операнда'*, и соответствует второму элементу *пятиэлементной sc-конструкции*.

В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то генерируется *sc-коннектор* указанного типа. В случае, если данный операнд помечен как *scp-операнд с заданным значением'*, то *sc-коннектор* не генерируется, а будут изменены оба или один из инцидентных ему *sc-элемента*. То есть можно сказать, что будут «переставлены» начало и/или конец данного *sc-коннектора*. При этом физический адрес *sc-коннектора* может измениться, но конфигурация окружающей его семантической сети (входящие и выходящие *sc-коннекторы*, если такие были) будет полностью соответствовать той, что была до выполнения оператора.

Третий операнд может иметь произвольный *тип значения scp-операнда'*, и соответствует третьему элементу *пятиэлементной sc-конструкции*.

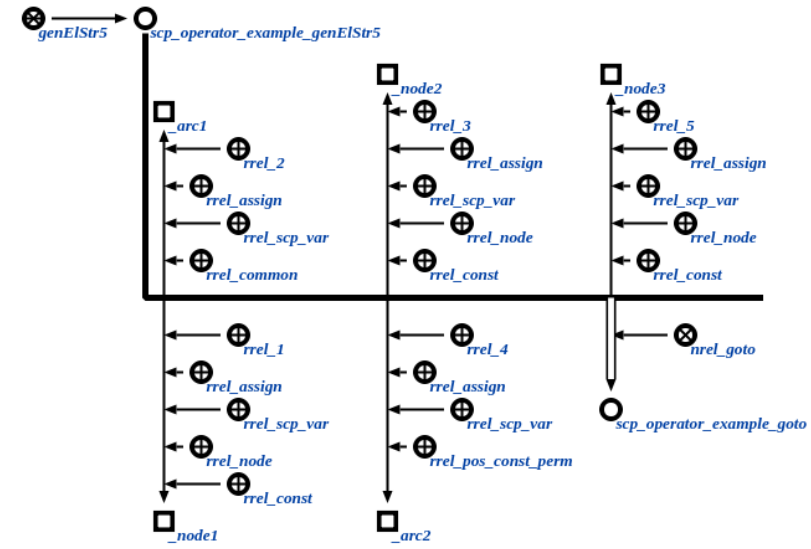
Четвертый операнд может иметь произвольный *тип значения scp-операнда'*, и соответствует четвертому элементу *пятиэлементной sc-конструкции*. Результаты выполнения оператора в зависимости от *типа значения scp-операнда'* определяются так же, как в случае со вторым операндом.

Пятый операнд может иметь произвольный *тип значения scp-операнда'*, и соответствует пятому элементу *пятиэлементной sc-конструкции*.

В результате выполнения оператора для всех операндов, помеченных как *scp-операнд со свободным значением'*, будет сгенерировано значение, т.е. новый *sc-элемент* соответствующего типа, с учетом уточнений типа. Тип генерируемого *sc-элемента* может быть уточнен при помощи подмножеств ролевого отношения *тип sc-элемента'*.

```
scp_operator_example_genElStr5 (*
        <- genElStr5;;
        -> rrel_1: rrel_assign: rrel_scp_var: rrel_node: rrel_const: _node1;;
        -> rrel_2: rrel_assign: rrel_common: rrel_scp_var: _arc1;;
        -> rrel_3: rrel_assign: rrel_scp_var: rrel_node: rrel_const: _node2;;
        -> rrel_4: rrel_assign: rrel_scp_var: rrel_pos_const_perm: _arc2;;
        -> rrel_5: rrel_assign: rrel_scp_var: rrel_node: rrel_const: _node3;;

        => nrel_goto: scp_operator_example_goto;;
*);;
```



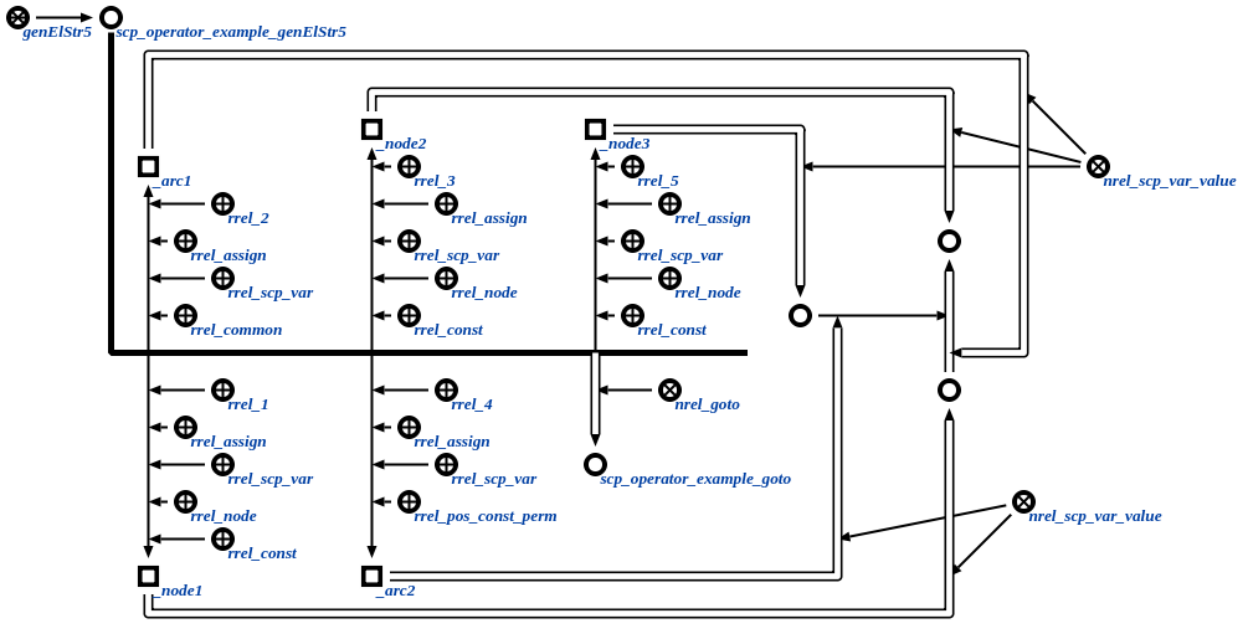




Рис. 3.12 Оператор генерации пятиэлементной конструкции (Пример 1)

```
scp_operator_example_genElStr5 (*
        <- genElStr5;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _node1;;
        -> rrel_2: rrel_assign: rrel_common: rrel_scp_var: _arc1;;
        -> rrel_3: rrel_fixed: rrel_scp_var: _node2;;
        -> rrel_4: rrel_assign: rrel_scp_var: rrel_pos_const_perm: _arc2;;
        -> rrel_5: rrel_assign: rrel_scp_var: rrel_node: rrel_const: _node3;;

        => nrel_goto: scp_operator_example_goto;;
*);;
```

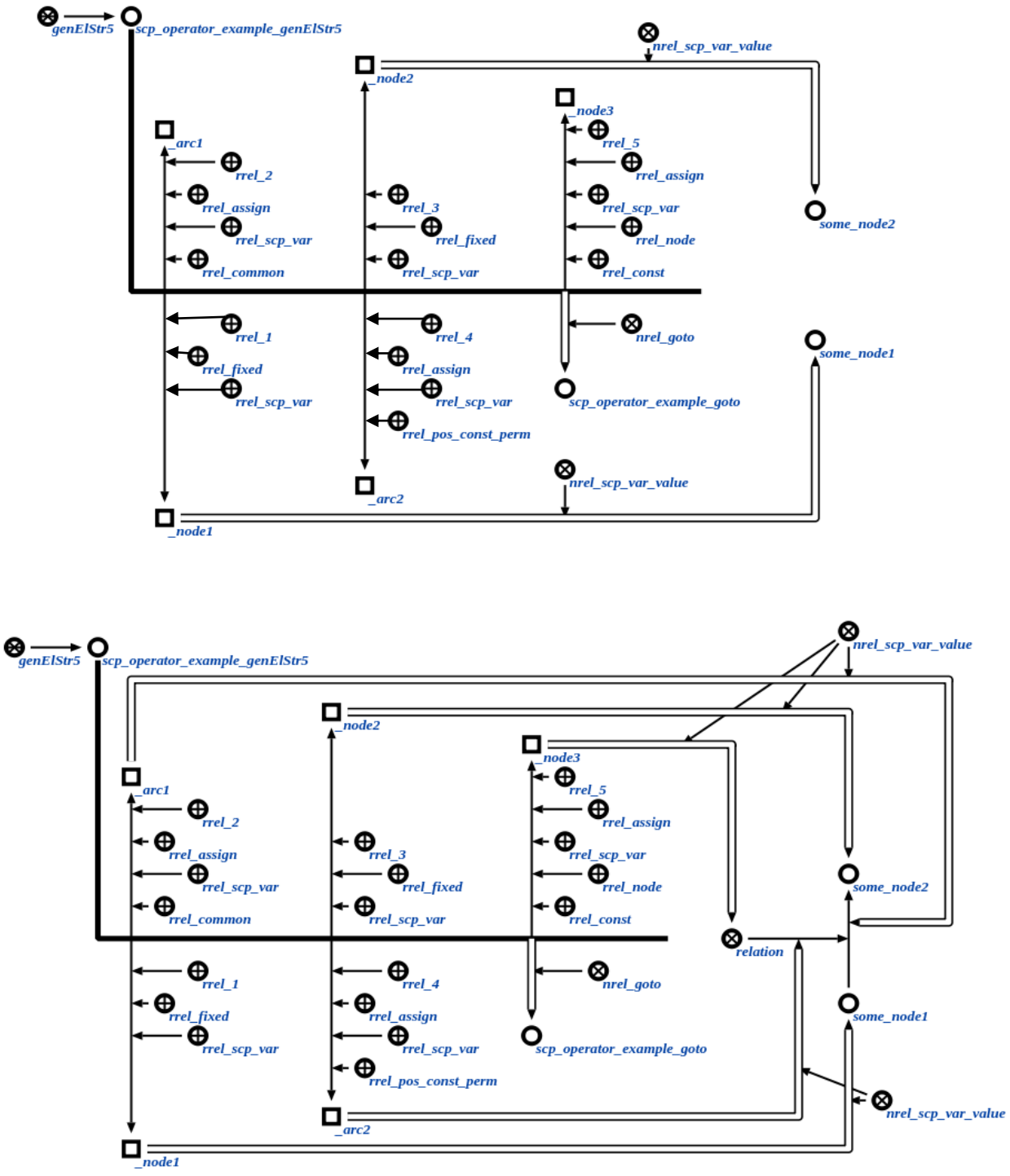
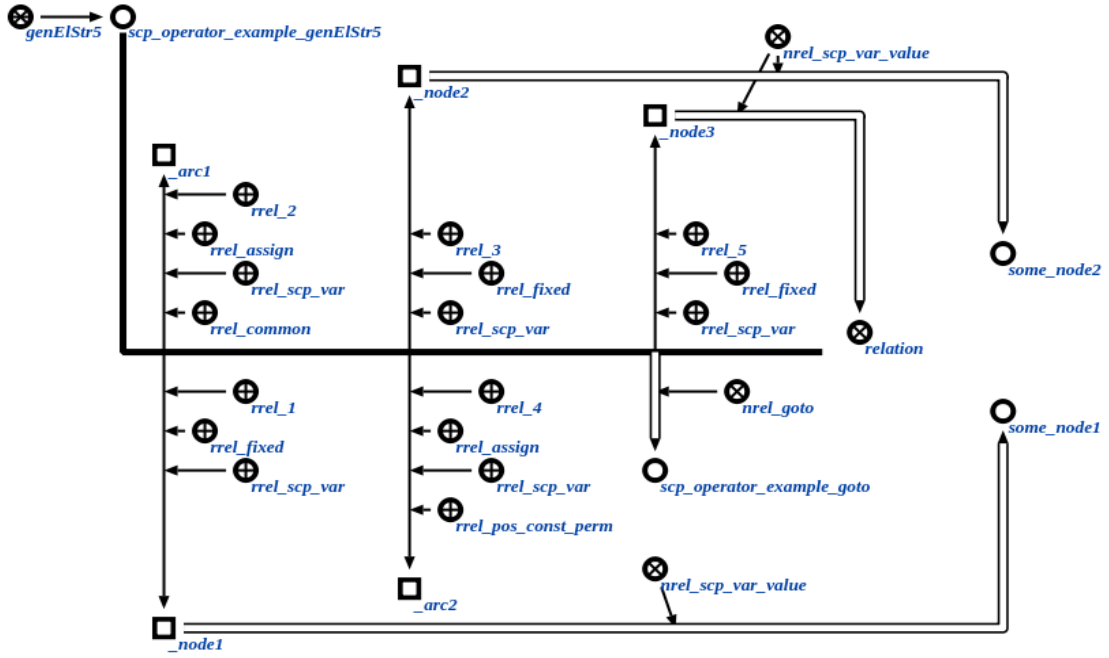



Рис. 3.13 Оператор генерации пятиэлементной конструкции (Пример 2)

```
scp_operator_example_genElStr5 (*
        <- genElStr5;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _node1;;
        -> rrel_2: rrel_assign: rrel_common: rrel_scp_var: _arc1;;
        -> rrel_3: rrel_fixed: rrel_scp_var: _node2;;
        -> rrel_4: rrel_assign: rrel_scp_var: rrel_pos_const_perm: _arc2;;
        -> rrel_5: rrel_fixed: rrel_scp_var: _node3;;

        => nrel_goto: scp_operator_example_goto;;
*);;
```



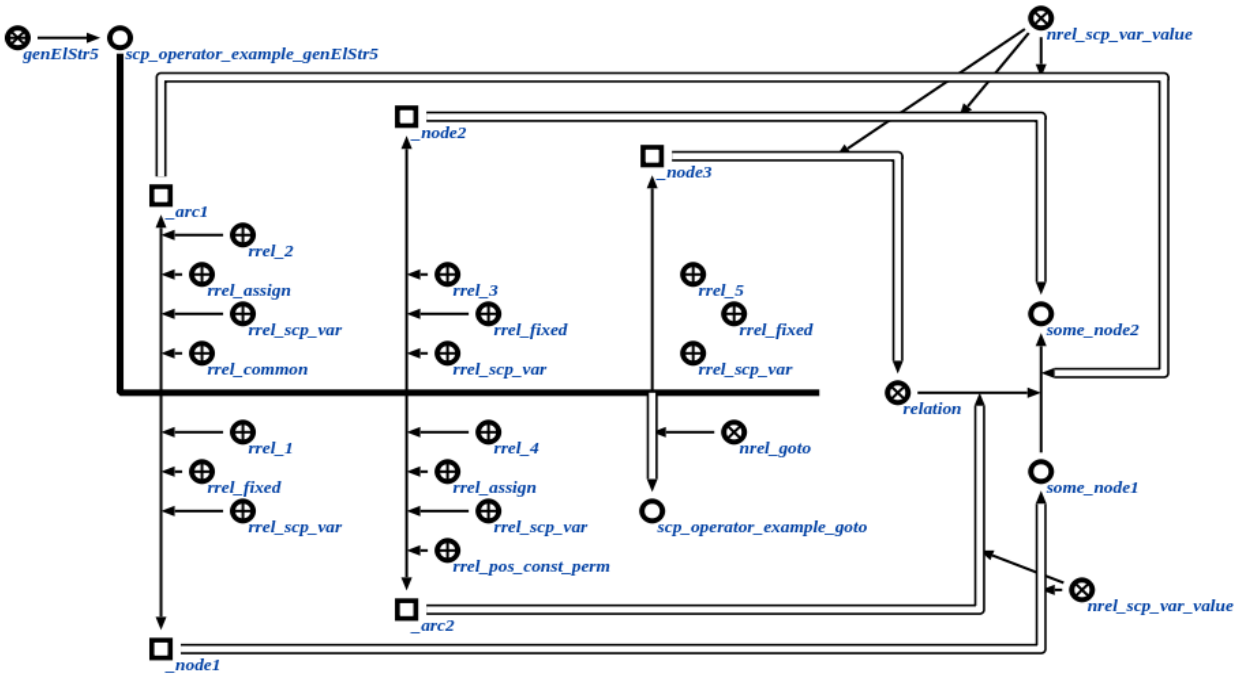




Рис. 3.14 Оператор генерации пятиэлементной конструкции (Пример 3)

Операторы класса ***scp-оператор генерации sc-конструкции по произвольному образцу*** описывают действие генерации *sc-элементов* указанного типа, составляющих *sc-конструкцию* произвольного вида. Данный оператор может применяться и для генерации стандартных *sc-конструкций*, однако его использование в таком случае нецелесообразно, поскольку для этого имеются специальные операторы. Каждый оператор данного класса содержит четыре операнда.

Первый операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-узел*, обозначающий шаблон генерации, содержащий хотя бы один *переменный sc-элемент.* Константные *sc-элементы* не могут иметь быть *sc-коннекторами*, поскольку в таком случае нарушается семантика самого понятия *sc-коннектора*, гласящая, что *sc-коннектор* имеет в один момент времени ровно два инцидентных ему *sc-элемента*.

Второй операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного оператора является знак множества, представляющего собой результат генерации, то есть множество ориентированных пар соответствия всех переменных из шаблона генерации их сгенерированным значениям (если значения не были заданы в третьем операнде). В случае, если данный операнд помечен как *scp-операнд с заданным значением'*, то необходимо явно указать пары соответствия, в которых под атрибутом *1'* указывается переменный *sc-элемент* из шаблона генерации (помеченный, как *scp-константа'*), под атрибутом *2' - scp-переменные',* в значения которых будут записаны сгенерированные *sc-элементы*, соответствующие указанным переменным *sc-элементам* из шаблона генерации (помеченные, как *scp-переменная'* и *scp-операнд со свободным значением'*). При этом пары соответствия для остальных переменных из шаблона не будут явно формироваться в памяти. Указанное множество может быть пустым, тогда пары соответствия не будут формироваться вообще, но конструкция, соответствующая шаблону генерации будет сгенерирована в любом случае. В случае, если данный операнд помечен как *scp-операнд со свободным значением',* то будут сформированы пары соответствия сгенерированным значениям для всех *sc-переменных* из шаблона генерации. Значением операнда станет знак множества указанных пар.

Третий операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного оператора является знак множества параметров генерации, то есть ориентированных пар соответствия некоторых переменных

из шаблона генерации их значениям, в случае, если значения заранее известны. При этом каждый из элементов пар должен быть помечен, как *scp-операнд с заданным значением'*, так же при необходимости должен быть указан *тип scp-операнда'*. Указанное множество может быть пустым.

Четвертый операнд может иметь произвольный *тип значения scp-операнда'*. Данный операнд является необязательным и может быть опущен. Значением данного операнда является знак множества всех *sc-элементов*, сгенерированных в результате выполнения данного *scp-оператора*. В случае, если операнд помечен, как *операнд с заданным значением'*, то будет дополнено множество, являющееся значением данного операнда, если как *операнд со свободным значением'*, то множество будет сгенерировано и сформировано с нуля.

```
scp_operator_example_sys_gen (*
        <- sys_gen;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_assign: rrel_scp_var: _el2;;
        -> rrel_3: rrel_fixed: rrel_scp_var: _el3;;
        => nrel_goto: next_operator;;
*);;
```

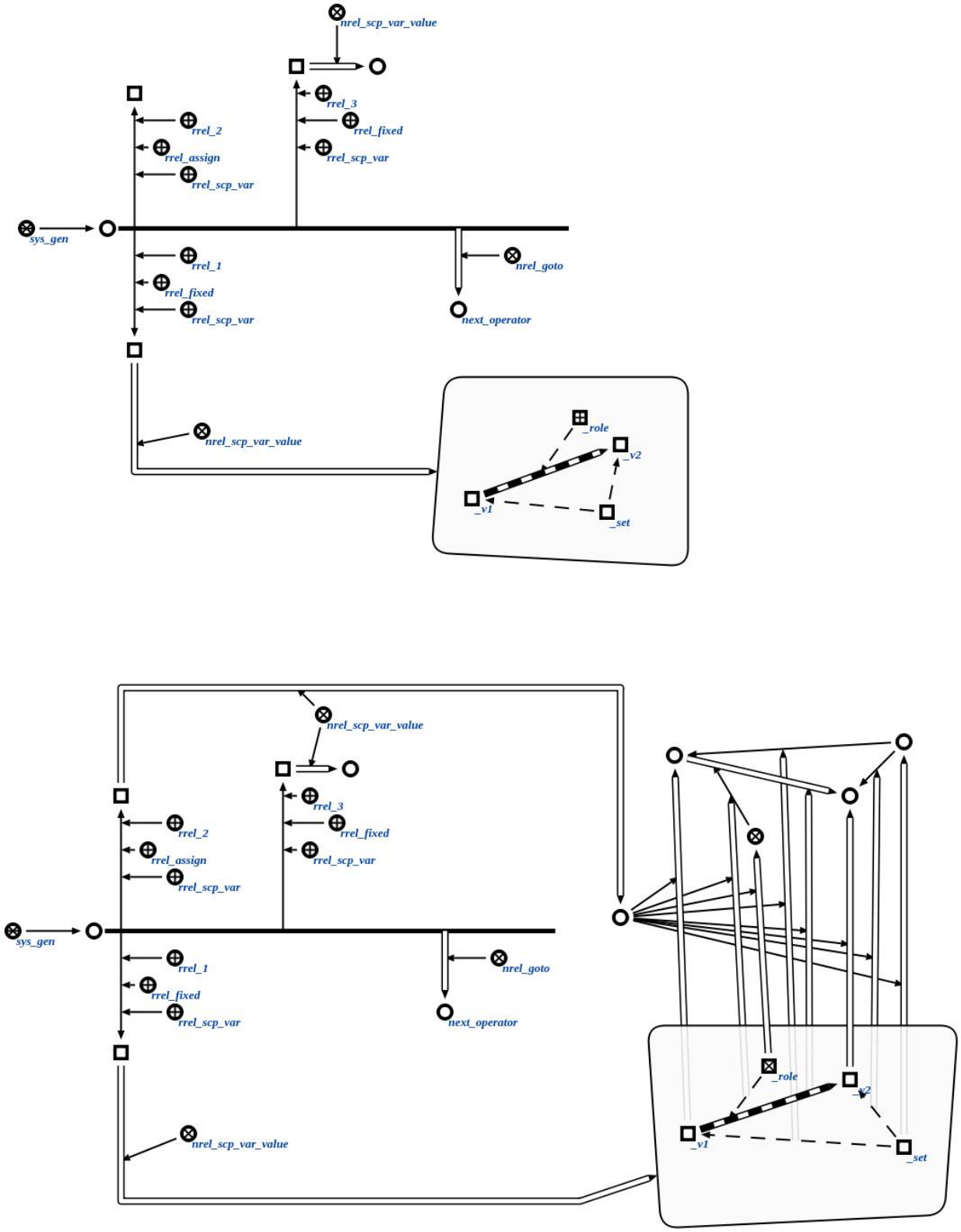



Рис. 3.15 Оператор генерации конструкции по образцу (Пример 1)

```
scp_operator_example_sys_gen (*
        <- sys_gen;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_var: _el2;;
        -> rrel_3: rrel_fixed: rrel_scp_var: _el3;;
        -> rrel_4: rrel_fixed: rrel_scp_var: _el4;;
        => nrel_goto: next_operator;;
*);;
```

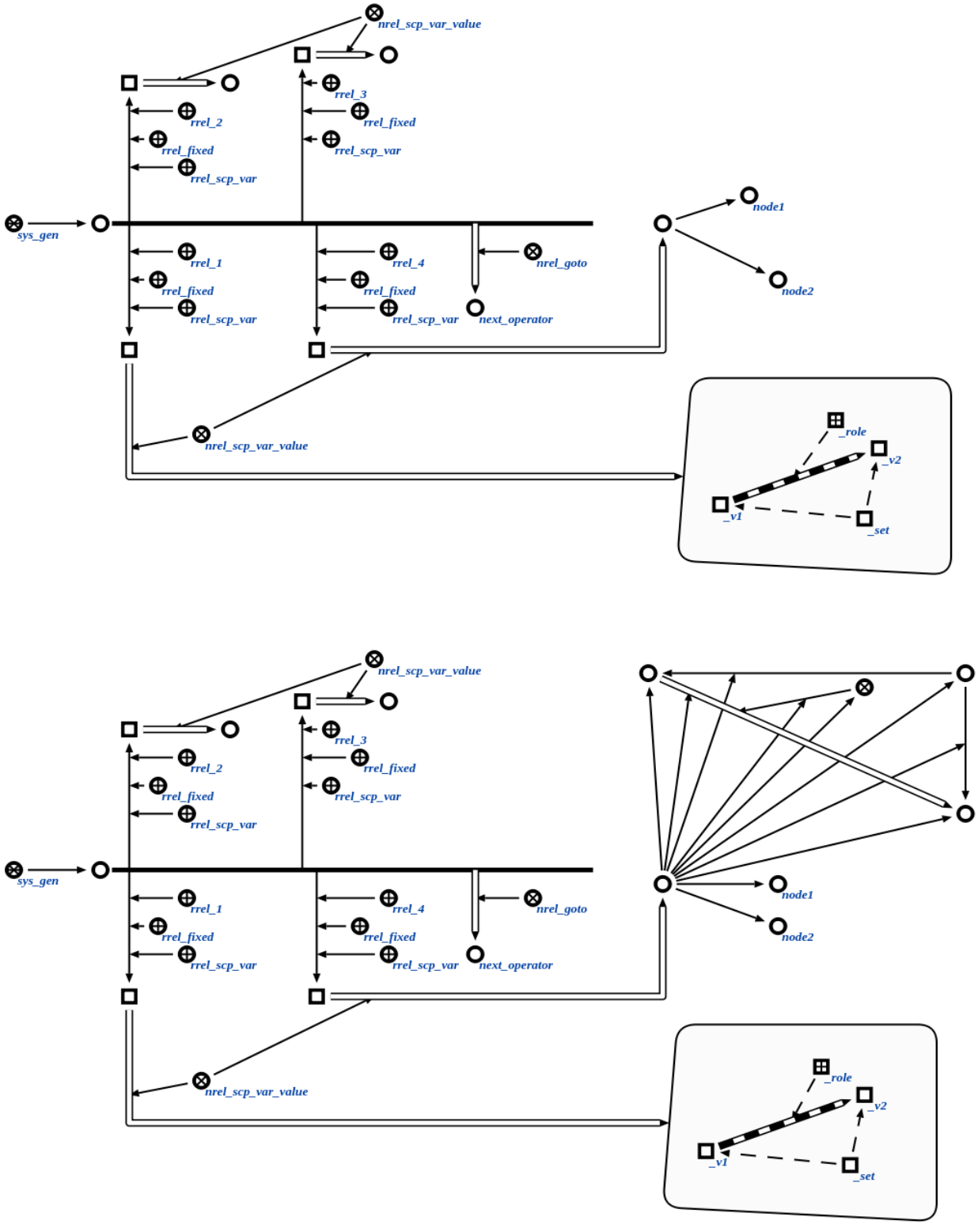



Рис. 3.16 Оператор генерации конструкции по образцу (Пример 2)

```
scp_operator_example_sys_gen (*
        <- sys_gen;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_var: _el2;;
        -> rrel_3: rrel_fixed: rrel_scp_var: _el3;;
        -> rrel_4: rrel_assign: rrel_scp_var: _el4;;
        => nrel_goto: next_operator;;
*);;
```

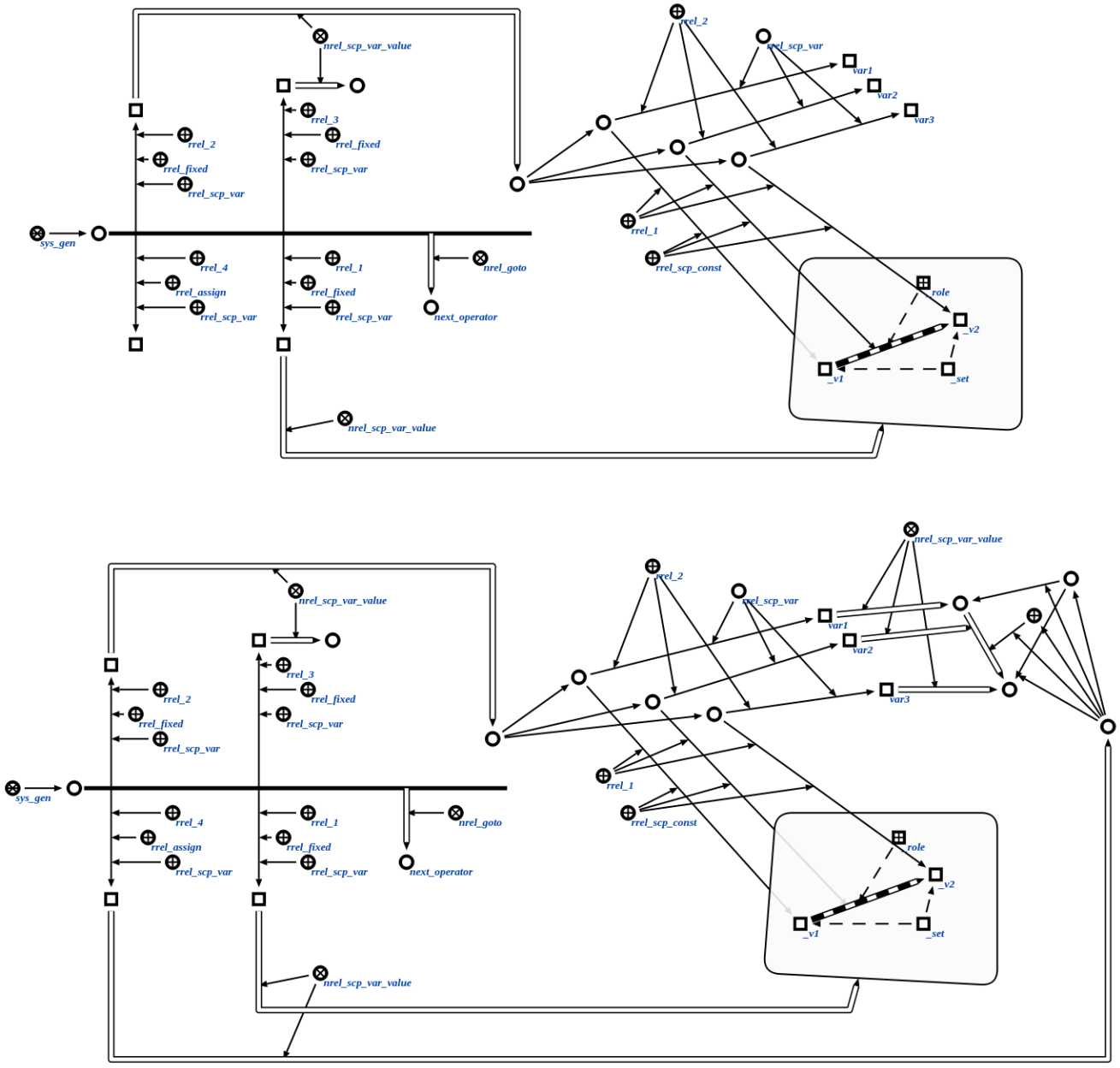



Рис. 3.17 Оператор генерации конструкции по образцу (Пример 3)

```
scp_operator_example_sys_gen (*
        <- sys_gen;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_assign: rrel_scp_var: _el2;;
        -> rrel_3: rrel_fixed: rrel_scp_var: _el3;;
        => nrel_goto: next_operator;;
*);;
```

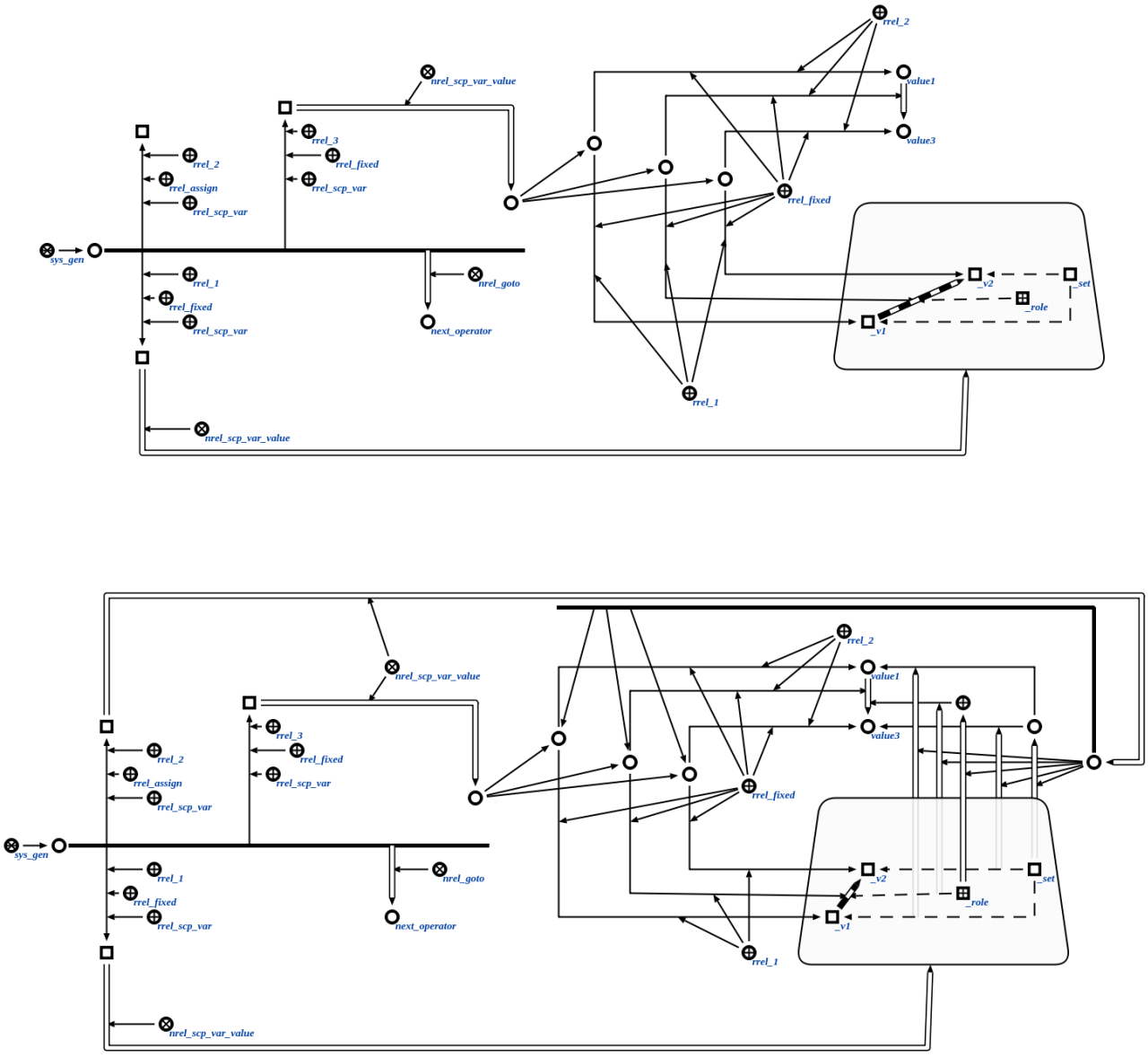



Рис. 3.18 Оператор генерации конструкции по образцу (Пример 4)

## 3.2 Scp-операторы ассоциативного поиска sc-конструкций

Операторы класса ***scp-оператор ассоциативного поиска sc-конструкций*** описывают действие поиска *sc-элементов*, удовлетворяющих некоторому заданному образцу. Образец может представлять собой как одну из *sc-конструкций стандартного вида*, так и *sc-конструкцию нестандартного вида*. Операторы класса предполагают поиск только по знаниям, явно

представленным в базе знаний, и не предполагают дополнительных действий, таких как, например, логический вывод, позволяющий получить дополнительные сведения на основе имеющихся знаний. Операторы класса предполагают ветвление программы в зависимости от того, оказалась ли успешной попытка поиска. Указание следующих операторов может осуществляться при помощи подмножеств отношения *следующий оператор**. В случае, если программист заранее уверен в успешности поиска, может быть использовано само по себе отношение *следующий оператор**.

Данный класс *scp-операторов* разбивается на следующие подклассы:

- ***scp-оператор поиска трехэлементной sc-конструкции;***
- ***scp-оператор поиска пятиэлементной sc-конструкции;***
- ***scp-оператор поиска трехэлементной sc-конструкции с формированием множеств;***
- ***scp-оператор поиска пятиэлементной sc-конструкции с формированием множеств;***
- ***scp-оператор поиска sc-конструкции по произвольному образцу.***

Операторы класса ***scp-оператор поиска трехэлементной sc-конструкции*** описывают действие поиска некоторых из *sc-элементов*, составляющих *трехэлементную sc-конструкцию*, а также проверки инцидентности заданных sc-элементов. Каждый оператор данного класса содержит три операнда.

Первый, второй и третий операнды соответствуют первому, второму и третьему элементам *трехэлементной sc-конструкции*.

Каждый из операндов может иметь произвольный *тип значения scp-операнда'*. При выполнении оператора осуществляется попытка поиска *sc-элементов*, соответствующих операндам, помеченным, как *операнд со свободным значением'*. Однако запрещена ситуация, когда все операнды помечены, как *операнд со свободным значением'*.

В случае, если операнд, соответствующий *sc-коннектору*, помечен как *операнд с заданным значением',* а операнды, соответствующие инцидентным *sc-коннектору* элементам *трехэлементной sc-конструкции* помечены как *операнд со свободным значением',* то значениями данных операндов станут соответствующие *sc-элементы*, инцидентные заданному *sc-коннектору*. Таким образом можно, например, найти начальный и конечный элементы *sc-дуги*.

В случае, если операнд, соответствующий *sc-коннектору*, помечен как *операнд с заданным значением'*, и операнды, соответствующие инцидентным *sc-коннектору* элементам *трехэлементной sc-конструкции* также помечены как *операнд с заданным значением'*, то будет осуществлена проверка,

действительно ли известный *sc-коннектор* и *sc-элемент (sc-элементы)* инцидентны указанным образом. Значения соответствующих операндов изменены не будут, а переход к следующему оператору будет осуществлен по одному из отношений *следующий оператор при успешном выполнении** или *следующий оператор при безуспешном выполнении** в зависимости от успешности указанной проверки. Таким образом можно, например, проверить, являются ли найденные ранее *sc-элементы* началом и концом заданной *sc-дуги*.

Важно заметить, что в случае, если заданному критерию поиска соответствуют несколько возможных результатов, то будет выбран один произвольный вариант.

```
scp_operator_example_searchElStr3 (*
        <- searchElStr3;;
        -> rrel_1: rrel_fixed: rrel_scp_const: some_node_1;;
        -> rrel_2: rrel_assign: rrel_common: rrel_const: rrel_scp_var: _arc;;
        -> rrel_3: rrel_fixed: rrel_scp_const: rrel_node: some_node_2;;
        => nrel_then: next_operator_1;;
        => nrel_else: next_operator_2;;
*);;
```

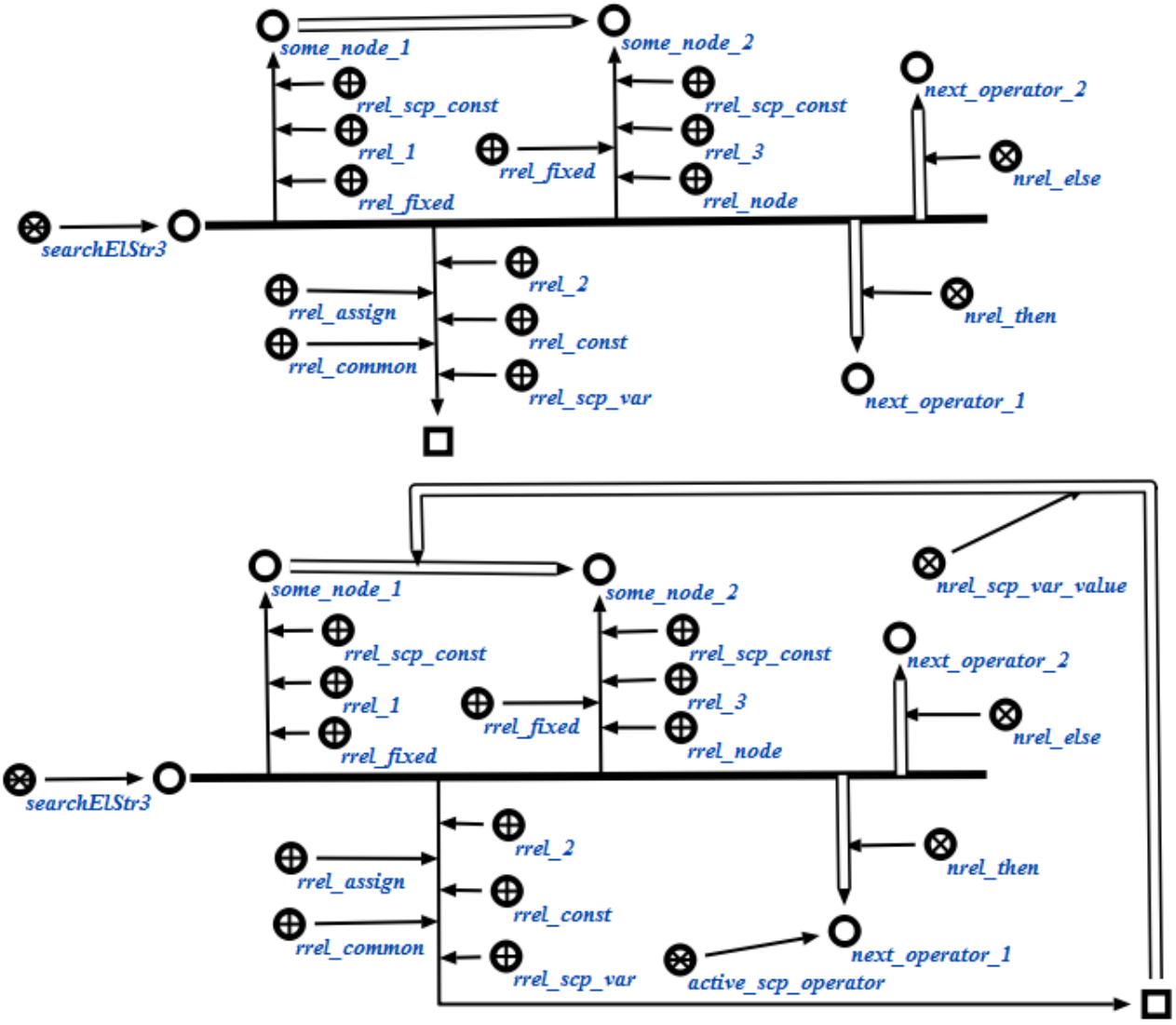



Рис. 3.19 Оператор поиска трехэлементной конструкции (Пример 1)

```
scp_operator_example_searchElStr3 (*
        <- searchElStr3;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: come_arc;;
        -> rrel_3: rrel_assign: rrel_scp_var: _el2;;
        => nrel_then: next_operator_1;;
        => nrel_else: next_operator_2;;
*);;
```

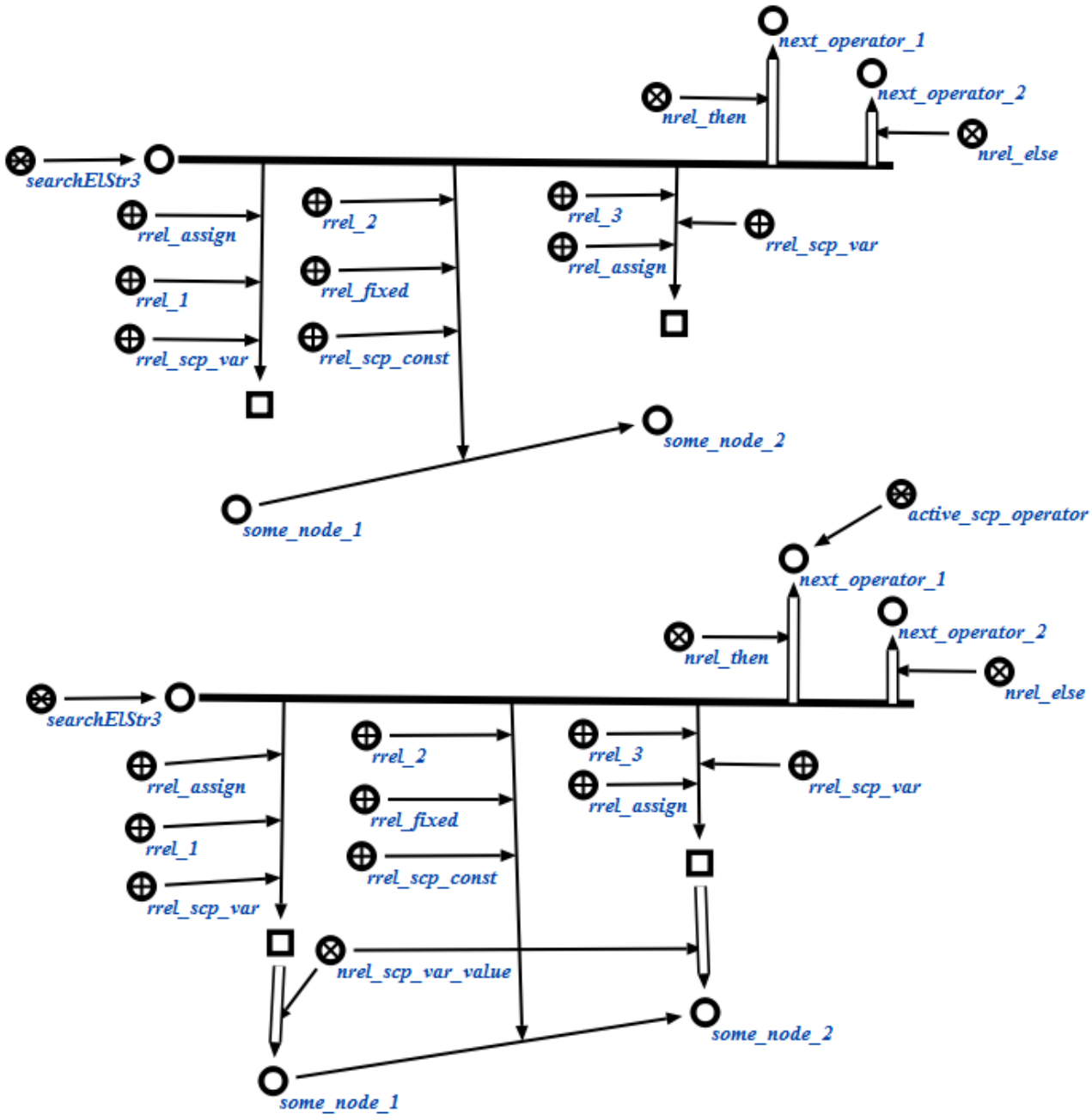



Рис. 3.20 Оператор поиска конструкции по образцу (Пример 2)

Операторы класса ***scp-оператор поиска пятиэлементной sc-конструкции*** описывают действие поиска некоторых из *sc-элементов*,

составляющих *пятиэлементную sc-конструкцию*, а также проверки инцидентности заданных sc-элементов. Каждый оператор данного класса содержит пять операндов.

Первый, второй, третий, четвертый и пятый операнды соответствуют первому, второму, третьему, четвертому и пятому элементам *пятиэлементной sc-конструкции*.

Каждый из операндов может иметь произвольный *тип значения scp-операнда'*. При выполнении оператора осуществляется попытка поиска *sc-элементов*, соответствующих операндам, помеченным, как *операнд со свободным значением'*. Однако запрещена ситуация, когда все операнды помечены, как *операнд со свободным значением'*.

Различные варианты выполнения операторов данного класса в зависимости от комбинации *типов значений scp-операндов'* аналогичны вариантам выполнения *scp-оператор поиска трехэлементной sc-конструкции.*

Важно заметить, что в случае, если заданному критерию поиска соответствуют несколько возможных результатов, то будет выбран один произвольный вариант.

```
scp_operator_example_searchElStr5 (*
        <- searchElStr5;;
        -> rrel_1: rrel_assign: rrel_scp_const: some_set;;
        -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var: _arc;;
        -> rrel_3: rrel_assign: rrel_scp_var: _el3;;
        -> rrle_4: rrel_assign: rrel_scp_var: rrel_pos_const_perm: _arc2;;
        -> rrel_5: rrel_assogn: rrel_scp_var: _el5;;
        => nrel_then: next_operator_1;;
        => nrel_else: next_operator_2;;
*);;
```

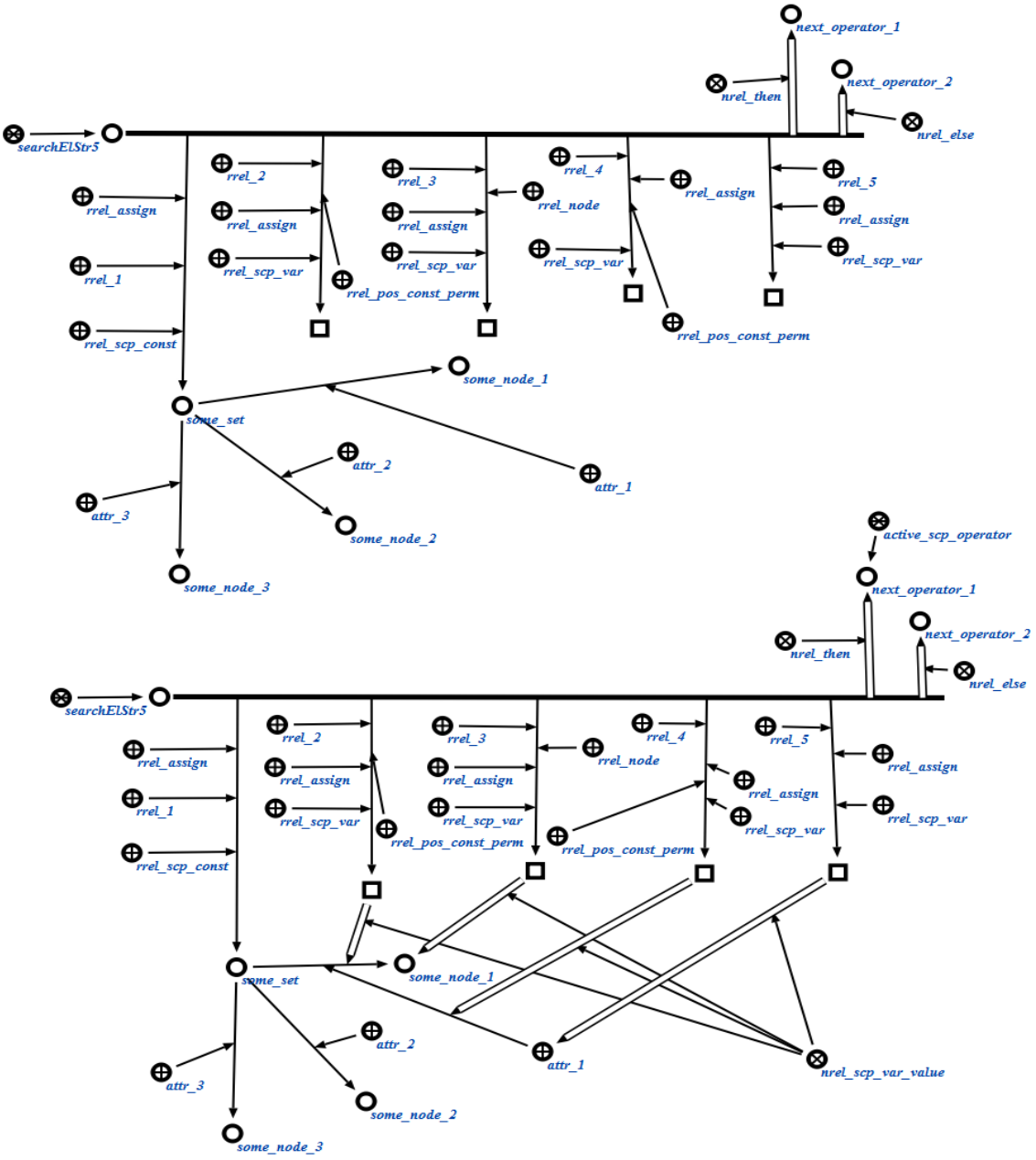
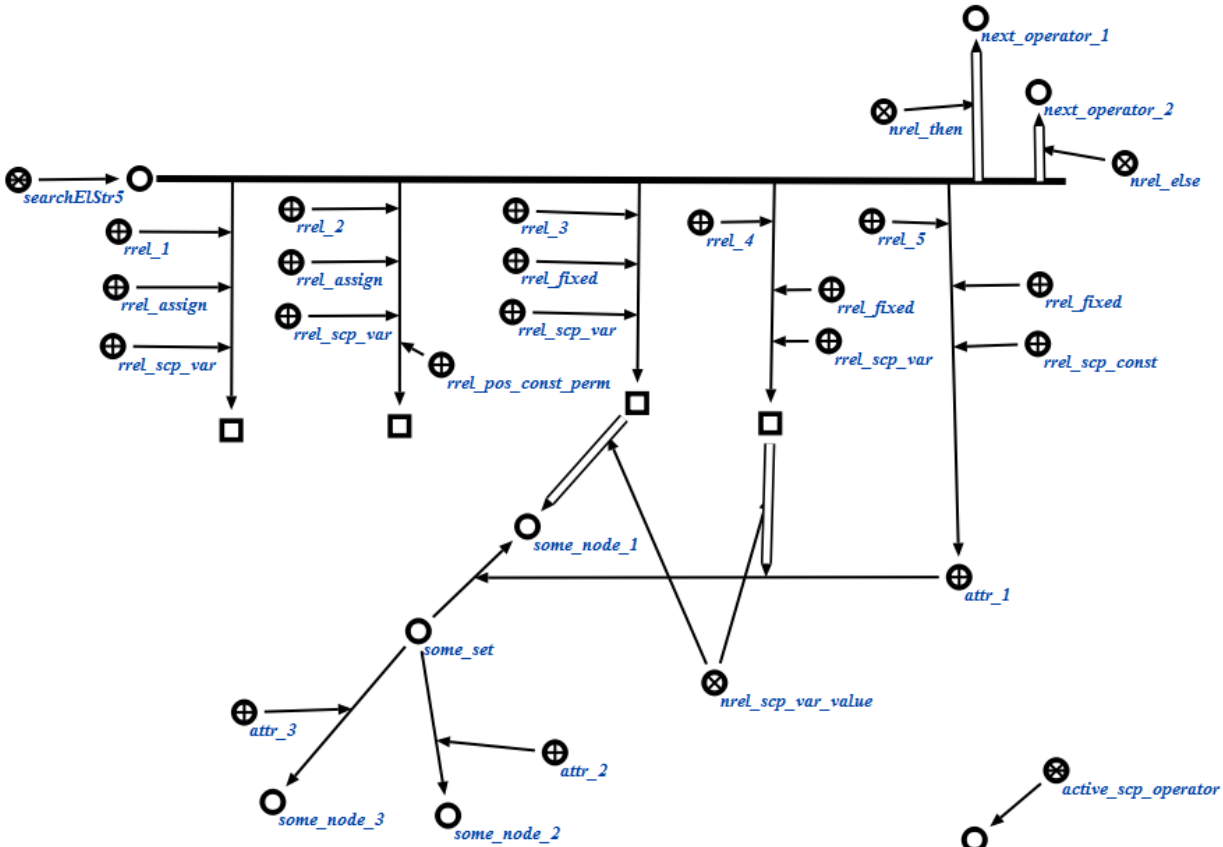



Рис. 3.21 Оператор поиска пятиэлементной конструкции (Пример 1)

```
scp_operator_example_searchElStr5 (*
        <- searchElStr5;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var: _arc;;
```

```
        -> rrel_3: rrel_fixed: rrel_scp_var: _el3;;
        -> rrle_4: rrel_fixed: rrel_scp_var: _arc2;;
        -> rrel_5: rrel_fixed: rrel_scp_const: attr_1;;
        => nrel_then: next_operator_1;;
        => nrel_else: next_operator_2;;
*);;
```



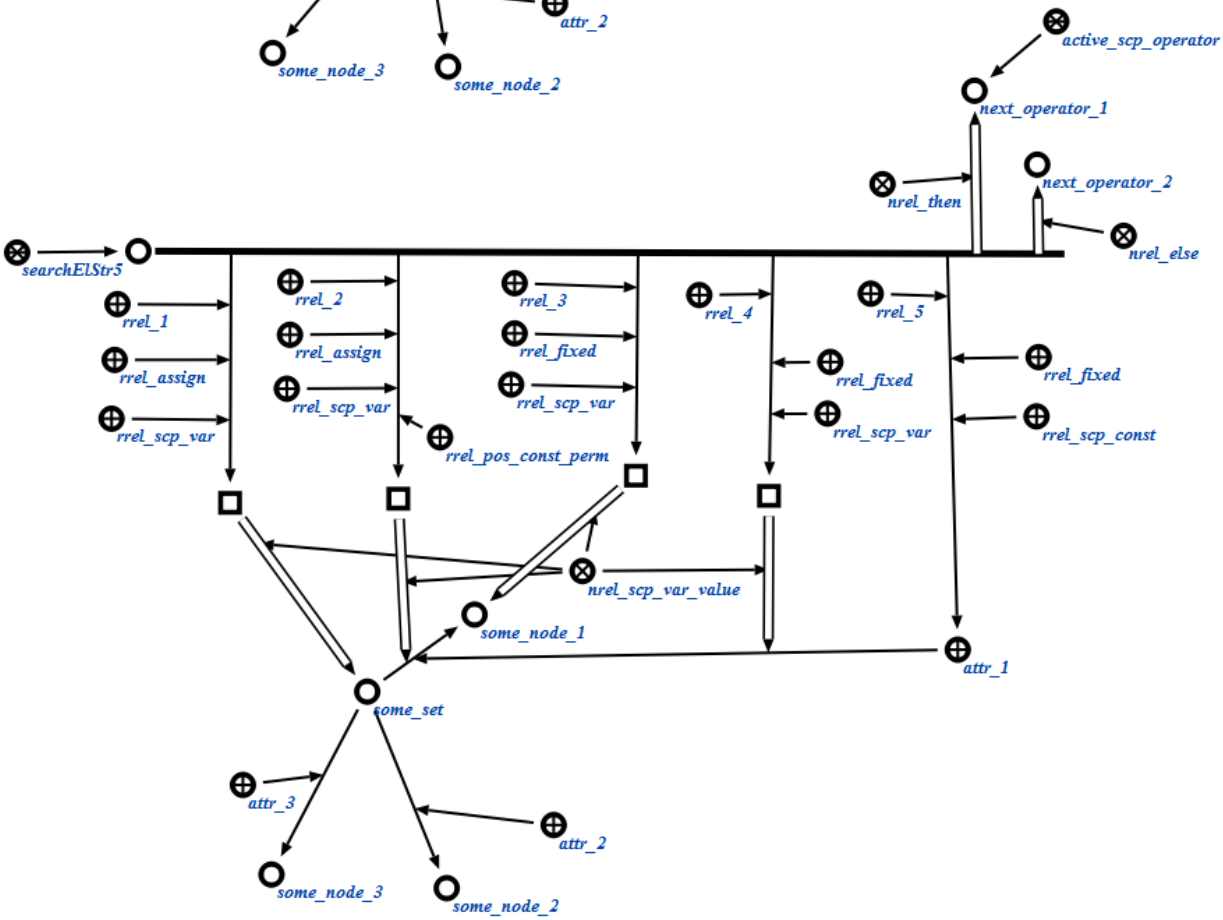



Рис. 3.22 Оператор поиска пятиэлементной конструкции (Пример 2)

Операторы класса ***scp-оператор поиска трехэлементной sc-конструкции с формированием множеств*** описывают действие формирования множества *sc-элементов*, соответствующих указанным элементам *трехэлементной sc-конструкции*. Каждый оператор данного класса содержит четыре или пять операндов.

Первый, второй и третий операнды соответствуют первому, второму и третьему элементам *трехэлементной sc-конструкции*. Указанные операнды в данном случае можно назвать основными.

Первый и третий операнд могут иметь произвольный *тип значения scp-операнда'*, но как минимум один из них должен быть помечен как *операнд с заданным значением'*. Второй операнд должен быть помечен как *операнд со свободным значением'*.

Помимо трех основных операндов каждый оператор данного класса содержит один или два дополнительных операнда, значениями которых являются множества, содержащие все *sc-элементы*, которые могли бы стать значениями соответствующего основного операнда при заданных условиях поиска. При этом множество возможных значений операнда, помеченного ролевым отношением *1'*, соответствует множеству, которое является значением операнда, помеченного ролевым отношением *формируемое множество 1'* и т.д. Если операнд, соответствующий такому множеству, помечен как *операнд со свободным значением'*, то множество будет сформировано заново (сгенерирован новый *sc-элемент*), в противном случае будут добавлены элементы во множество, являющееся значением данного операнда.

В качестве примера применения данного класса *scp-операторов* можно рассматривать задачу копирования множества (т.е. создание множества, содержащего те же элементы). Также Операторы класса применяются для организации циклических переборов каких-либо элементов *sc-множества*.

```
scp_operator_example_searchSetStr3 (*
        <- searchSetStr3;;
        -> rrel_1: rrel_fixed: rrel_scp_const: some_set;;
        -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var: _arc;;
        -> rrel_3: rrel_assign: rrel_node: rrel_scp_var: _el3;;
        -> rrel_set_3: rrel_assign: rrel_scp_var: _set3;;
        => nrel_then: next_operator_1;;
        => nrel_else: next_operator_2;;
*);;
```

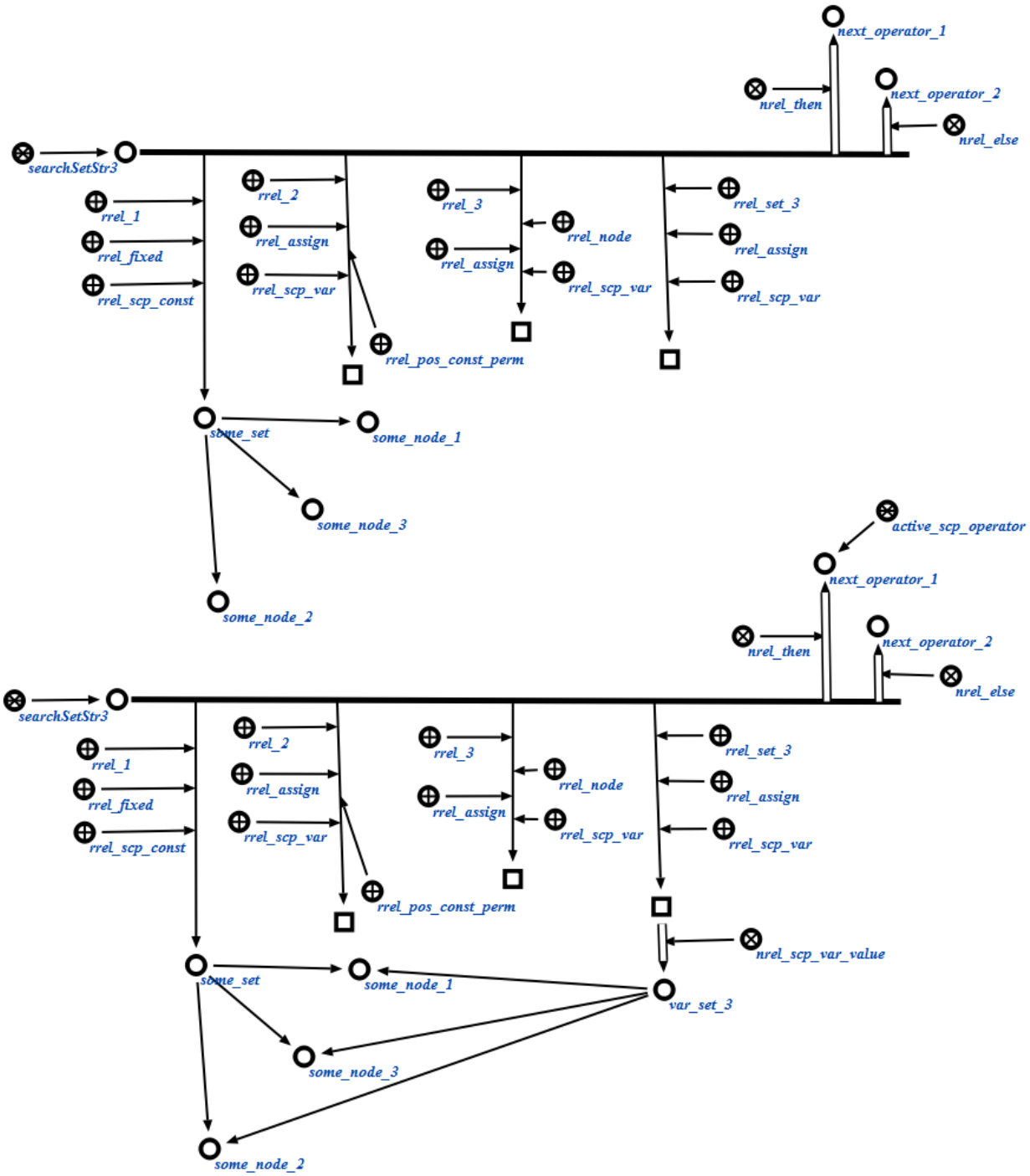



Рис. 3.23 Оператор поиска трехэлементной конструкции с формированием множеств

Операторы класса ***scp-оператор поиска пятиэлементной sc-конструкции с формированием множеств*** описывают действие формирования множества *sc-элементов*, соответствующих указанным элементам *пятиэлементной sc-конструкции*. Каждый оператор данного класса содержит от шести до десяти операндов.

Первый, второй, третий, четвертый и пятый операнды соответствуют первому, второму, третьему, четвертому и пятому элементам *пятиэлементной sc-конструкции*. Указанные операнды в данном случае можно назвать основными.

Первый, третий и пятый операнд могут иметь произвольный *тип значения scp-операнда'*, но как минимум один из них должен быть помечен как *операнд с заданным значением'*. Второй и четвертый операнд должны быть помечены как *операнд со свободным значением'*.

Помимо пяти основных операндов каждый оператор данного класса содержит от одного до четырех дополнительных операндов, значениями которых являются множества, содержащие все *sc-элементы*, которые могли бы стать значениями соответствующего основного операнда при заданных условиях поиска. При этом множество возможных значений операнда, помеченного ролевым отношением *1'*, соответствует множеству, которое является значением операнда, помеченного ролевым отношением *формируемое множество 1'* и т.д. Если операнд, соответствующий такому множеству, помечен как *операнд со свободным значением'*, то множество будет сформировано заново (сгенерирован новый *sc-элемент*), в противном случае будут добавлены элементы во множество, являющееся значением данного операнда.

Операторы класса применяются в тех же случаях, что и *scp-операторы поиска трехэлементной sc-конструкции с формированием множеств.*

```
scp_operator_example_searchSetStr5 (*
        <- searchSetStr5;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var: _arc;;
        -> rrel_3: rrel_assign: rrel_scp_var: _el3;;
        -> rrel_4: rrel_assign: rrel_scp_var: rrel_pos_const_perm: _arc2;;
        -> rrel_5: rrel_assign: rrel_scp_var: _el5;;
        -> rrel_set_2: rrel_fixed: rrel_scp_const: some_node;;
        -> rrel_set_5: rrel_assign: rrel_scp_var: _set5;;
        => nrel_then: next_operator_1;;
        => nrel_else: next_operator_2;;
*);;
```

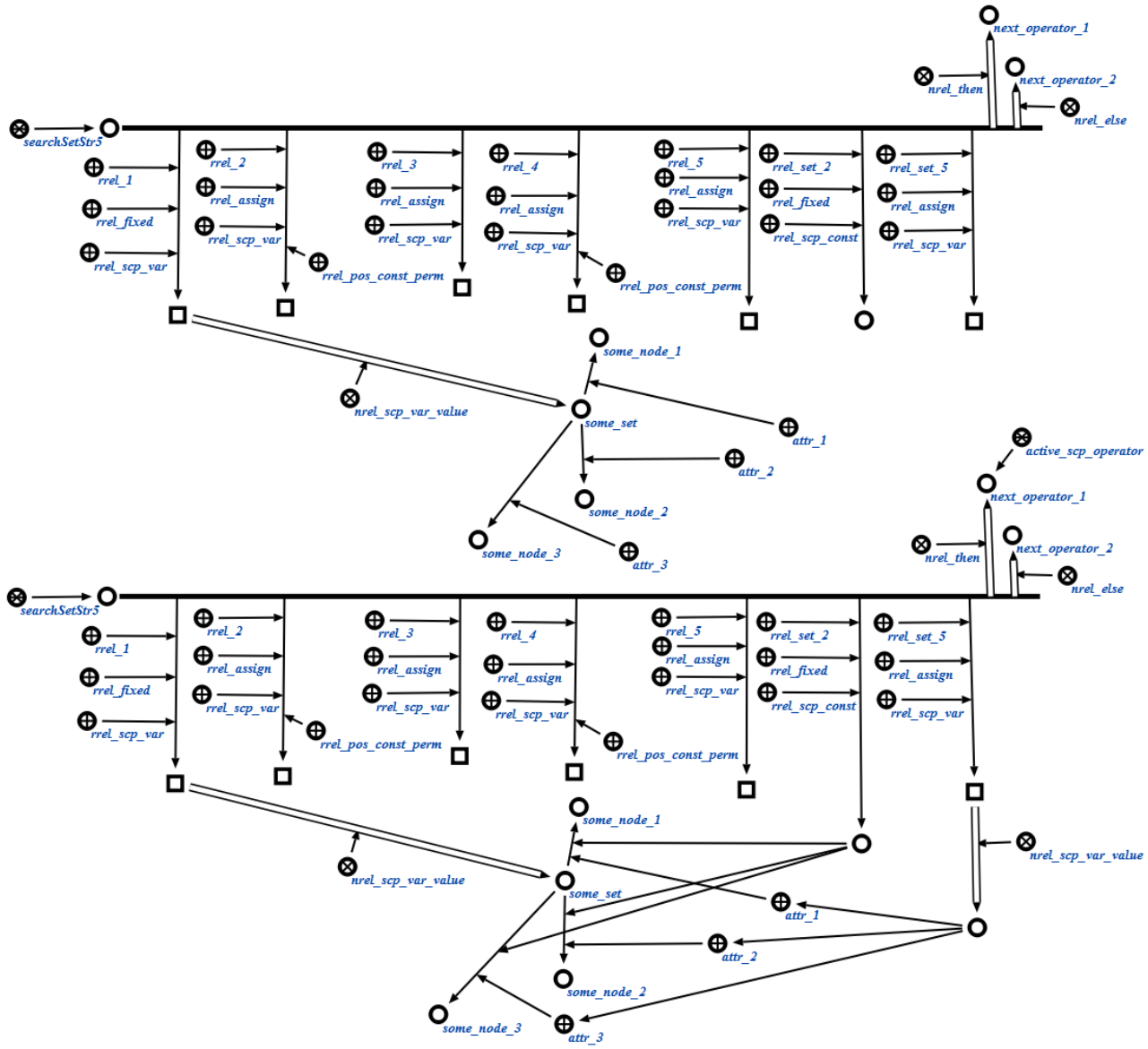



Рис. 3.24 Оператор поиска пятиэлементной конструкции с формированием множеств

Операторы класса ***scp-оператор поиска sc-конструкции по произвольному образцу*** описывают действие поиска всех либо некоторых *sc-элементов* указанного типа, составляющих *sc-конструкцию* произвольного

вида. Данный оператор может применяться для поиска на основе стандартных *sc-конструкций*, однако его использование в таком случае нецелесообразно, поскольку для этого имеются специальные операторы. Каждый оператор данного класса содержит четыре операнда.

Первый операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-узел*, обозначающий шаблон поиска, содержащий хотя бы один *переменный sc-элемент* и один *константный sc-элемент. Константные sc-элементы* не могут иметь быть *sc-коннекторами*, поскольку в таком случае нарушается семантика самого понятия *sc-коннектора*, гласящая, что *sc-коннектор* имеет в один момент времени ровно два инцидентных ему *sc-элемента*. *Константные sc-элементы* могут отсутствовать, если в третьем операнде указано значение хотя бы для одного *переменного sc-элемента* из шаблона поиска.

Второй операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного оператора является знак множества, содержащего все возможные результаты поиска, каждый из которых является множеством ориентированных пар соответствия всех переменных из шаблона поиска их найденным значениям (если значения не были заданы в третьем операнде). В случае, если данный операнд помечен как *scp-операнд с заданным значением'*, то необходимо явно указать пары соответствия, в которых под атрибутом *1'* указывается переменный *sc-элемент* из шаблона поиска (помеченный, как *scp-константа'*), под атрибутом *2' - scp-переменные'*, в значения которых будут записаны найденные *sc-элементы*, соответствующие указанным переменным *sc-элементам* из шаблона поиска (помеченные, как *scp-переменная'* и *scp-операнд со свободным значением'*). При этом пары соответствия для остальных переменных из шаблона не будут явно формироваться в памяти. Элемент пары под атрибутом *2'* может быть также помечен как *формируемое множество'*, тогда значением соответствующей *scp-переменной'* станет множество, содержащее все возможные *sc-элементы*, соответствующие указанному переменному *sc-элементу* из шаблона с учетом критериев поиска. В противном случае значением указанной *scp-переменной'* станет один из *sc-элементов*, удовлетворяющих условиям поиска.

Указанное множество может быть пустым, тогда пары соответствия не будут формироваться вообще, а оператор может быть использован для проверки факта, имеется ли в памяти конструкция изоморфная заданному шаблону поиска с учетом параметров. В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то будут сформированы пары соответствия сгенерированным значениям для всех *sc-переменных* из шаблона

генерации. Значением операнда станет знак множества результатов поиска, каждый из которых представляет собой возможную комбинацию указанных пар соответствия переменных и их значений, удовлетворяющую критериям поиска. Количество возможных комбинаций зависит от конкретного вида шаблона поиска.

Третий операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного оператора является знак множества параметров поиска, то есть ориентированных пар соответствия некоторых переменных из шаблона поиска их значениям, в случае, если значения заранее известны. При этом каждый из элементов пар должен быть помечен, как *scp-операнд с заданным значением'*, так же при необходимости должен быть указан *тип scp-операнда'*. Указанное множество может быть пустым.

Четвертый операнд может иметь произвольный *тип значения scp-операнда'*. Данный операнд является необязательным и может быть опущен. Значением данного операнда является знак множества всех *sc-элементов*, найденных в результате выполнения данного *scp-оператора*. В случае, если операнд помечен, как *операнд с заданным значением'*, то будет дополнено множество, являющееся значением данного операнда, если как *операнд со свободным значением'*, то множество будет сгенерировано и сформировано с нуля. Указанное множество не содержит кратных элементов.

```
scp_operator_example_sys_search (*
        <- sys_search;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_assign: rrel_scp_var: _el2;;
        -> rrel_3: rrel_fixed: rrel_scp_var: _el3;;
        -> rrel_4: rrel_assign: rrel_scp_var: _el4;;
        => nrel_goto: next_operator;;
*);;
```

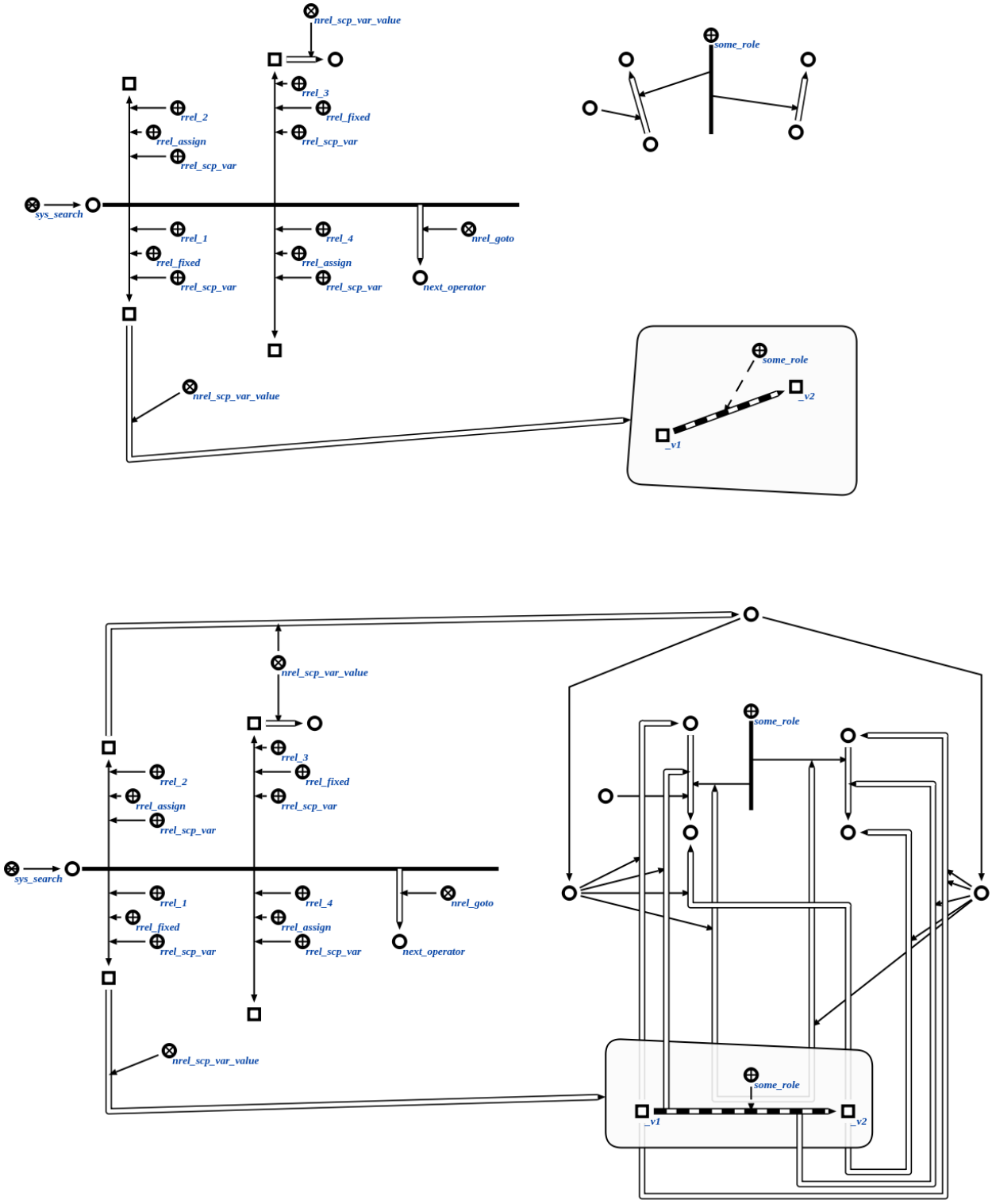



Рис. 3.25 Оператор поиска конструкции по образцу (Пример 1)

```
scp_operator_example_sys_search (*
        <- sys_search;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_var: _el2;;
        -> rrel_3: rrel_fixed: rrel_scp_var: _el3;;
        -> rrel_4: rrel_assign: rrel_scp_var: _el4;;
        => nrel_goto: next_operator;;
*);;
```

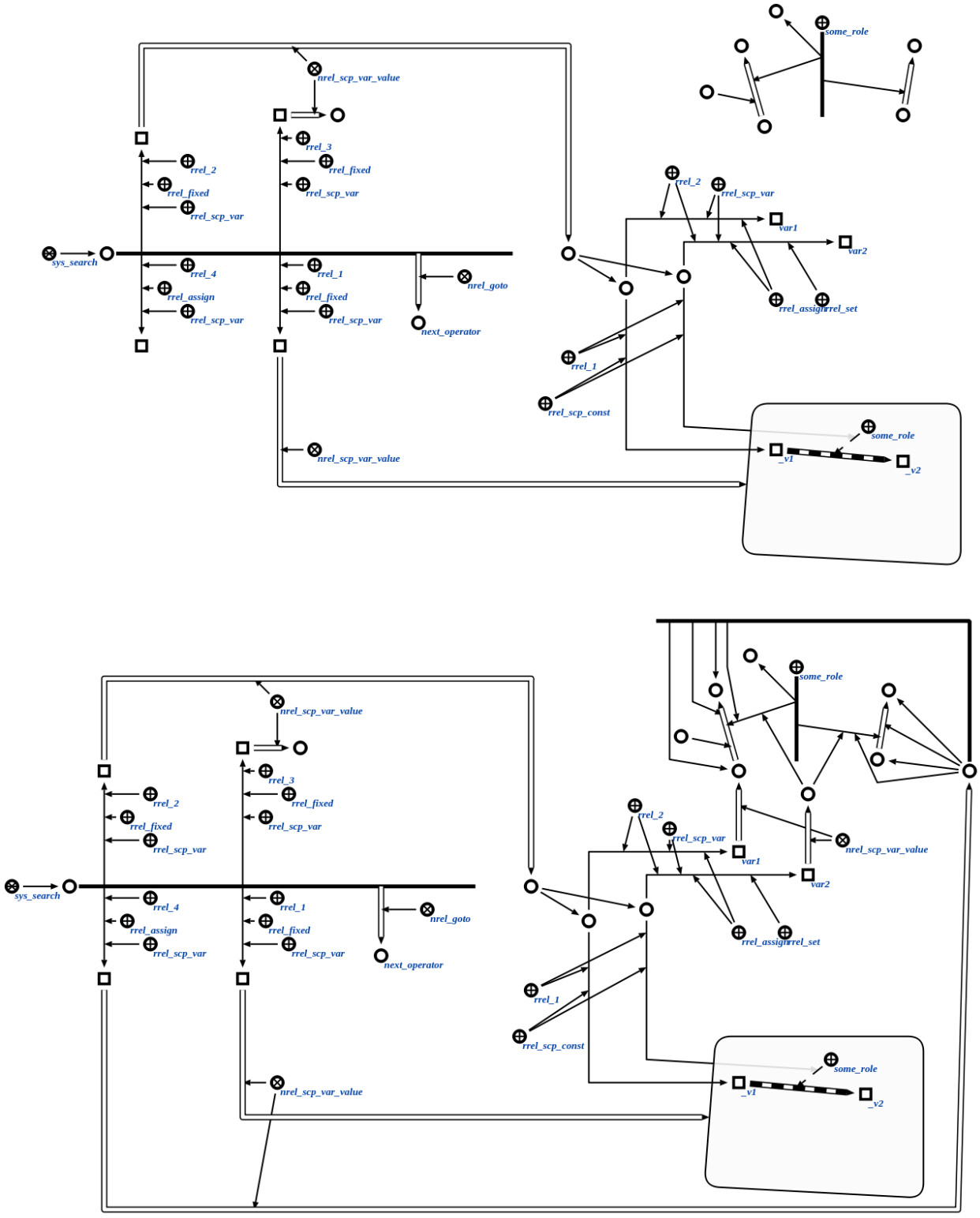



Рис. 3.26 Оператор поиска конструкции по образцу (Пример 2)

### 3.3 Scp-операторы удаления sc-конструкций

Операторы класса ***scp-оператор удаления sc-конструкций*** описывают действие удаления множества *sc-элементов*, удовлетворяющих некоторому условию. При этом следует заметить, что при указании того факта, что значение операнда должно быть удалено (при помощи ролевого отношения *удаляемый sc-элемент'*), удален будет физически именно тот элемент, который является значением операнда, а не только связка отношения *значение**, связывающая операнд с его значением (если такая есть). Таким образом, удалена может быть и *scp-константа'*. Следует помнить, что при удалении *sc-элемента* автоматически удаляются все инцидентные ему *sc-коннекторы*, а также *sc-коннекторы*, инцидентные указанным *sc-коннекторам* и так далее рекурсивно.

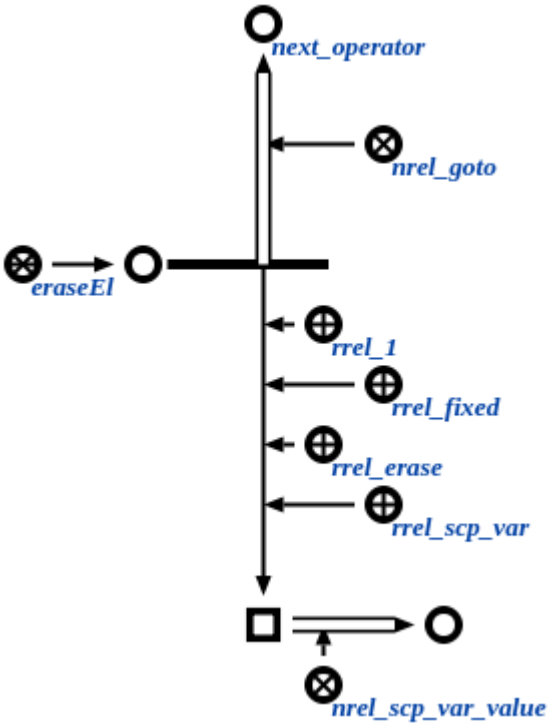
Операторы класса не предполагают ветвления программы в зависимости от выполнения дополнительных условий, поэтому указание следующих операторов осуществляется только при помощи отношения *следующий оператор**, без подмножеств.

Данный класс *scp-операторов* разбивается на следующие подклассы:

- ***scp-оператор удаления одноэлементной sc-конструкции;***
- ***scp-оператор удаления трехэлементной sc-конструкции;***
- ***scp-оператор удаления пятиэлементной sc-конструкции;***
- ***scp-оператор удаления множества элементов трехэлементной sc-конструкции;***
- ***scp-оператор удаления множества элементов пятиэлементной sc-конструкции.***

Операторы класса ***scp-оператор удаления одноэлементной sc-конструкции*** описывают действие удаления одного заданного *sc-элемента*. Каждый оператор данного класса содержит один операнд, обязательно помеченный как *scp-операнд с заданным значением'*.

```
scp_operator_example_eraseEl (*
        <- eraseEl;;
        -> rrel_1: rrel_fixed: rrel_erase: rrel_scp_var: _el1;;
        => nrel_goto: next_operator;;
*);;
```



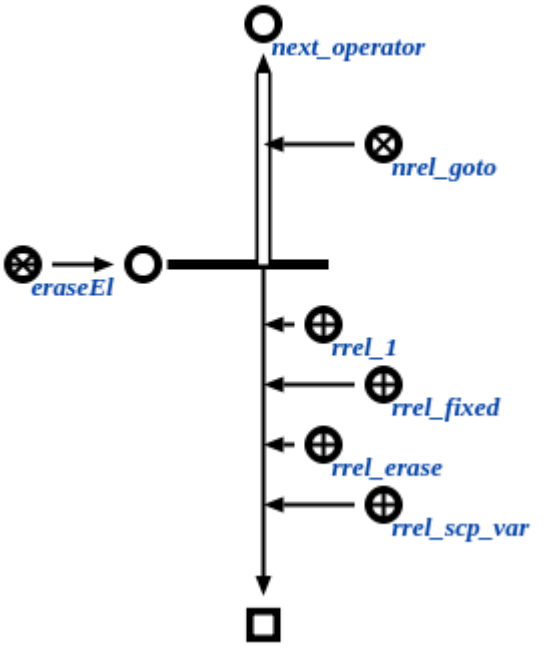
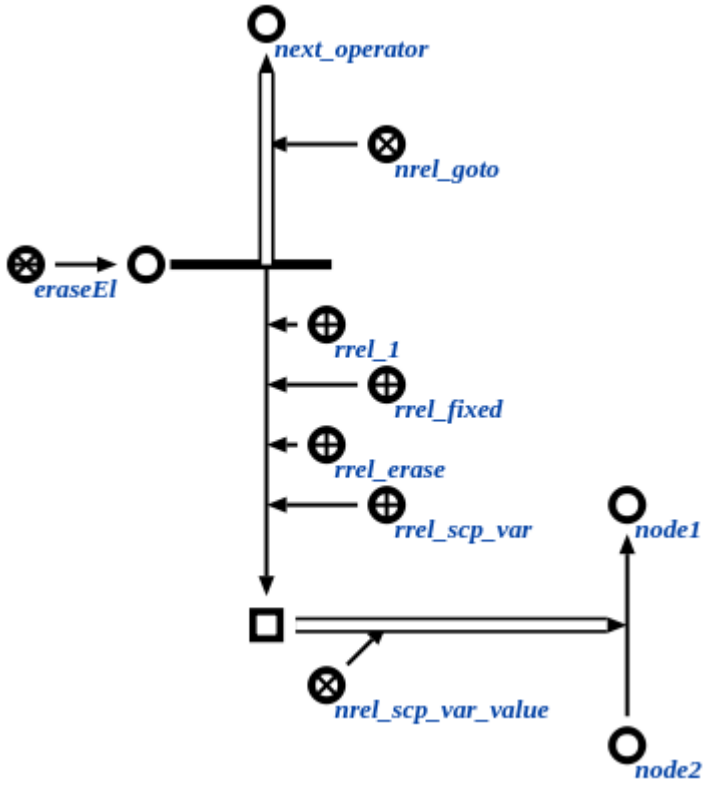




Рис. 3.27 Оператор удаления одноэлементной конструкции (Пример 1)

```
scp_operator_example_eraseEl (*
        <- eraseEl;;
        -> rrel_1: rrel_fixed: rrel_erase: rrel_scp_var: _el1;;
        => nrel_goto: next_operator;;
*);;
```



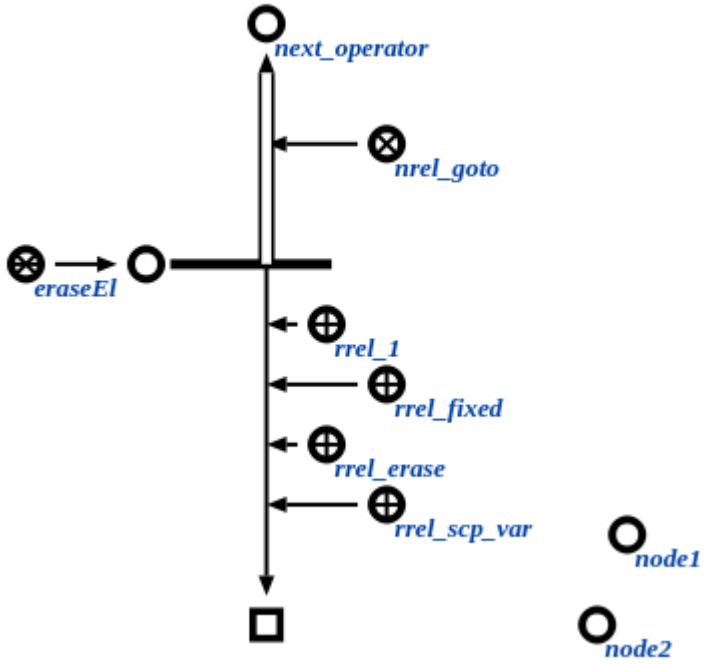
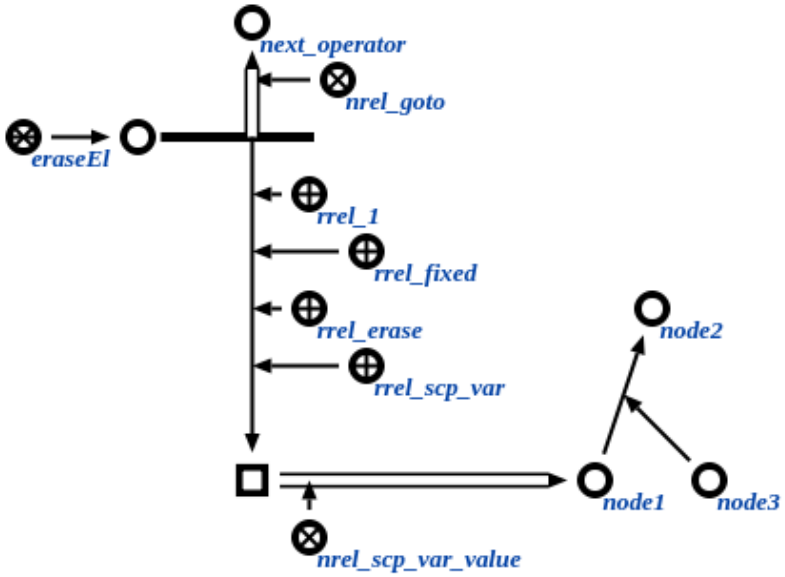




Рис. 3.28 Оператор удаления одноэлементной конструкции (Пример 2)

```
scp_operator_example_eraseEl (*
        <- eraseEl;;
        -> rrel_1: rrel_fixed: rrel_erase: rrel_scp_var: _el1;;
        => nrel_goto: next_operator;;
*);;
```



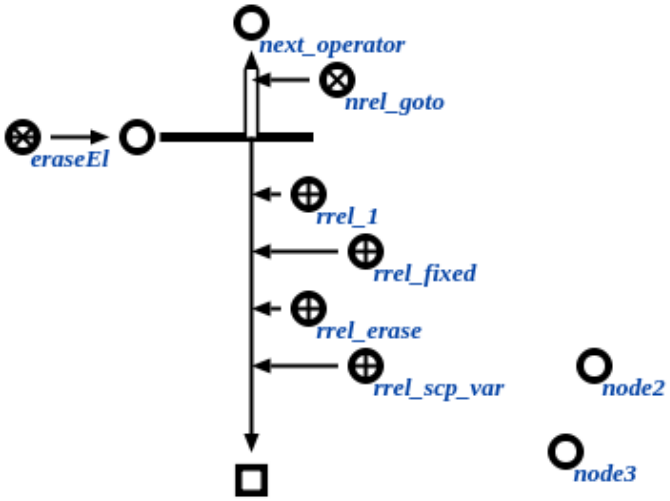




Рис. 3.29 Оператор удаления одноэлементной конструкции (Пример 3)

Операторы класса ***scp-оператор удаления трехэлементной sc-конструкции*** описывают действие удаления одного или нескольких sc-элементов, составляющих *трехэлементную sc-конструкцию*. Каждый оператор данного класса содержит три операнда.

При выполнении операторов данного класса выполняется поиск *sc-элементов*, полностью аналогичный поиску в *scp-операторах поиска трехэлементных sc-конструкций*, после чего выполняется удаление значений

операндов, помеченных как *удаляемый sc-элемент'*. Если условиям поиска удовлетворяет несколько имеющихся в памяти *sc-конструкций*, то будут удалены элементы только одной из них.

```
scp_operator_example_eraseElStr3 (*
        <- eraseElStr3;;
        -> rrel_1: rrel_fixed: rrel_scp_const: some_node;;
        -> rrel_2: rrel_assign: rrel_scp_var: rrel_pos_const_perm: rrel_erase:
_arc;;
        -> rrel_3: rrel_fixed: rrel_scp_var: rrel_erase: _el3;;
        => nrel_goto: next_operator;;
*);;
```

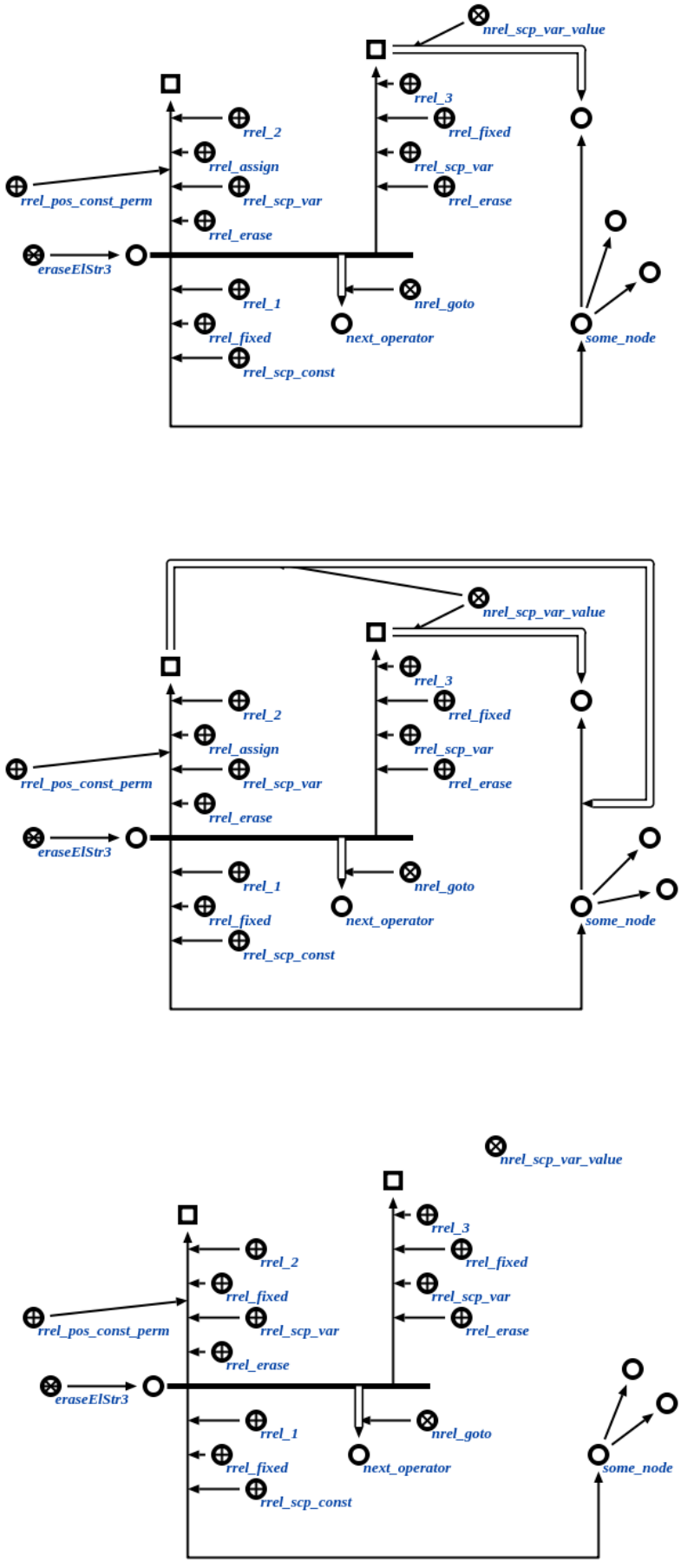



Рис. 3.30 Оператор удаления трехэлементной конструкции (Пример 1)

```
scp_operator_example_eraseElStr3 (*
        <- eraseElStr3;;
        -> rrel_1: rrel_fixed: rrel_scp_const: some_node;;
        -> rrel_2: rrel_assign: rrel_scp_var: rrel_erase: rrel_temp: rrel_fuz:
rrel_var: _arc;;
        -> rrel_3: rrel_assign: rrel_scp_var: rrel_node: _el3;;
        => nrel_goto: next_operator;;
*);;
```

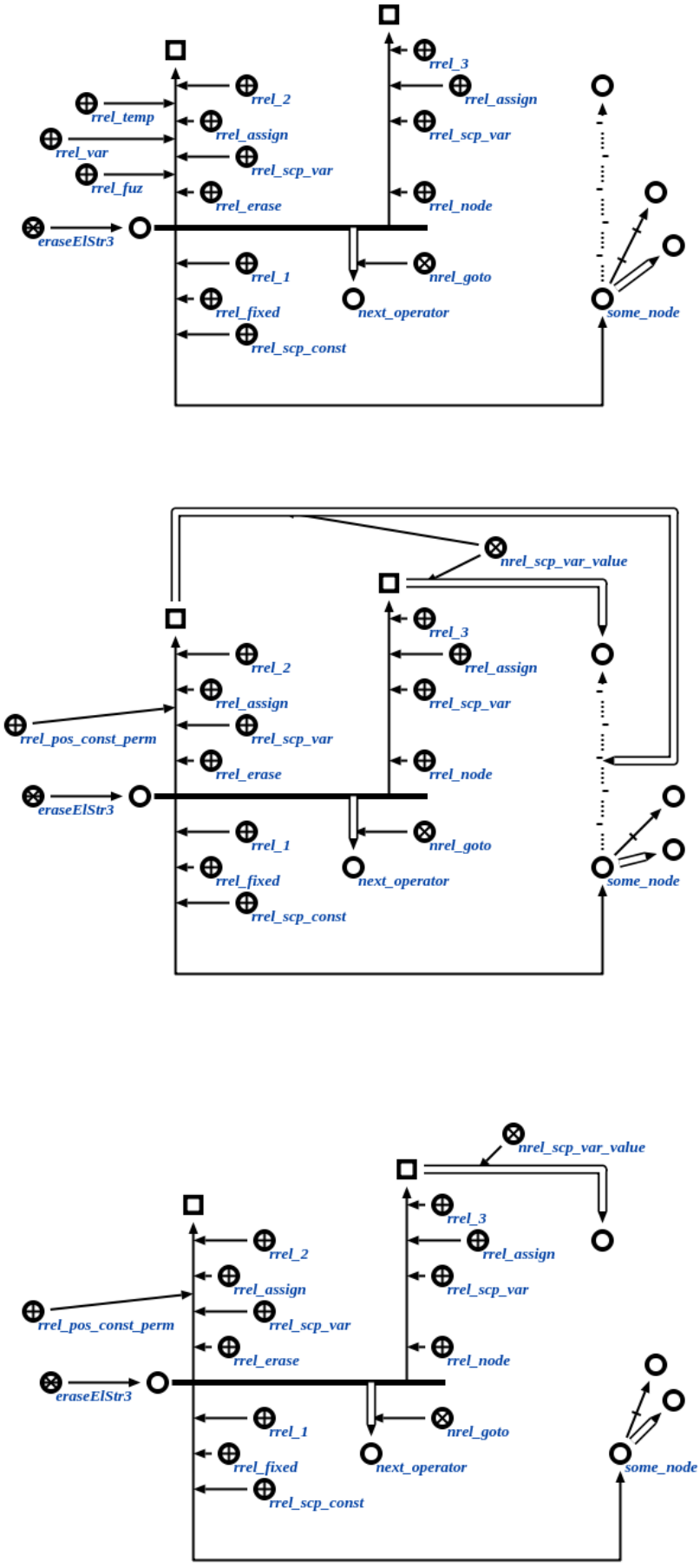



Рис. 3.31 Оператор удаления трехэлементной конструкции (Пример 2)

Операторы класса ***scp-оператор удаления пятиэлементной sc-конструкции*** описывают действие удаления одного или нескольких sc-элементов, составляющих *пятиэлементную sc-конструкцию*. Каждый оператор данного класса содержит пять операнда.

При выполнении операторов данного класса выполняется поиск *sc-элементов*, полностью аналогичный поиску в *scp-операторах поиска пятиэлементных sc-конструкций*, после чего выполняется удаление значений операндов, помеченных как *удаляемый sc-элемент'*. Если условиям поиска удовлетворяет несколько имеющихся в памяти *sc-конструкций*, то будут удалены элементы только одной из них.

```
scp_operator_example_eraseElStr5 (*
        <- eraseElStr5;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_assign: rrel_scp_var: rrel_erase: _arc;;
        -> rrel_3: rrel_assign: rrel_scp_var: rrel_node: _el3;;
        -> rrel_4: rrel_fixed: rrel_scp_var: rrel_pos_const_perm: _arc2;;
        -> rrel_5: rrel_fixed: rrel_scp_var: _el5;;
        => nrel_goto: next_operator;;
*);;
```

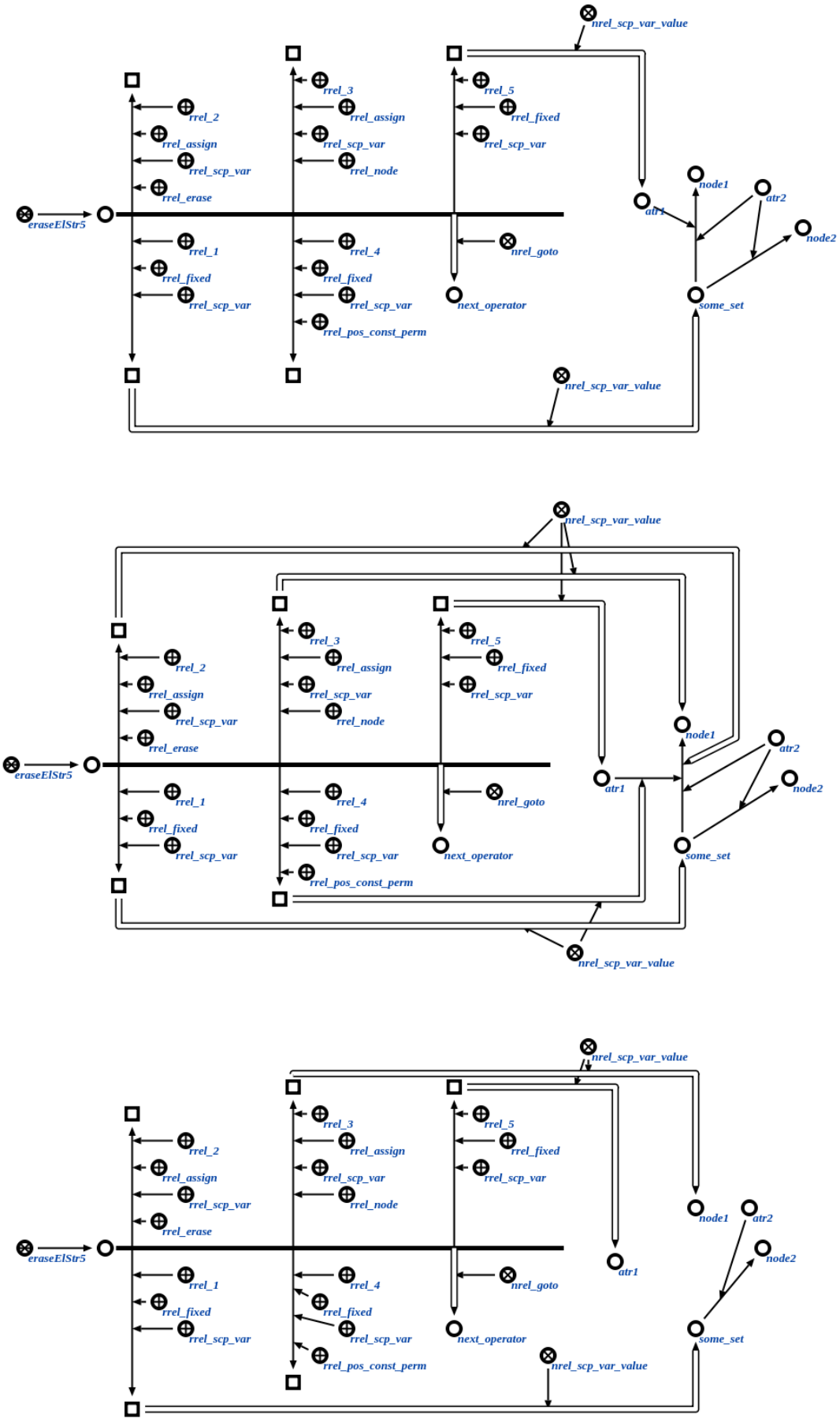



Рис. 3.32 Оператор удаления пятиэлементной конструкции (Пример 1)

Операторы класса ***scp-оператор удаления множества элементов трехэлементной sc-конструкции*** описывают действие, полностью аналогичное действию, выполняемому *scp-операторами удаления трехэлементной sc-конструкции*, однако удалены будут всевозможные *sc-элементы*, удовлетворяющие заданному условию поиска и соответствующие операнду, помеченному как *удаляемый sc-элемент'*.

```
scp_operator_example_eraseSetStr3 (*
        <- eraseSetStr3;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_assign: rrel_scp_var: rrel_erase: _el2;;
        -> rrel_3: rrel_assign: rrel_scp_var: rrel_node: _el3;;
        => nrel_goto: next_operator;;
*);;
```

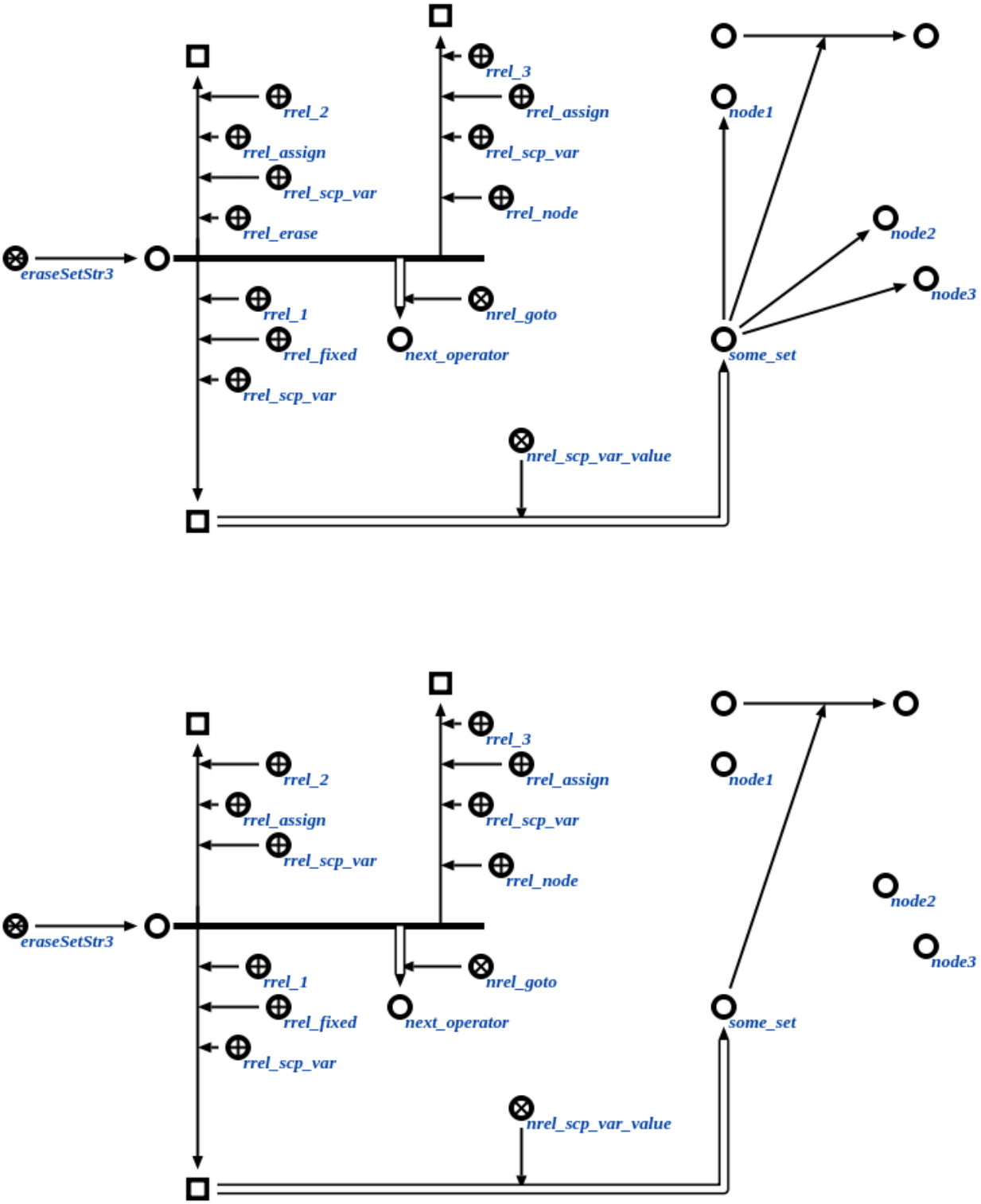



Рис. 3.33 Оператор удаления множества элементов трехэлементной конструкции (Пример 1)

```
scp_operator_example_eraseSetStr3 (*
        <- eraseSetStr3;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_assign: rrel_scp_var: _el2;;
        -> rrel_3: rrel_assign: rrel_scp_var: rrel_node: rrel_erase: _el3;;
        => nrel_goto: next_operator;;
*);;
```

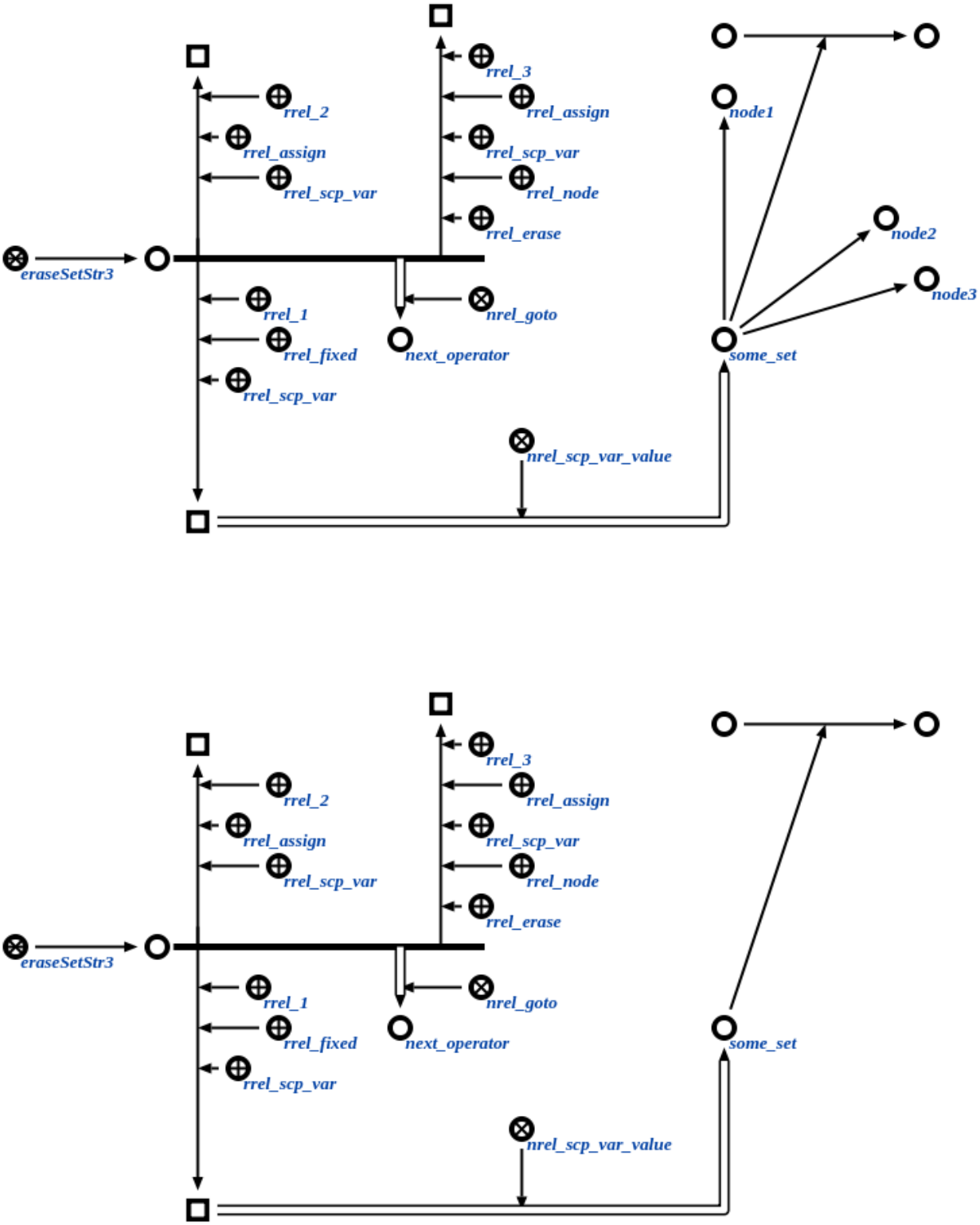



Рис. 3.34 Оператор удаления множества элементов трехэлементной конструкции (Пример 2)

```
scp_operator_example_eraseSetStr3 (*
        <- eraseSetStr3;;
        -> rrel_1: rrel_assign: rrel_scp_var: rrel_node: _el1;;
        -> rrel_2: rrel_assign: rrel_scp_var: _el2;;
        -> rrel_3: rrel_fixed: rrel_scp_var: rrel_erase: _el3;;
        => nrel_goto: next_operator;;
*);;
```

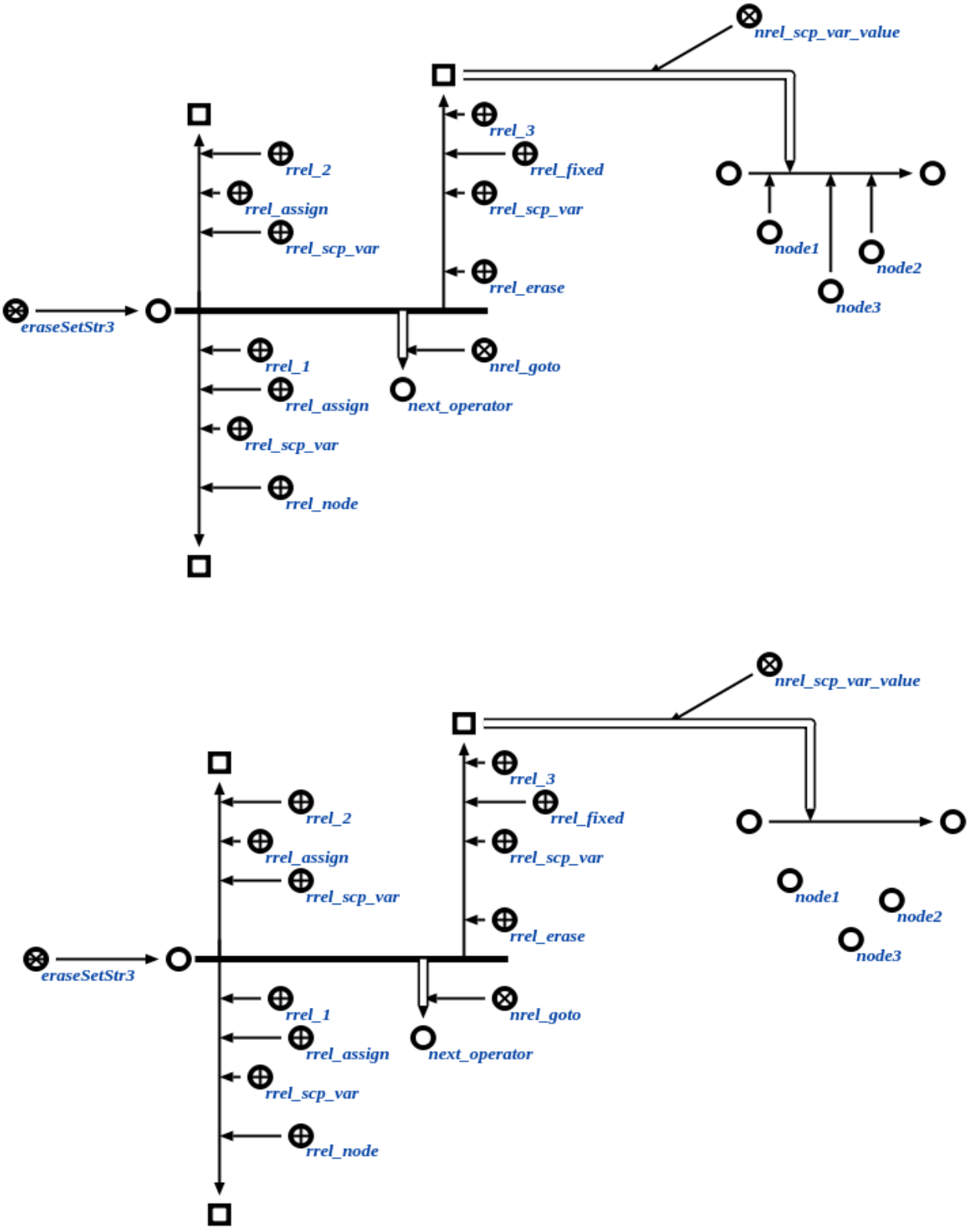



Рис. 3.35 Оператор удаления множества элементов трехэлементной конструкции (Пример 3)

Операторы класса ***scp-оператор удаления множества элементов пятиэлементной sc-конструкции*** описывают действие, полностью аналогичное действию, выполняемому *scp-операторами удаления пятиэлементной sc-конструкции*, однако удалены будут всевозможные *sc-элементы*, удовлетворяющие заданному условию поиска и соответствующие операнду, помеченному как *удаляемый sc-элемент'*.

```
scp_operator_example_eraseSetStr5 (*
        <- eraseSetStr5;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_assign: rrel_scp_var: rrel_erase: _el2;;
        -> rrel_3: rrel_assign: rrel_scp_var: rrel_node: _el3;;
        -> rrel_4: rrel_assign: rrel_scp_var: _el4;;
        -> rrel_5: rrel_fixed: rrel_scp_var: _el5;;
        => nrel_goto: next_operator;;
*);;
```

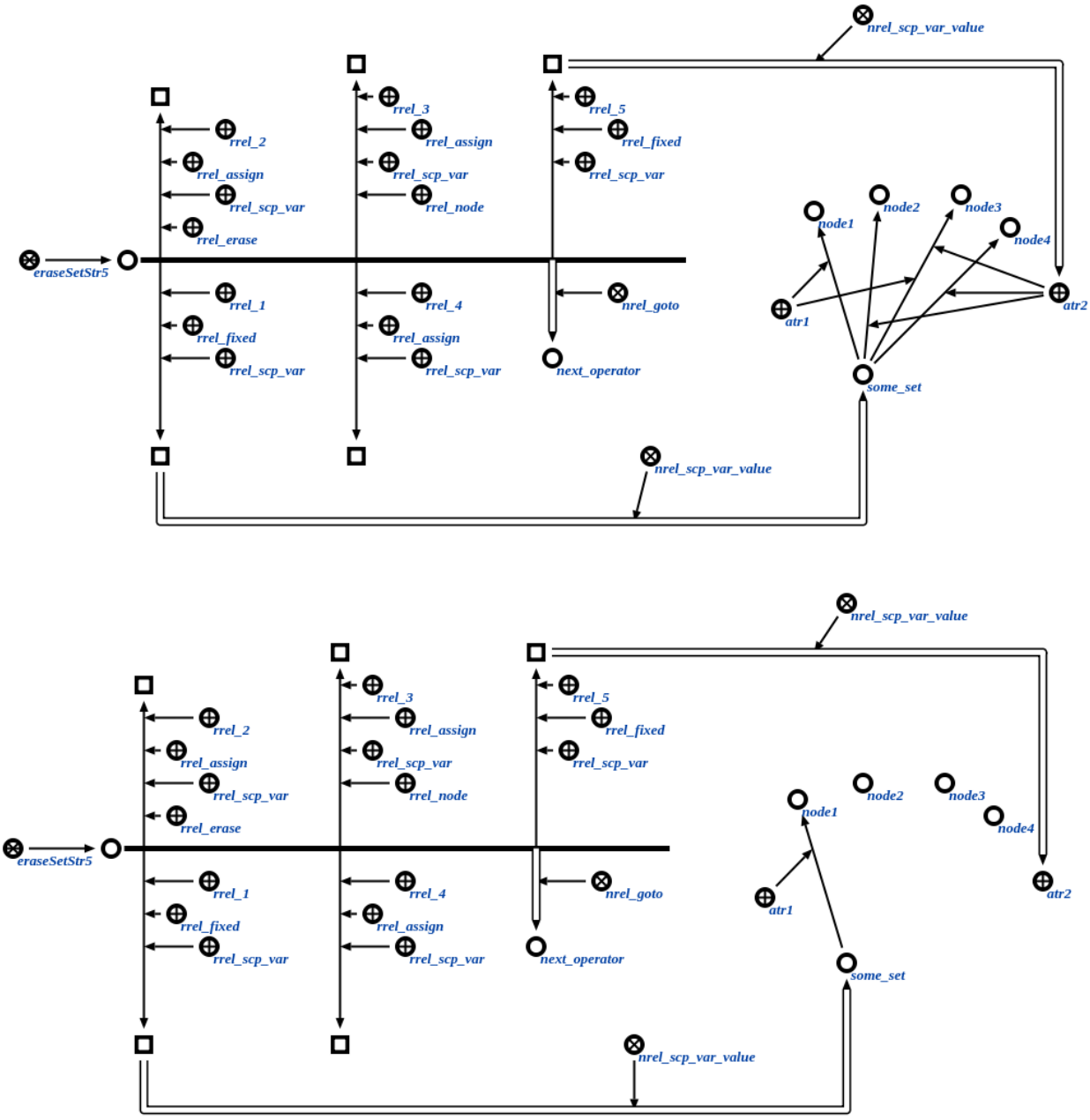






Рис. 3.36 Оператор удаления множества элементов пятиэлементной конструкции (Пример 1)

```
scp_operator_example_eraseSetStr5 (*
        <- eraseSetStr5;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_assign: rrel_scp_var: _el2;;
        -> rrel_3: rrel_assign: rrel_scp_var: rrel_node: _el3;;
        -> rrel_4: rrel_assign: rrel_scp_var: rrel_erase: _el4;;
        -> rrel_5: rrel_fixed: rrel_scp_var: _el5;;
        => nrel_goto: next_operator;;
*);;
```

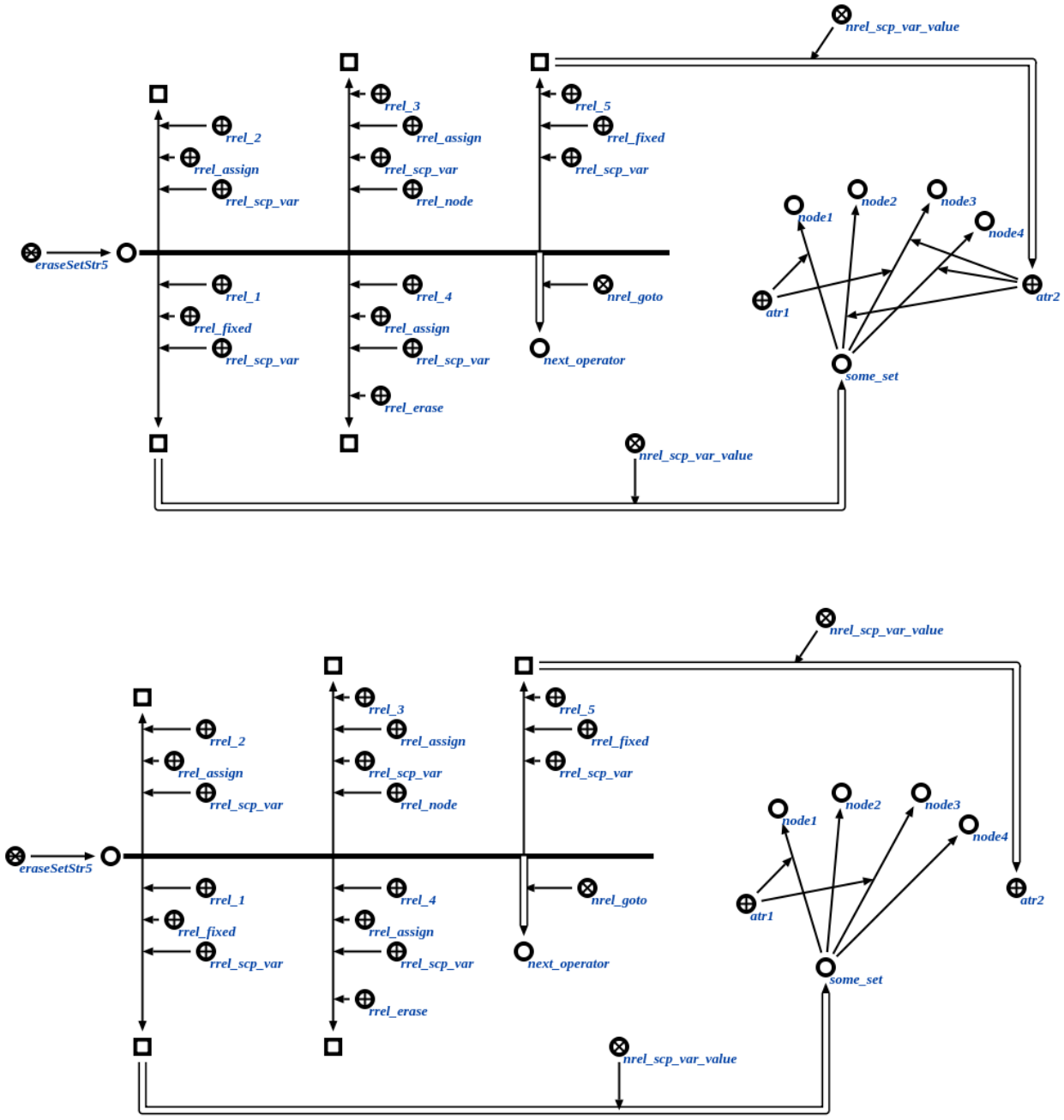



Рис. 3.37 Оператор удаления множества элементов пятиэлементной конструкции (Пример 2)

### 3.4 Scp-операторы проверки условий

Операторы класса ***scp-оператор проверки условий*** описывают действие проверки некоторого условия с целью ветвления *scp-программы*. При этом

состояние памяти системы не изменяется, а после выполнения оператора осуществляется переход к следующему оператору по связке отношения *следующий оператор при успешном выполнении** или *следующий оператор при безуспешном выполнении**, в зависимости от истинности некоторого условия.

Данный класс *scp-операторов* разбивается на следующие подклассы:

- ***scp-оператор проверки типа sc-элемента;***
- ***scp-оператор проверки наличия значения у переменной;***
- ***scp-оператор проверки наличия содержимого у sc-ссылки;***
- ***scp-оператор проверки совпадения значений операндов;***
- ***scp-оператор проверки равенства числовых содержимых sc-ссылок;***
- ***scp-оператор сравнения числовых содержимых sc-ссылок.***

Операторы класса ***scp-оператор проверки типа sc-элемента*** описывают действие проверки того, соответствует ли *sc-элемент*, являющийся значением единственного операнда, указанному типу. Успешным выполнение считается, если структурный тип значения операнда соответствует заданному.

Каждый оператор данного класса содержит один операнд, который должен быть помечен как *scp-операнд с заданным значением'*. Тип *sc-элемента*, на соответствие которому будет осуществляться проверка, указывается при помощи подмножеств ролевого отношения *тип sc-элемента'*.

```
scp_operator_example_ifType (*
        <- ifType;;
        -> rrel_1: rrel_scp_const: some_node;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```

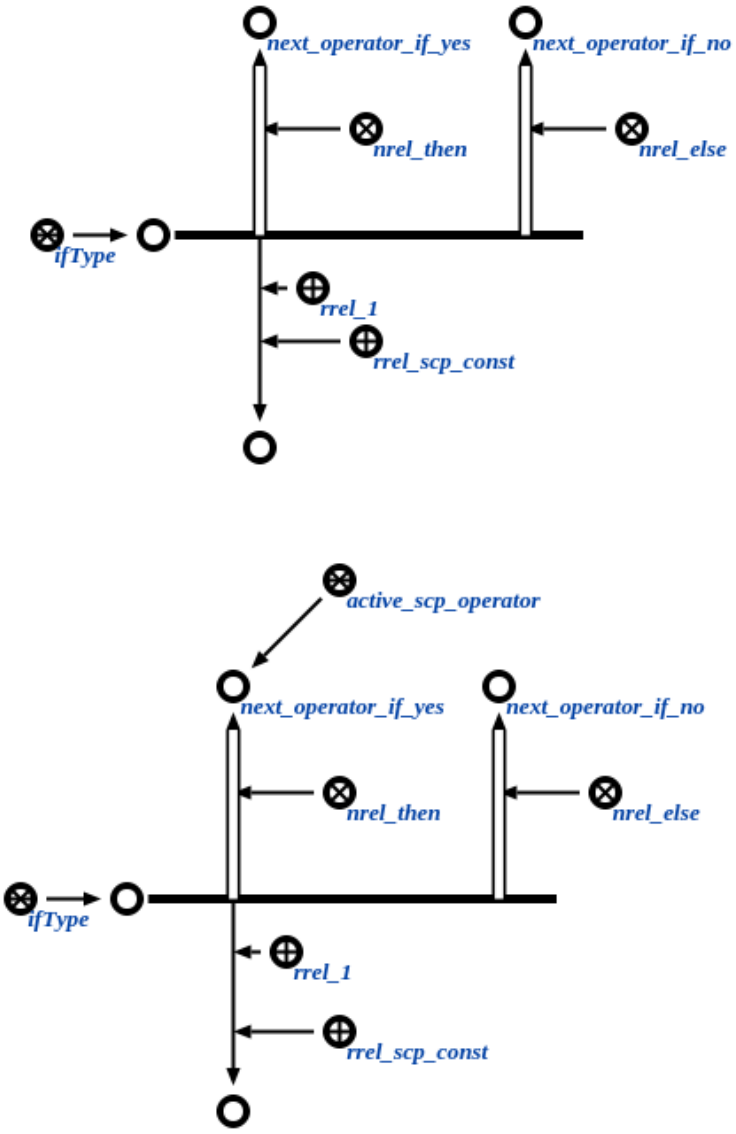
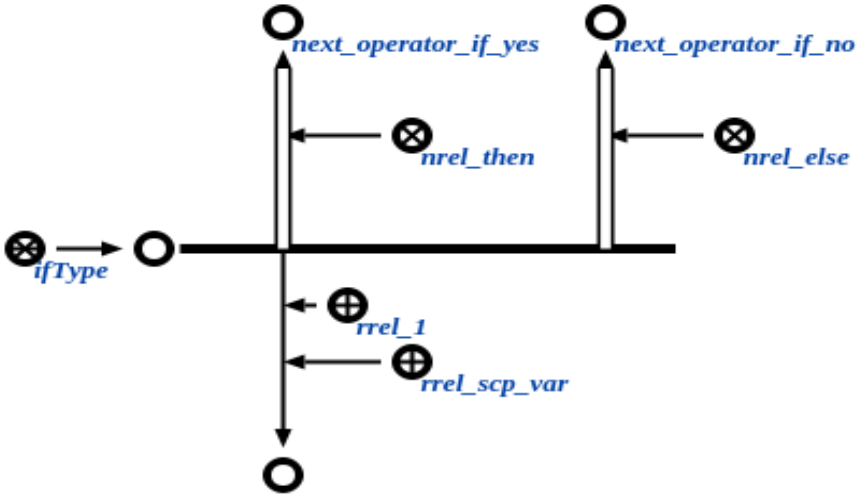



Рис. 3.38 Оператор проверки типа sc-элемента (Пример 1)

```
scp_operator_example_ifType (*
        <- ifType;;
        -> rrel_1: rrel_scp_var: some_node;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```



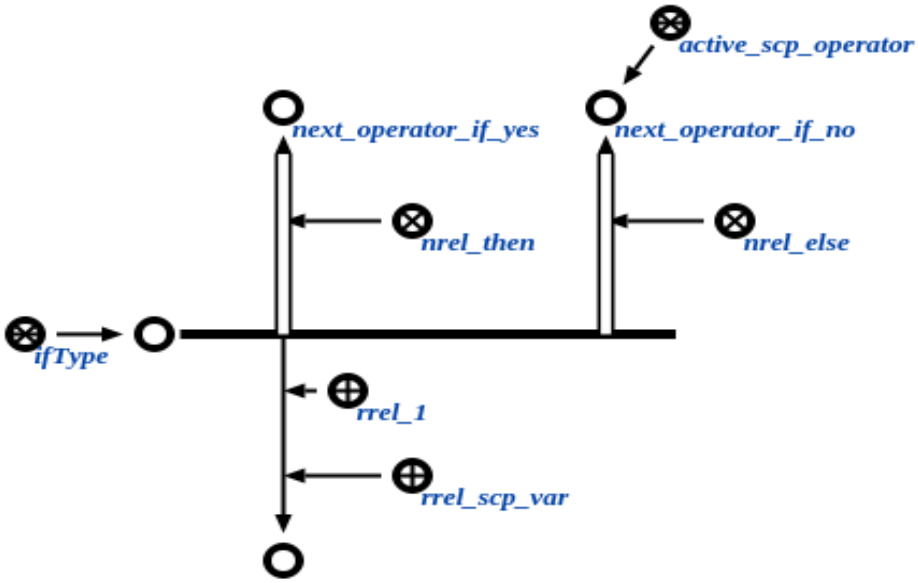




Рис. 3.39 Оператор проверки типа sc-элемента (Пример 2)

```
scp_operator_example_ifType (*
        <- ifType;;
        -> rrel_1: rrel_scp_var: rrel_arc: some_node;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```

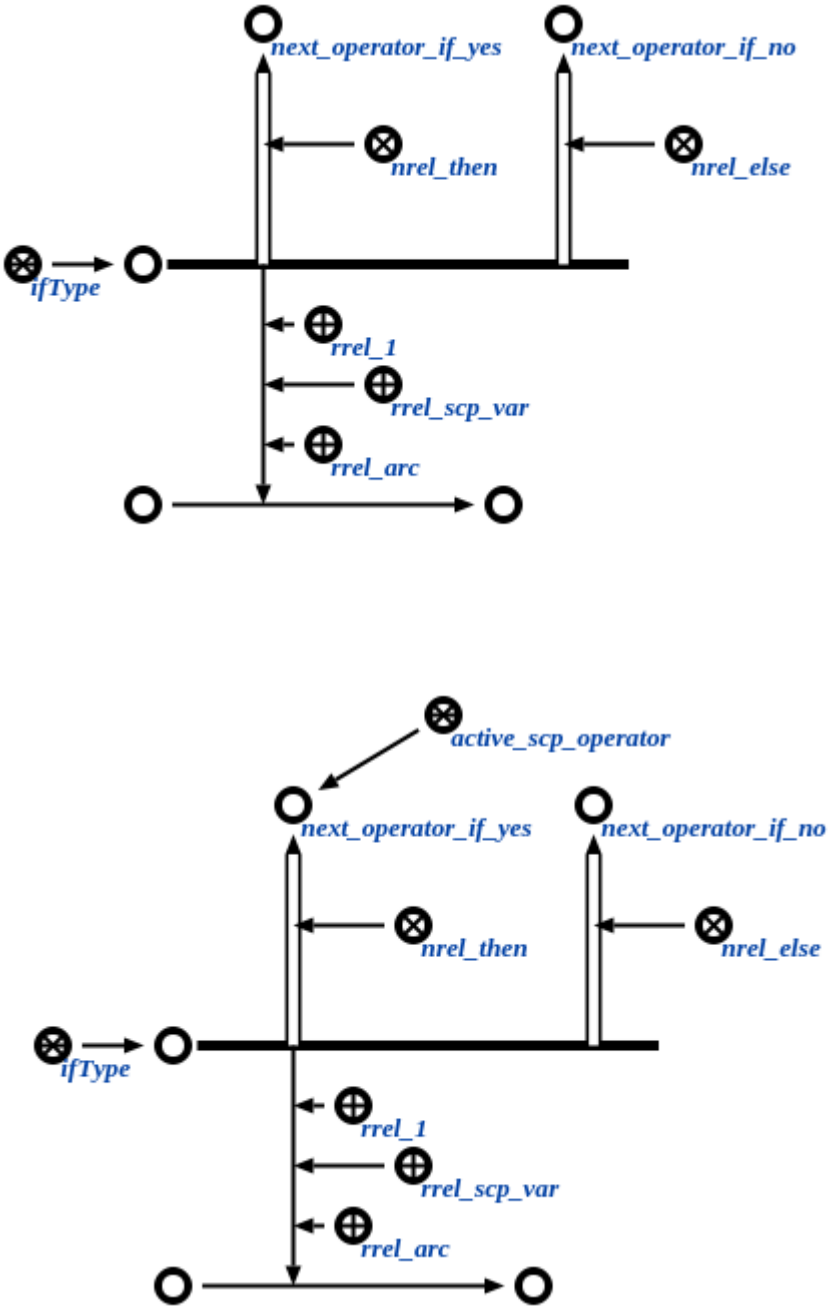


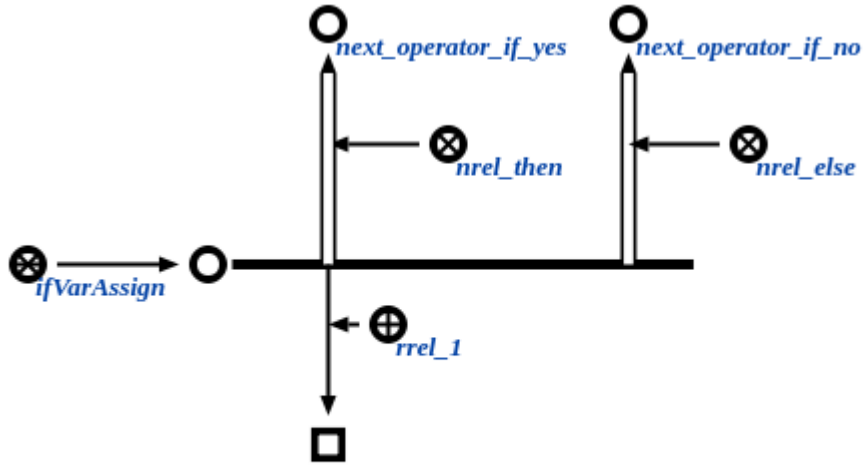

Рис. 3.40 Оператор проверки типа sc-элемента (Пример 3)

Операторы класса ***scp-оператор проверки наличия значения у переменной*** описывают действие проверки того, имеет ли единственный операнд значение. Успешным выполнение считается, если значение найдено.

Каждый оператор данного класса содержит один операнд, который должен быть помечен как *scp-операнд с заданным значением'*. Если указанный операнд

является *scp-переменной'*, то осуществляется попытка поиска связки отношения *значение** для данного операнда. Использование данного класса *scp-операторов* с *scp-константами'* не целесообразно, так как *scp-константа'* всегда имеет значение.

```
scp_operator_example_ifVarAssign (*
        <- ifVarAssign;;
        -> rrel_1: _node;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```



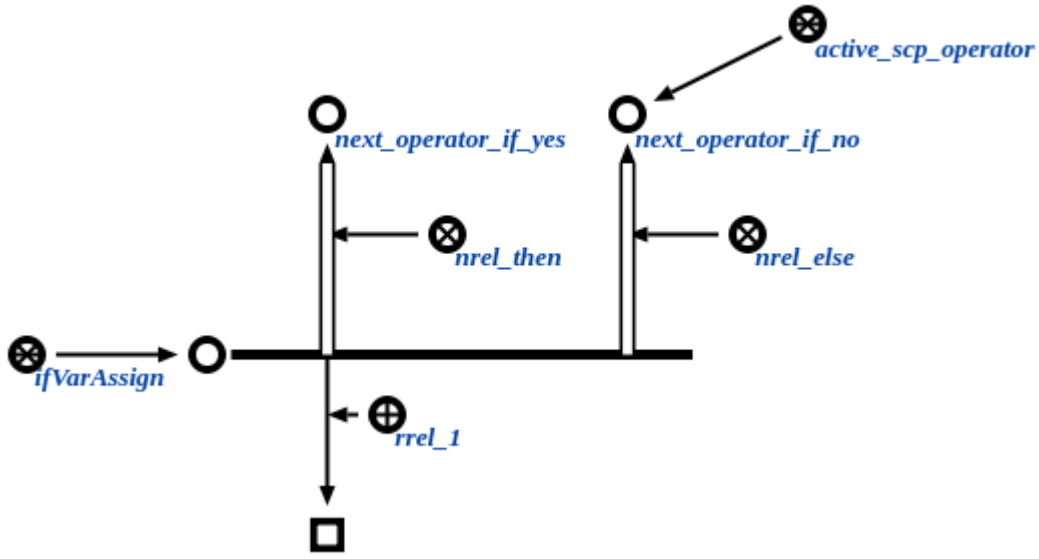




Рис. 3.41 Оператор проверки наличия значения у переменной (Пример 1)

```
scp_operator_example_ifVarAssign (*
        <- ifVarAssign;;
        -> rrel_1: _node;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```

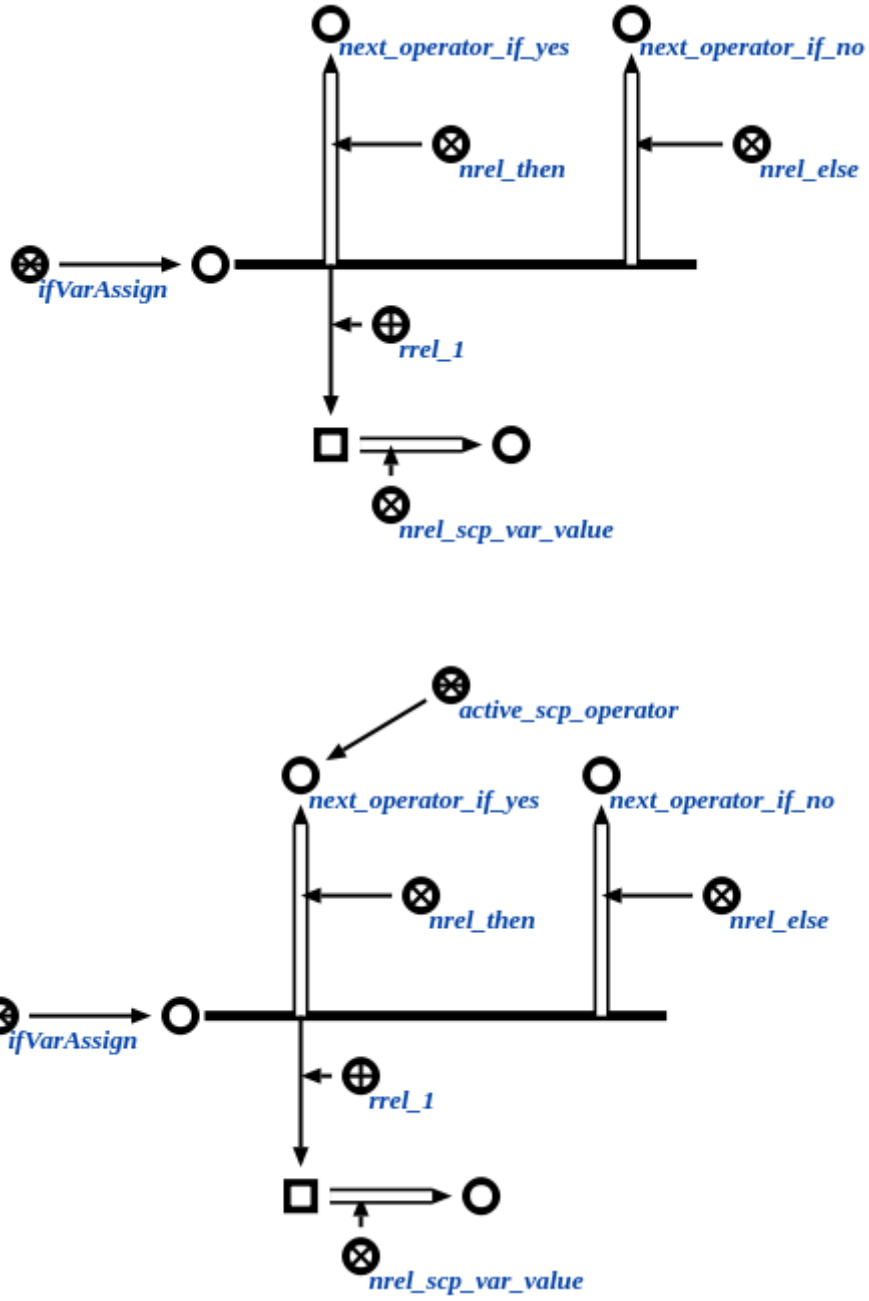



Рис. 3.42 Оператор проверки наличия значения у переменной (Пример 2)

Операторы класса ***scp-оператор проверки наличия содержимого у sc-ссылки*** описывают действие проверки того, имеет ли заданная sc-ссылка сформированное содержимое, т.е. существует ли соответствующий данной ссылке файл. Успешным выполнение считается, если содержимое сформировано.

Каждый оператор данного класса содержит один операнд, который должен быть помечен как *scp-операнд с заданным значением'*. *Sc-элемент*, являющийся значением операнда должен быть *sc-ссылкой*.

```
scp_operator_example_ifFormCount (*
        <- ifFormCount;;
        -> rrel_1: rrel_scp_var: _node;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```

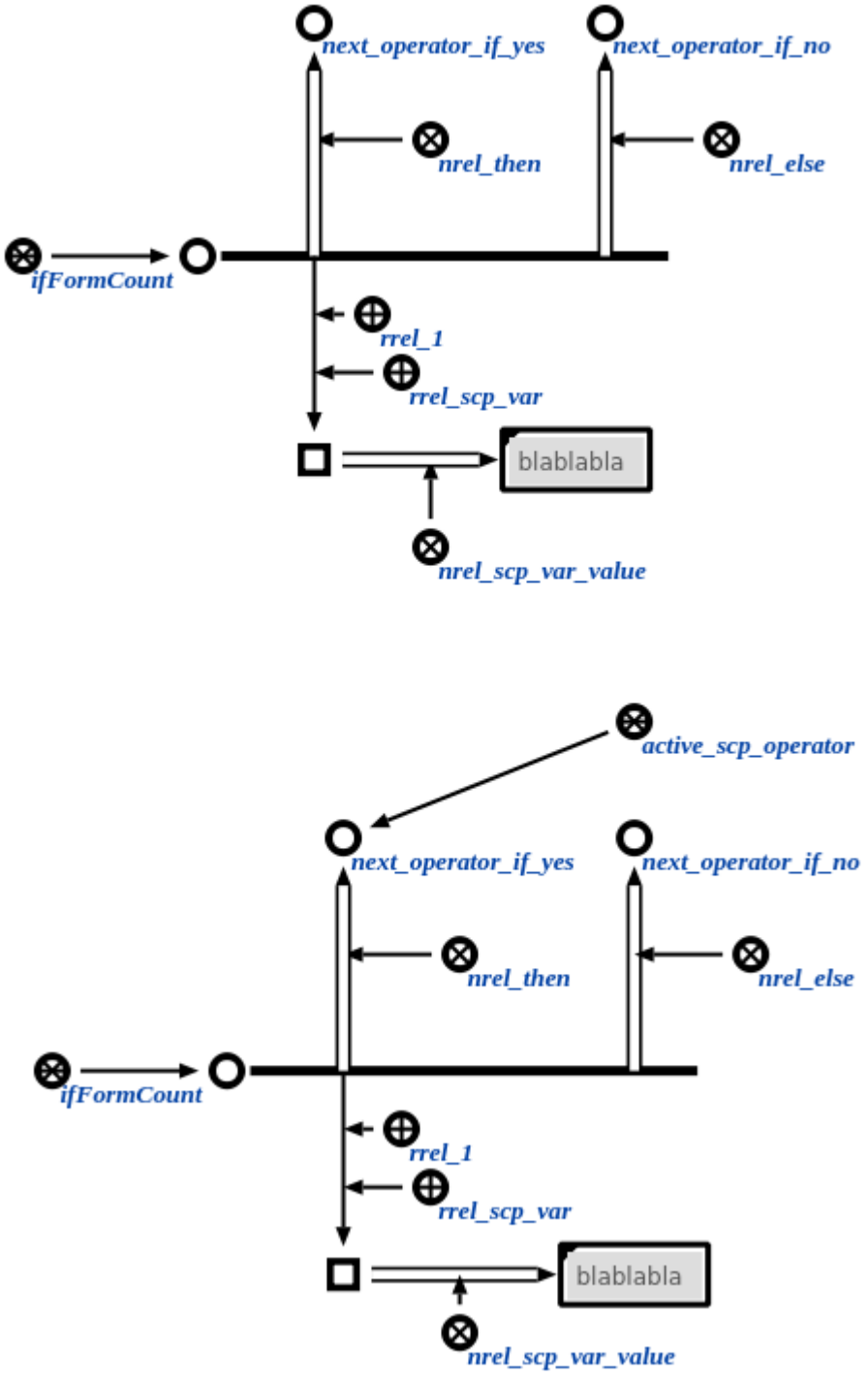
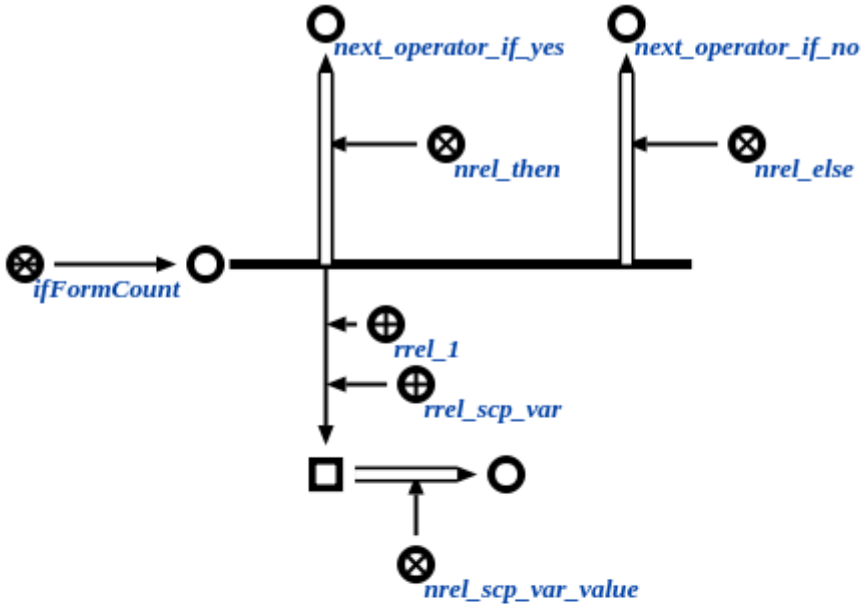



Рис. 3.43 Оператор проверки наличия содержимого у sc-ссылки (Пример 1)

```
scp_operator_example_ifFormCount (*
        <- ifFormCount;;
        -> rrel_1: rrel_scp_var: _node;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```



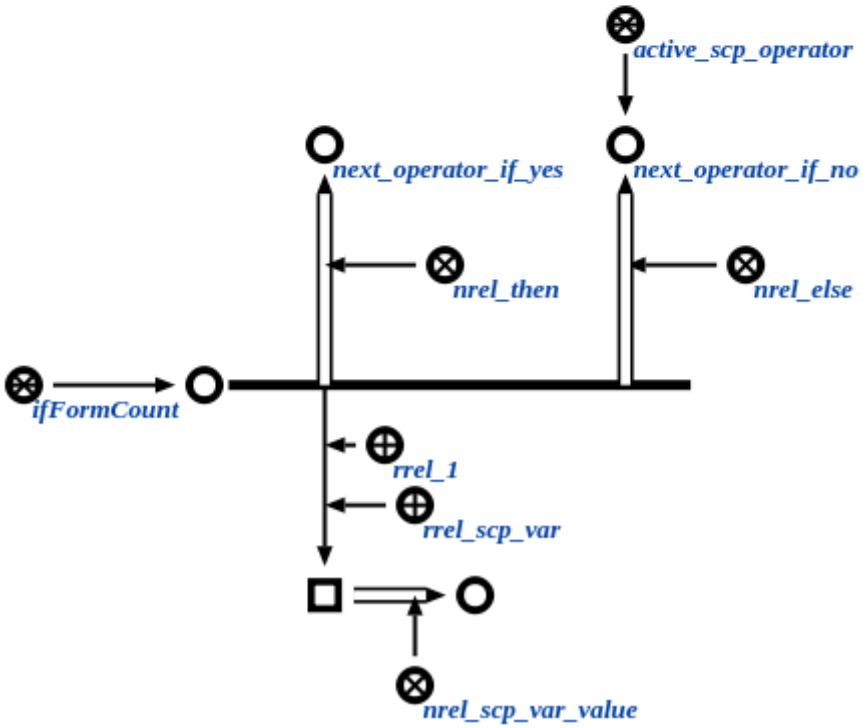
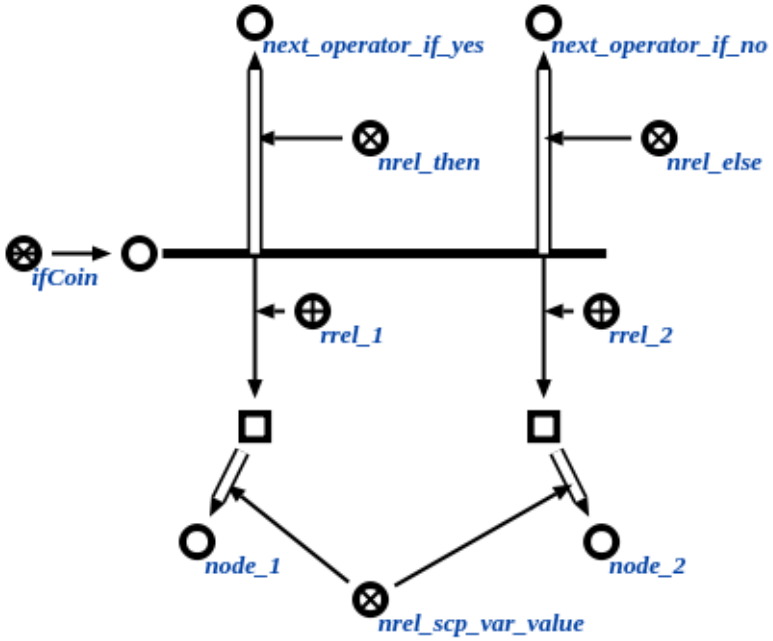




Рис. 3.44 Оператор проверки наличия содержимого у sc-ссылки (Пример 2)

Операторы класса ***scp-оператор проверки совпадения значений операндов*** описывают действие проверки того, совпадают ли значения операндов. Успешным выполнение считается, если значения совпадают. При этом учитывается только явное совпадение двух *sc-элементов*, дополнительные проверки на семантическую или логическую эквивалентность не осуществляются.

Каждый оператор данного класса содержит два операнда, которые должны быть помечены как *scp-операнд с заданным значением'*. Каждый из операндов может быть как *scp-константой'*, так и *scp-переменной'*, однако сравнение значений двух *scp-констант'* не является целесообразным.

```
scp_operator_example_ifCoin (*
        <- ifCoin;;
        -> rrel_1: _node1;;
        -> rrel_2: _node2;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```



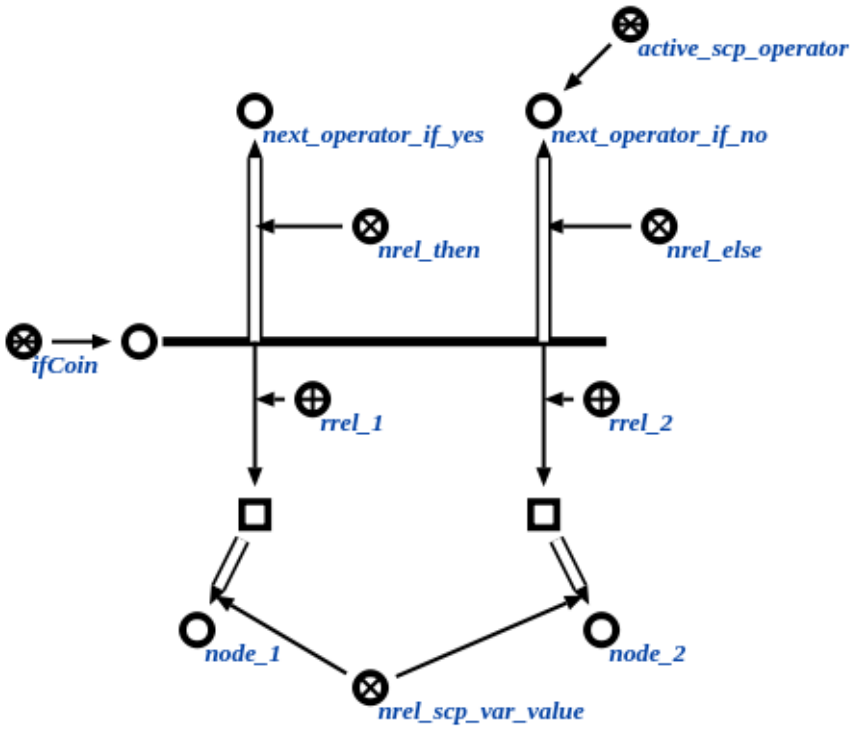
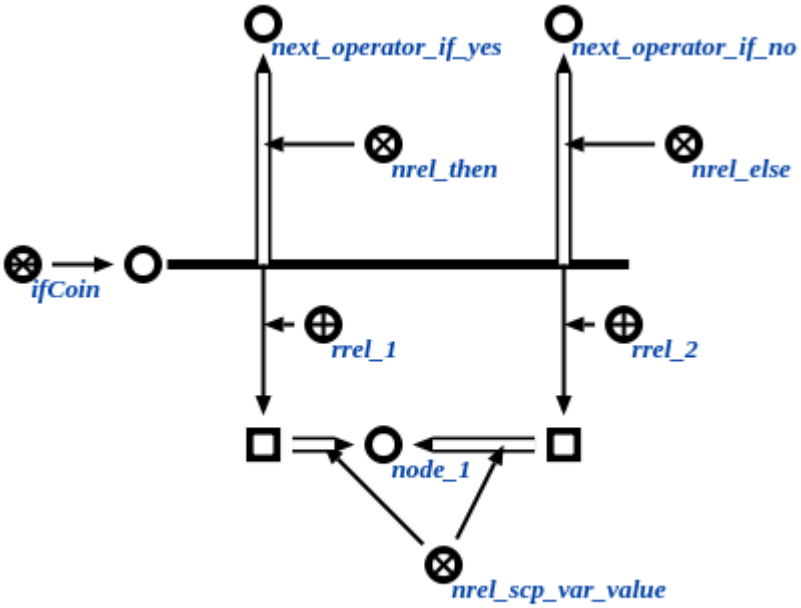




Рис. 3.45 Оператор проверки совпадения значений операндов (Пример 1)

```
scp_operator_example_ifCoin (*
        <- ifCoin;;
        -> rrel_1: _node1;;
        -> rrel_2: _node2;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```



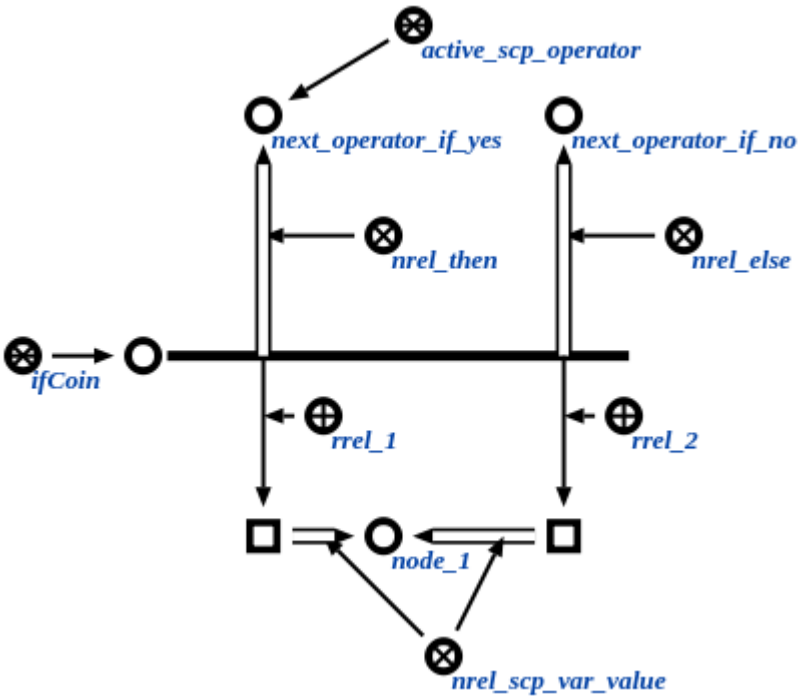
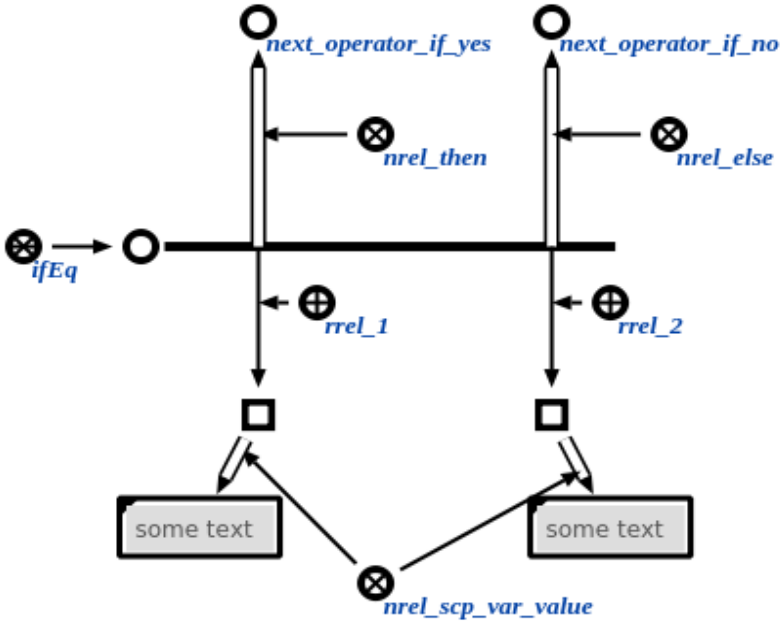




Рис. 3.46 Оператор проверки совпадения значений операндов (Пример 2)

Операторы класса ***scp-оператор проверки равенства числовых содержимых sc-ссылок*** описывают действие проверки того, равны ли между собой числовые содержимые двух *sc-ссылок*, являющихся значениями операндов. Под числовым содержимым понимаются те же виды файлов, что и во случае с *scp-операторами обработки содержимых sc-ссылок.* Успешным выполнение считается, если численные содержимые равны с точки зрения теории чисел. Например, числа «1e3» и «1000» считаются равными.

Каждый оператор данного класса содержит два операнда, которые должны быть помечены как *scp-операнд с заданным значением'*. Каждый из операндов может быть как *scp-константой'*, так и *scp-переменной'*.

```
scp_operator_example_ifEq (*
        <- ifEq;;
        -> rrel_1: _node1;;
        -> rrel_2: _node2;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```



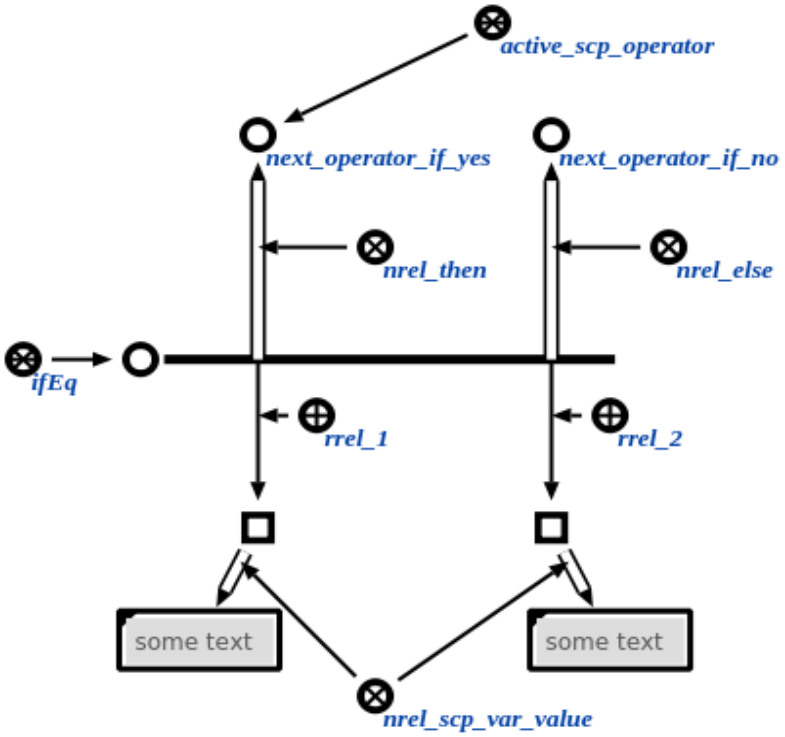
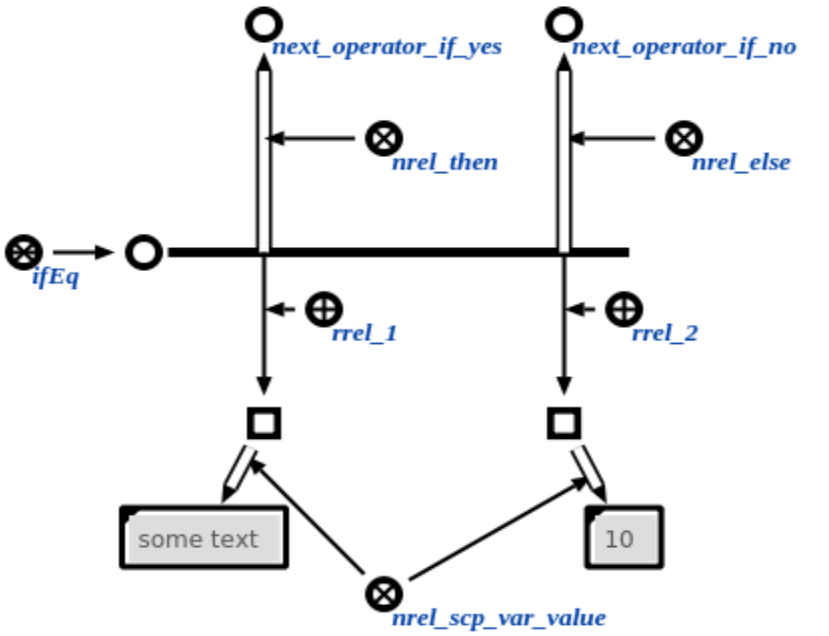




Рис. 3.47 Оператор проверки равенства числовых содержимых операндов (Пример 1)

```
scp_operator_example_ifEq (*
        <- ifEq;;
        -> rrel_1: _node1;;
        -> rrel_2: _node2;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```



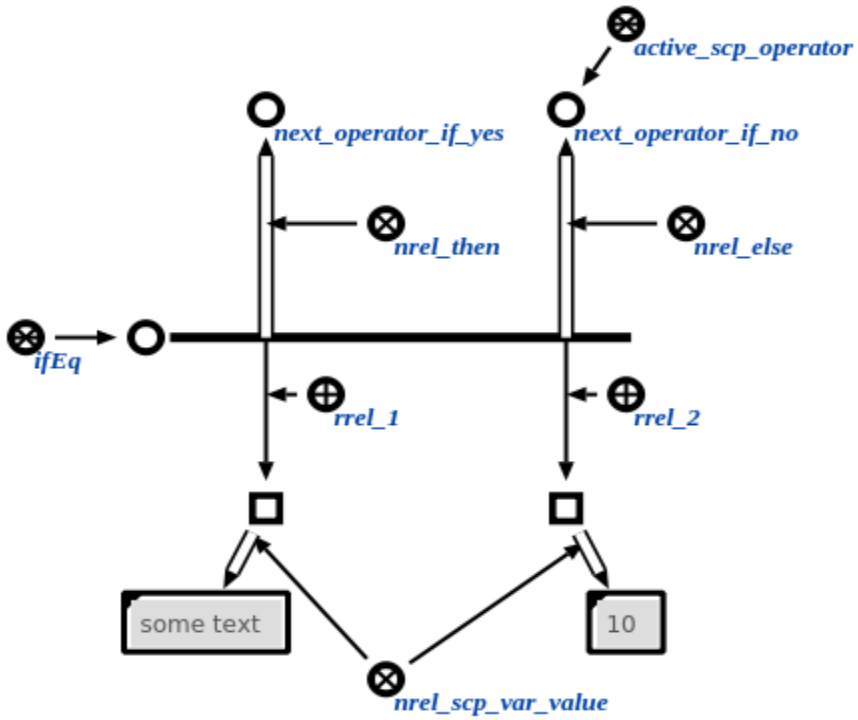




Рис. 3.48 Оператор проверки равенства числовых содержимых операндов (Пример 2)

```
scp_operator_example_ifEq (*
        <- ifEq;;
        -> rrel_1: _node1;;
        -> rrel_2: _node2;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```

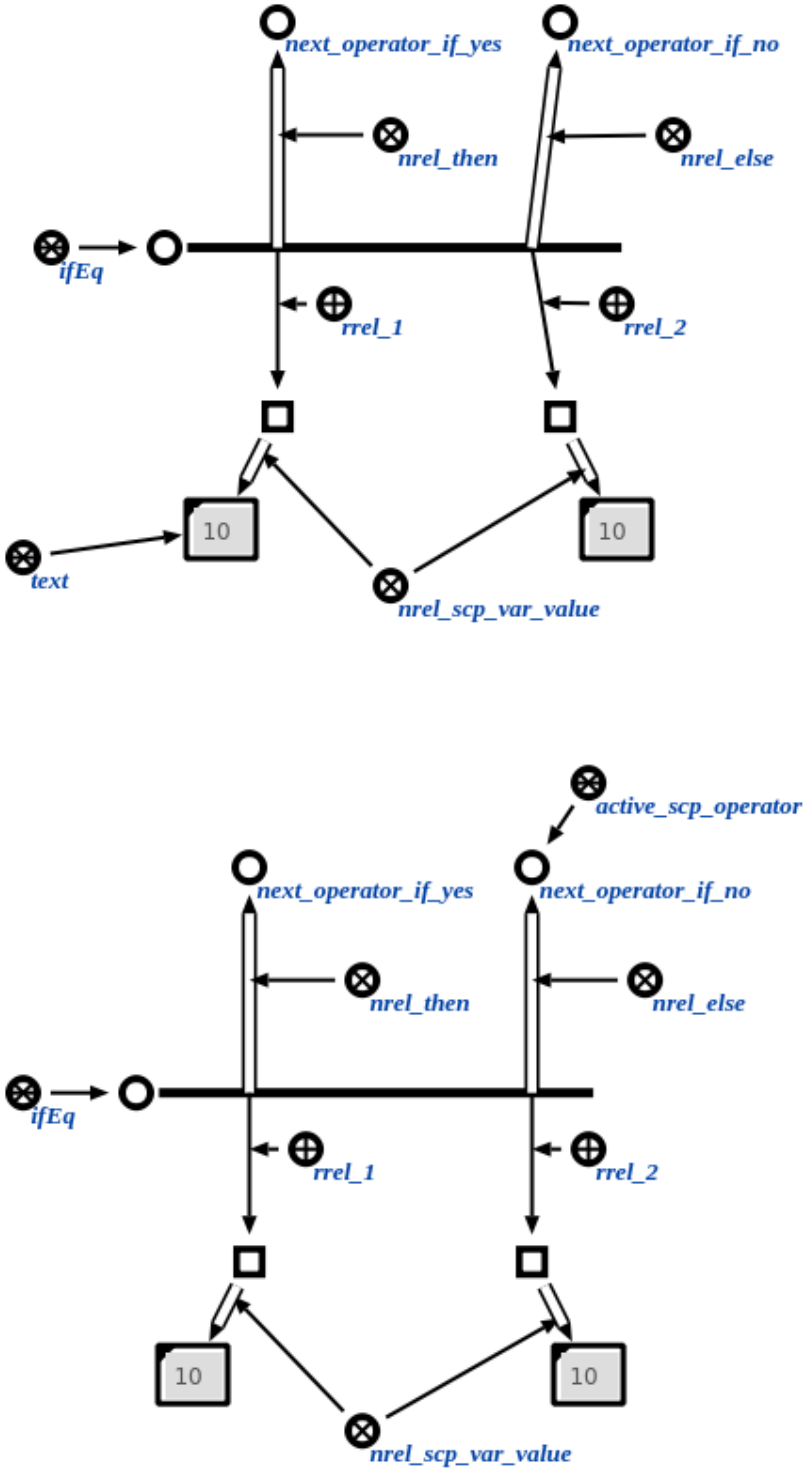
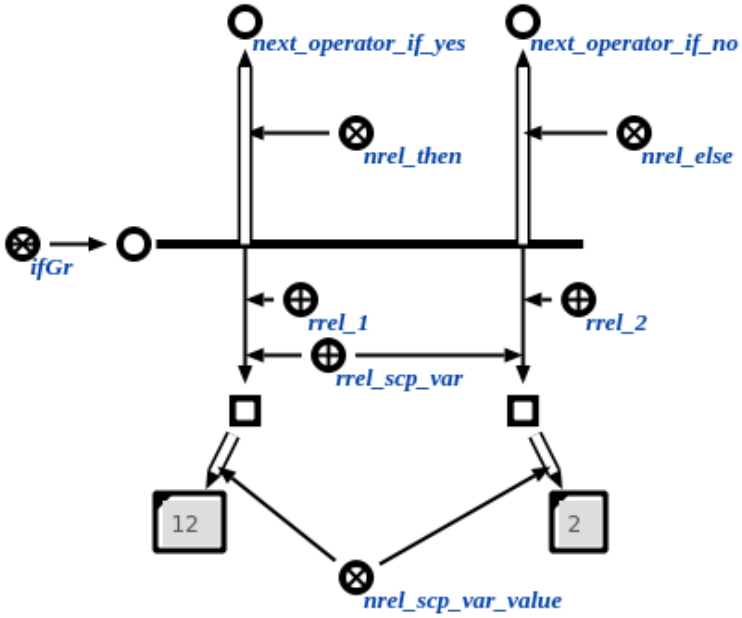



Рис. 3.49 Оператор проверки равенства числовых содержимых операндов (Пример 3)

Операторы класса ***scp-оператор сравнения числовых содержимых sc-ссылок*** описывают действие сравнения между собой числовых содержимых двух *sc-ссылок*, являющихся значениями операндов. Под числовым содержимым понимаются те же виды файлов, что и во случае с *scp-операторами обработки содержимых sc-ссылок.* Успешным выполнение считается, значение первого операнда строго больше значения второго операнда с точки зрения теории чисел. Например, число «1e3» больше, чем число «987».

Каждый оператор данного класса содержит два операнда, которые должны быть помечены как *scp-операнд с заданным значением'*. Каждый из операндов может быть как *scp-константой'*, так и *scp-переменной'*.

```
scp_operator_example_ifGr (*
        <- ifGr;;
        -> rrel_1: rrel_scp_var: _node1;;
        -> rrel_2: rrel_scp_var: _node2;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```



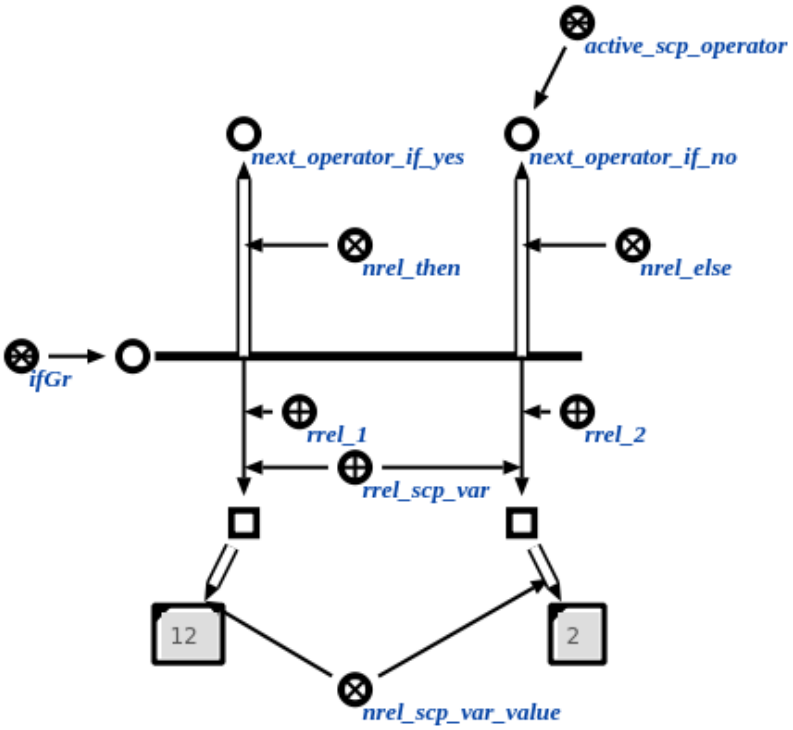
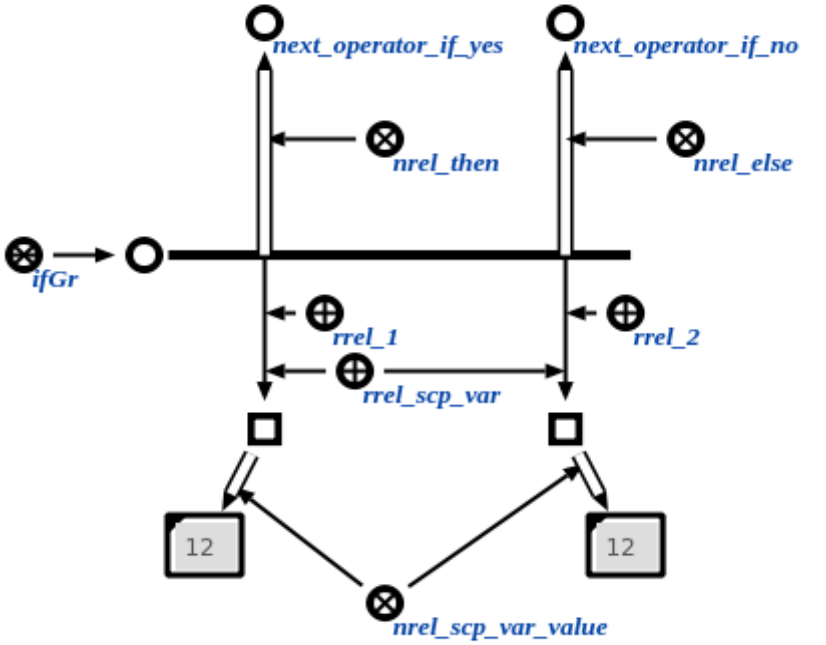




Рис. 3.50 Оператор сравнения числовых содержимых операндов (Пример 1)

```
scp_operator_example_ifGr (*
        <- ifGr;;
        -> rrel_1: rrel_scp_var: _node1;;
        -> rrel_2: rrel_scp_var: _node2;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```



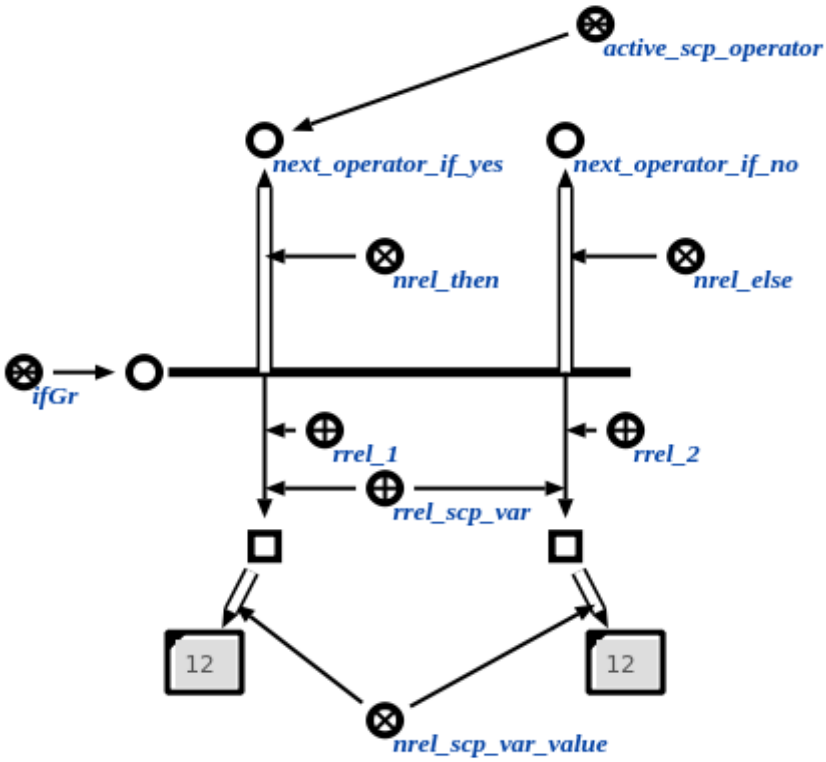




Рис. 3.51 Оператор сравнения числовых содержимых операндов (Пример 2)

### 3.5 Scp-операторы обработки содержимых sc-ссылок

Операторы класса ***scp-оператор обработки содержимых sc-ссылок*** описывают различные действия, связанные с обработкой содержимых *sc-ссылок*. Большинство операторов данного класса, за исключением *scp-операторов копирования содержимого sc-ссылки* и *scp-операторов удаления содержимого sc-ссылки* работают с числовыми содержимыми. Под числовым содержимым подразумевается файл, являющийся идентификатором некоторого действительного числа в десятичной системе счисления. При этом допускаются общепринятые форматы подобной идентификации, например, корректными считаются следующие идентификаторы: «4», «4.555», «6.022e23», «6.63e-34», «4e-7». Идентификаторы в других системах счисления либо идентификаторы, использующие какие-либо специальные обозначения, не допускаются.

Операторы класса не предполагают ветвления программы в зависимости от выполнения дополнительных условий, поэтому указание следующих операторов осуществляется только при помощи отношения *следующий оператор**, без подмножеств.

Данный класс *scp-операторов* разбивается на следующие подклассы:

- ***scp-оператор сложения числовых содержимых sc-ссылок;***
- ***scp-оператор вычитания числовых содержимых sc-ссылок;***
- ***scp-оператор умножения числовых содержимых sc-ссылок;***
- ***scp-оператор деления числовых содержимых sc-ссылок;***
- ***scp-оператор возведения числового содержимого sc-ссылки в степень;***
- ***scp-оператор вычисления логарифма числового содержимого sc-ссылки;***
- ***scp-оператор вычисления синуса числового содержимого sc-ссылки;***
- ***scp-оператор вычисления косинуса числового содержимого sc-ссылки;***
- ***scp-оператор вычисления тангенса числового содержимого sc-ссылки;***
- ***scp-оператор вычисления арксинуса числового содержимого sc-ссылки;***
- ***scp-оператор вычисления арккосинуса числового содержимого sc-ссылки;***
- ***scp-оператор вычисления арктангенса числового содержимого sc-ссылки;***

- ***scp-оператор копирования содержимого sc-ссылки;***
- ***scp-оператор удаления содержимого sc-ссылки.***

Операторы класса ***scp-оператор сложения числовых содержимых sc-ссылок*** описывают действие сложения между собой числовых содержимых двух *sc-ссылок*, являющихся значениями второго и третьего операндов.

Каждый оператор данного класса содержит три операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся суммой чисел, задаваемых вторым и третьим операндами. В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то *sc-ссылка* будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся *sc-ссылки*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся одним из слагаемых при сложении.

Третий операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся одним из слагаемых при сложении.

```
scp_operator_example_contAdd (*
        <- contAdd;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [30];;
        -> rrel_3: rrel_fixed: rrel_scp_const: [50];;
        => nrel_goto: next_operator;;
*);;
```

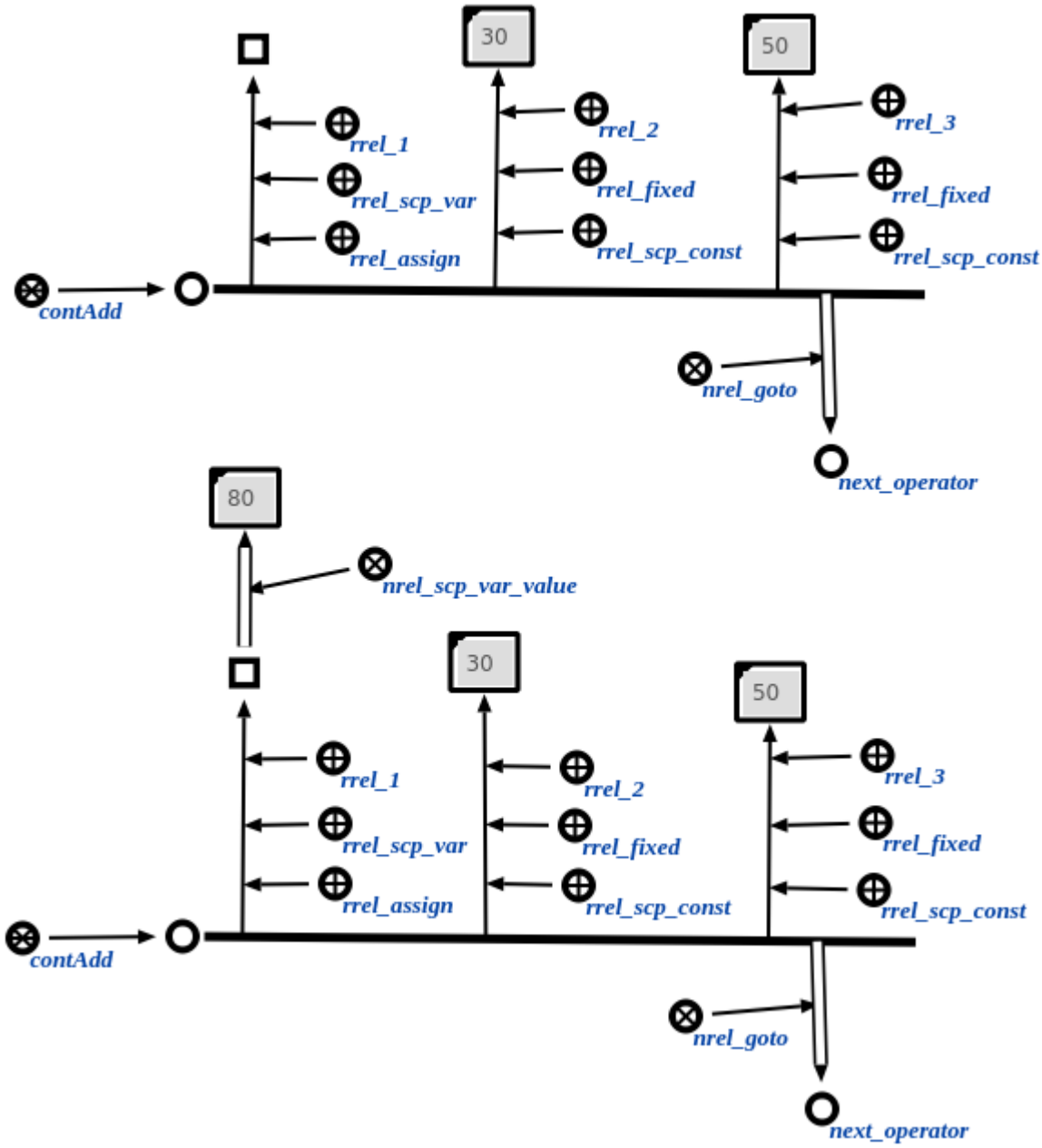



Рис. 3.52 Оператор сложения числовых содержимых операндов

Операторы класса ***scp-оператор вычитания числовых содержимых sc-ссылок*** описывают действие вычитания числовых содержимых двух *sc-ссылок*, являющихся значениями второго и третьего операндов.

Каждый оператор данного класса содержит три операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся разностью чисел, задаваемых вторым и третьим операндами. В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то *sc-ссылка* будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся *sc-ссылки*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся уменьшаемым при вычитании.

Третий операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся вычитаемым при сложении.

```
scp_operator_example_contSub (*
        <- contSub;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [500];;
        -> rrel_3: rrel_fixed: rrel_scp_const: [345];;
        => nrel_goto: next_operator;;
*);;
```

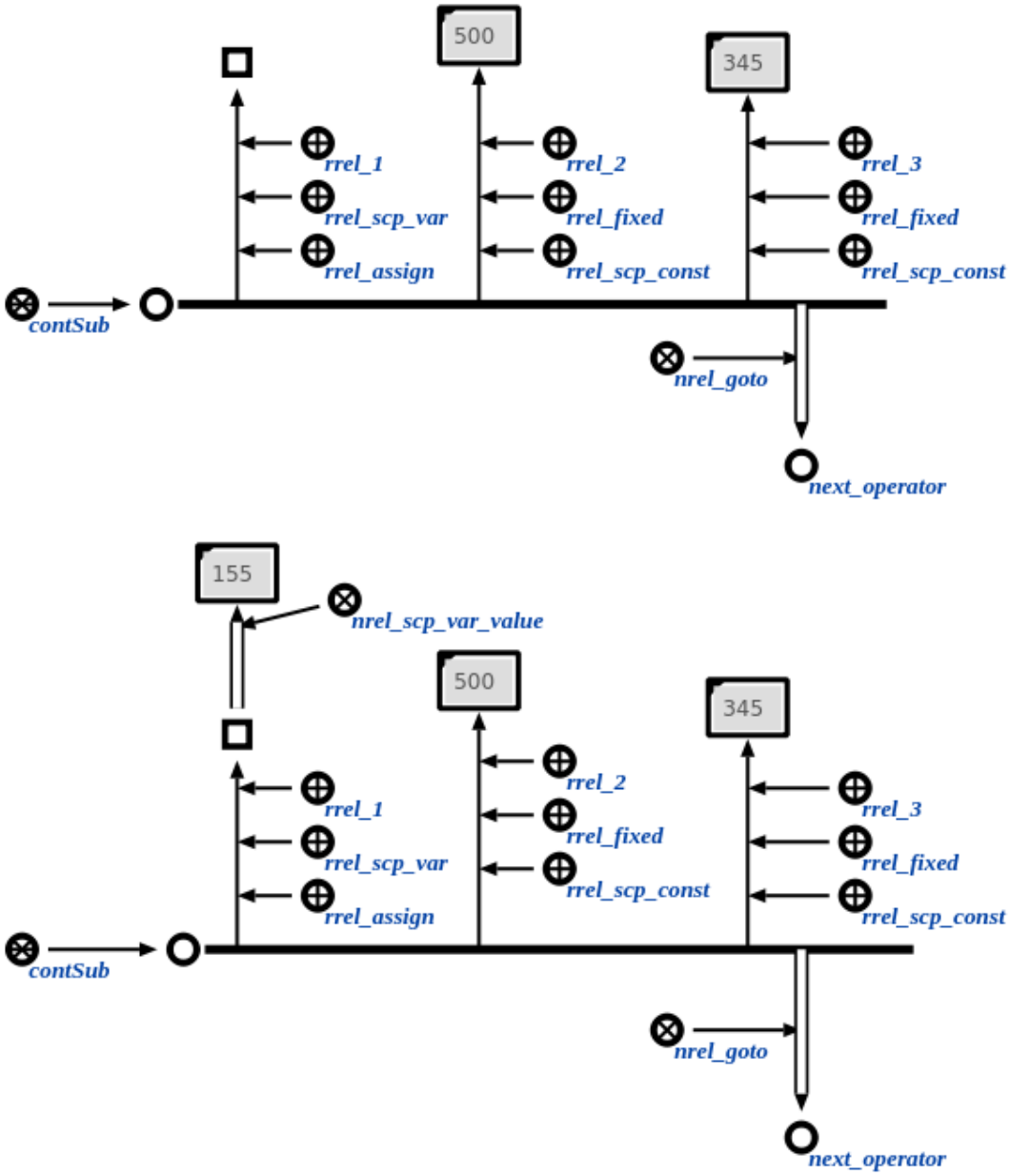



Рис. 3.53 Оператор вычитания числовых содержимых операндов

Операторы класса ***scp-оператор умножения числовых содержимых sc-ссылок*** описывают действие умножения между собой числовых содержимых двух *sc-ссылок*, являющихся значениями второго и третьего операндов.

Каждый оператор данного класса содержит три операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число,

являющееся произведением чисел, задаваемых вторым и третьим операндами. В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то *sc-ссылка* будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся *sc-ссылки*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся одним из множителей при произведении.

Третий операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся одним из множителей при произведении.

```
scp_operator_example_contMult (*
        <- contMult;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [30];;
        -> rrel_3: rrel_fixed: rrel_scp_const: [50];;
        => nrel_goto: next_operator;;
*);;
```

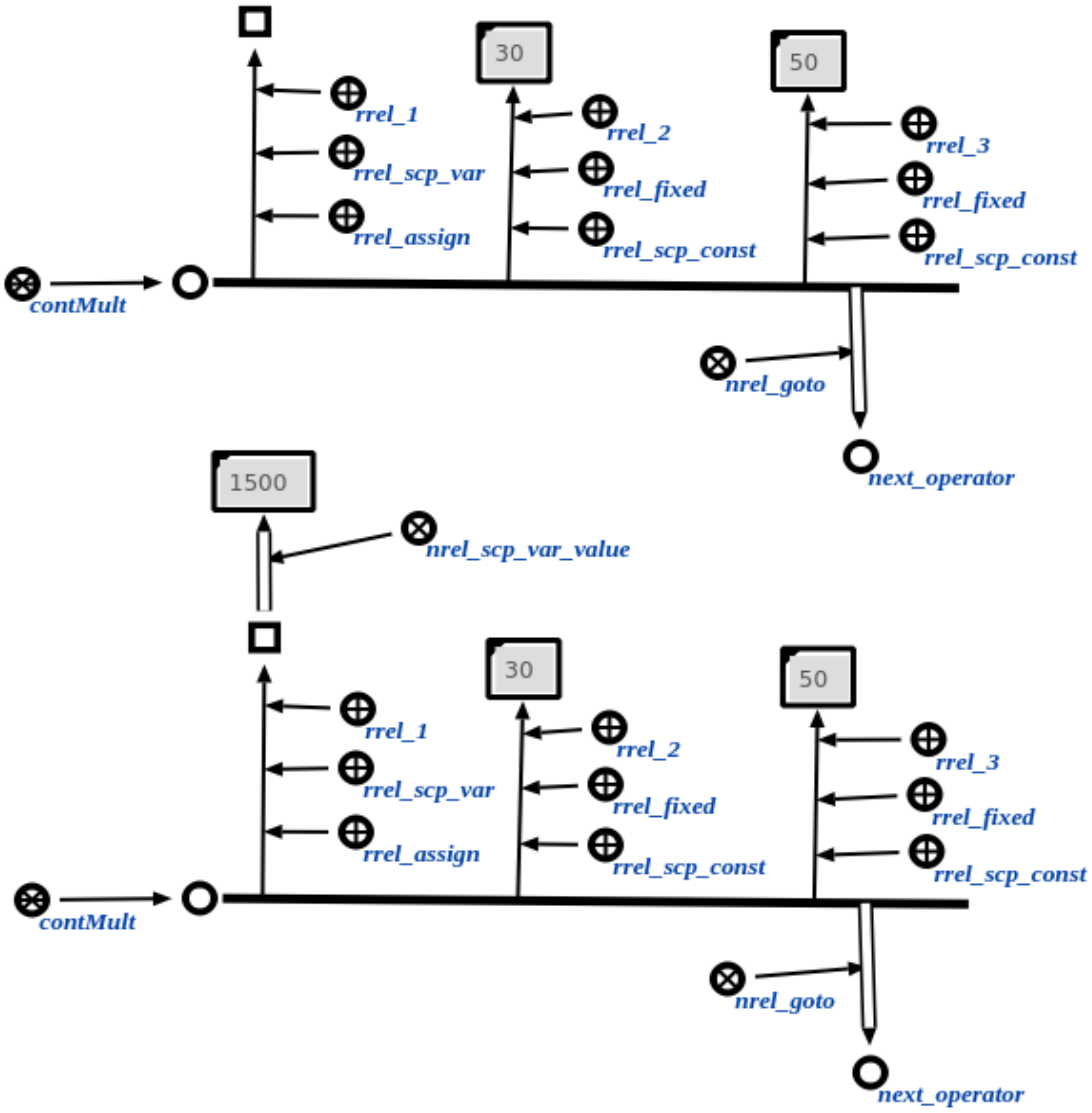



Рис. 3.54 Оператор умножения числовых содержимых операндов

Операторы класса ***scp-оператор деления числовых содержимых sc-ссылок*** описывают действие деления числовых содержимых двух *sc-ссылок*, являющихся значениями второго и третьего операндов.

Каждый оператор данного класса содержит три операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся частным чисел, задаваемых вторым и третьим операндами. В случае, если данный операнд помечен как *scp-операнд со свободным*

*значением'*, то *sc-ссылка* будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся *sc-ссылки*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся делимым при делении.

Третий операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся делителем при делении.

```
scp_operator_example_contDiv (*
        <- contDiv;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [45];;
        -> rrel_3: rrel_fixed: rrel_scp_const: [5];;
        => nrel_goto: next_operator;;
*);;
```

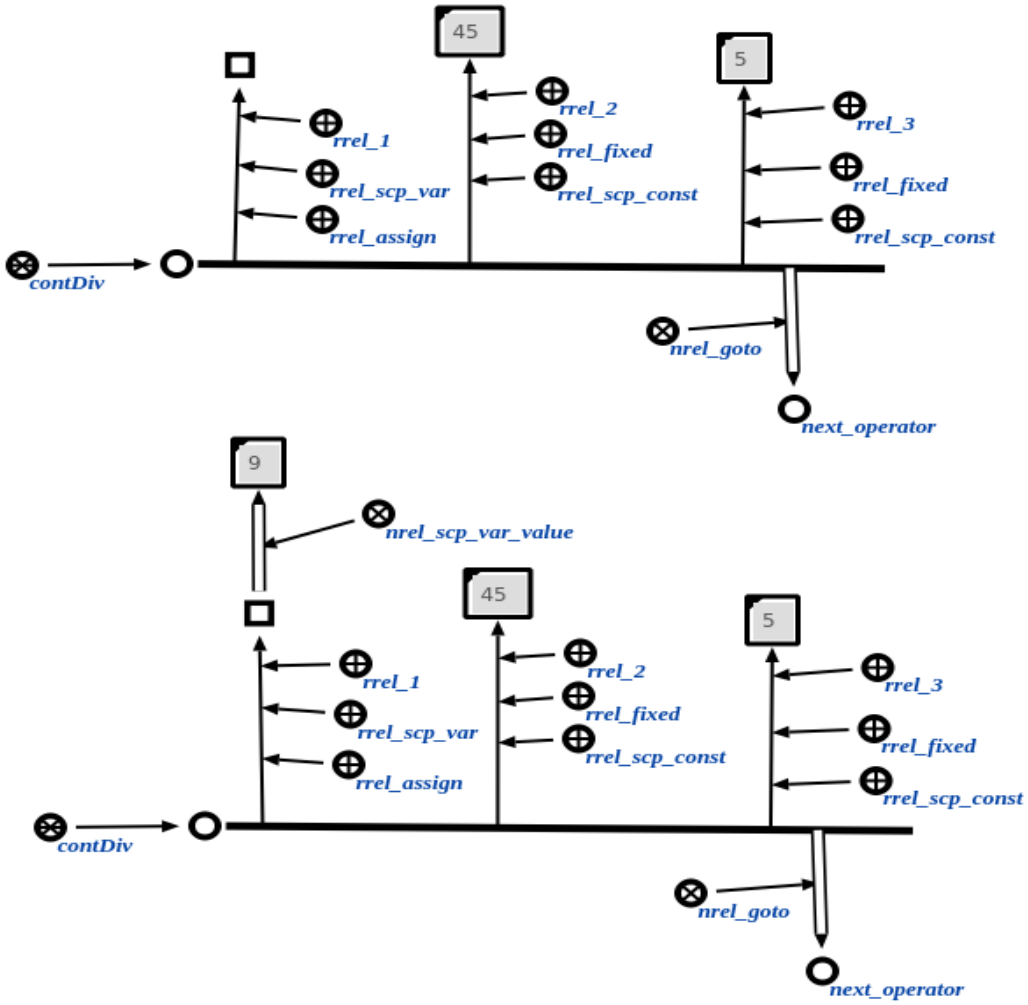



Рис. 3.55 Оператор деления числовых содержимых операндов

Операторы класса ***scp-оператор возведения числового содержимого sc-ссылки в степень*** описывают действие возведения в степень числового содержимого *sc-ссылки*, являющейся значением второго операнда.

Каждый оператор данного класса содержит три операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся результатом возведения числа, задаваемого вторым операндом, в степень, задаваемую третьим операндом. В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то *sc-ссылка* будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся *sc-ссылки*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся основанием при возведении в степень.

Третий операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся показателем при возведении в степень.

```
scp_operator_example_contPow (*
        <- contPow;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [15];;
        -> rrel_3: rrel_fixed: rrel_scp_const: [2];;
        => nrel_goto: next_operator;;
*);;
```

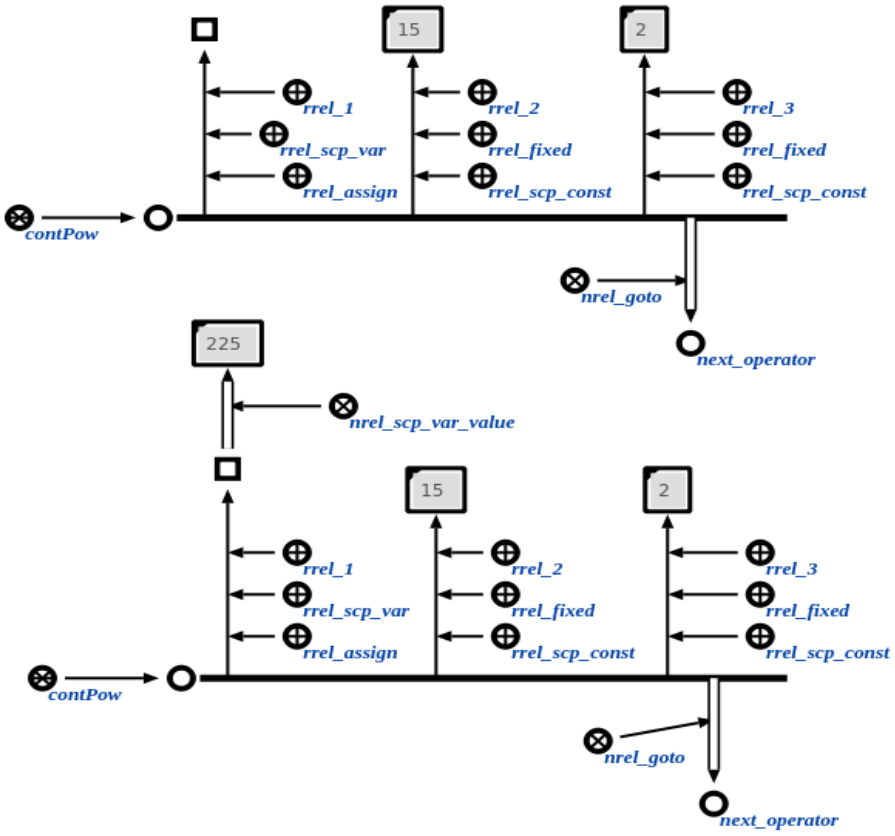



Рис. 3.56 Оператор возведения числовых содержимых операндов в степень

Операторы класса ***scp-оператор вычисления логарифма числового содержимого sc-ссылки*** описывают действие вычисления натурального логарифма числового содержимого *sc-ссылки*, являющейся значением второго операнда.

Каждый оператор данного класса содержит два операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся логарифмом числа, задаваемого вторым операндом. В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то *sc-ссылка* будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся *sc-ссылки*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся аргументом функции логарифма. При этом необходимо следить, чтобы численное содержимое данного операнда идентифицировало число, попадающее в область определения функции логарифма с точки зрения классической алгебры.

```
scp_operator_example_contLn (*
        <- contLn;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [15];;
        => nrel_goto: next_operator;;
*);;
```

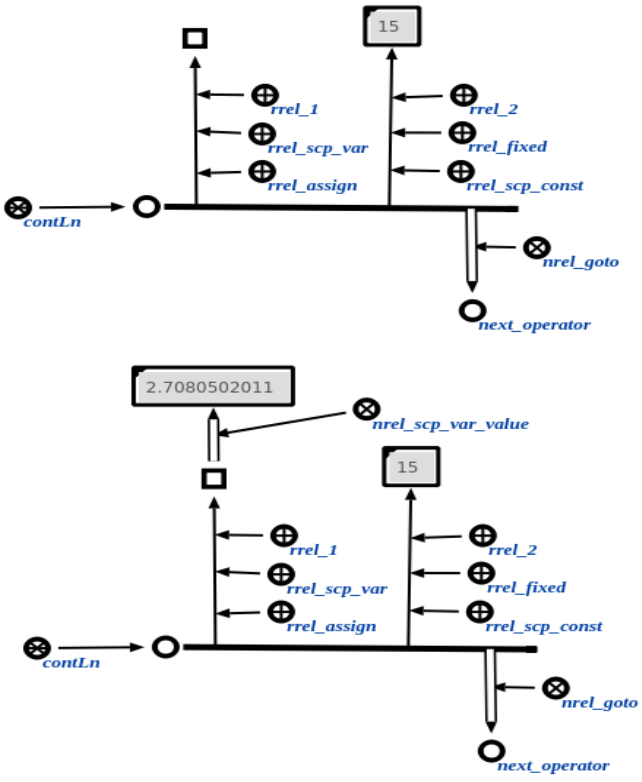



Рис. 3.57 Оператор вычисления логарифма

Операторы класса ***scp-оператор вычисления синуса числового содержимого sc-ссылки*** описывают действие вычисления синуса числового содержимого *sc-ссылки*, являющейся значением второго операнда.

Каждый оператор данного класса содержит два операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся синусом числа, задаваемого вторым операндом. В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то *sc-ссылка* будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся *sc-ссылки*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся аргументом функции синуса. Предполагается, что указанное число соответствует значению угла в радианах.

```
scp_operator_example_contSin (*
        <- contSin;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [30];;
        => nrel_goto: next_operator;;
*);;
```

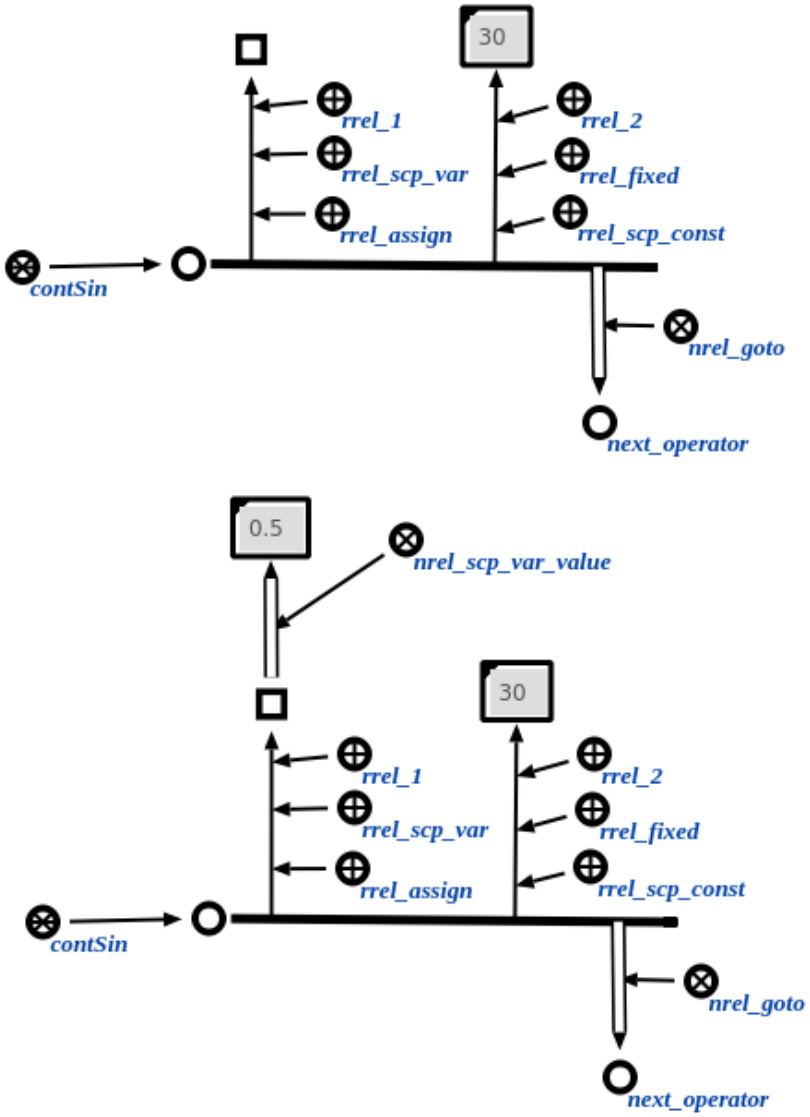



Рис. 3.58 Оператор вычисления синуса

Операторы класса ***scp-оператор вычисления косинуса числового содержимого sc-ссылки*** описывают действие вычисления косинуса числового содержимого *sc-ссылки*, являющейся значением второго операнда.

Каждый оператор данного класса содержит два операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся косинусом числа, задаваемого вторым операндом. В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то *sc-*

*ссылка* будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся *sc-ссылки*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся аргументом функции косинуса. Предполагается, что указанное число соответствует значению угла в радианах.

```
scp_operator_example_contCos (*
        <- contCos;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [60];;
        => nrel_goto: next_operator;;
*);;
```

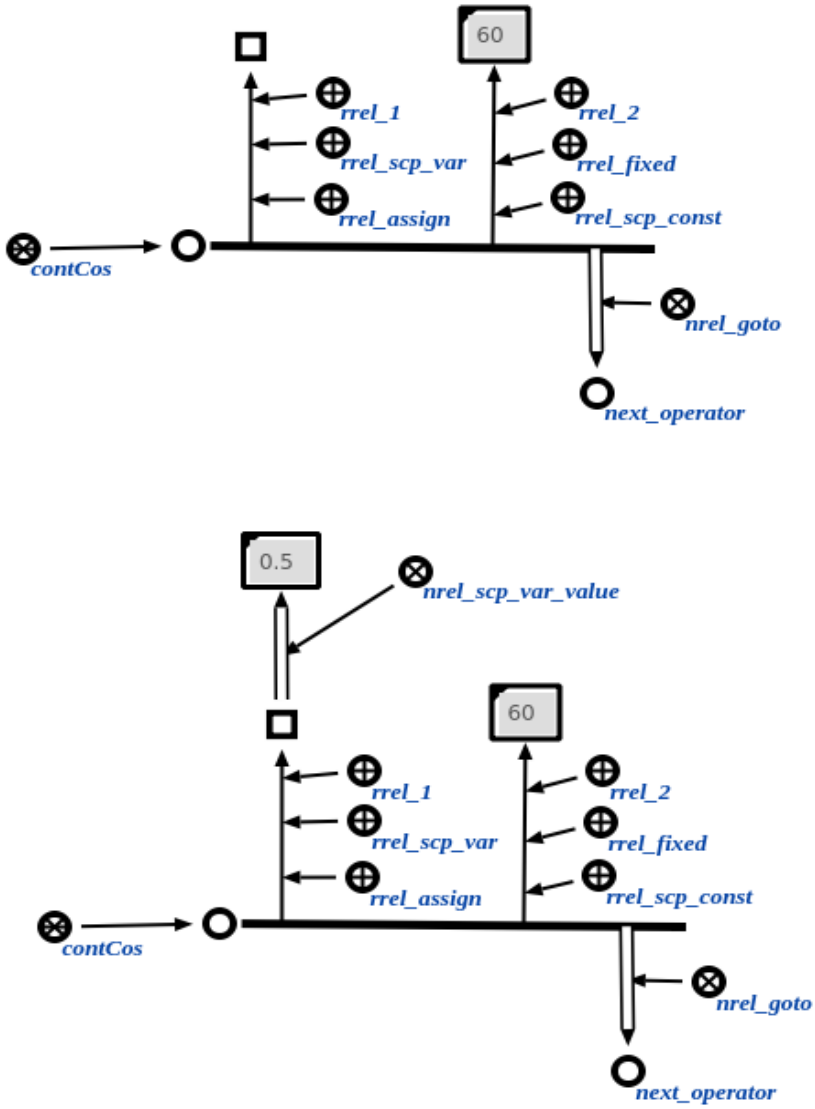



Рис. 3.59 Оператор вычисления косинуса

Операторы класса ***scp-оператор вычисления тангенса числового содержимого sc-ссылки*** описывают действие вычисления тангенса числового содержимого *sc-ссылки*, являющейся значением второго операнда.

Каждый оператор данного класса содержит два операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся тангенсом числа, задаваемого вторым операндом. В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то *sc-ссылка* будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся *sc-ссылки*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся аргументом функции тангенса. Предполагается, что указанное число соответствует значению угла в радианах.

```
scp_operator_example_contTg (*
        <- contTg;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [45];;
        => nrel_goto: next_operator;;
*);;
```

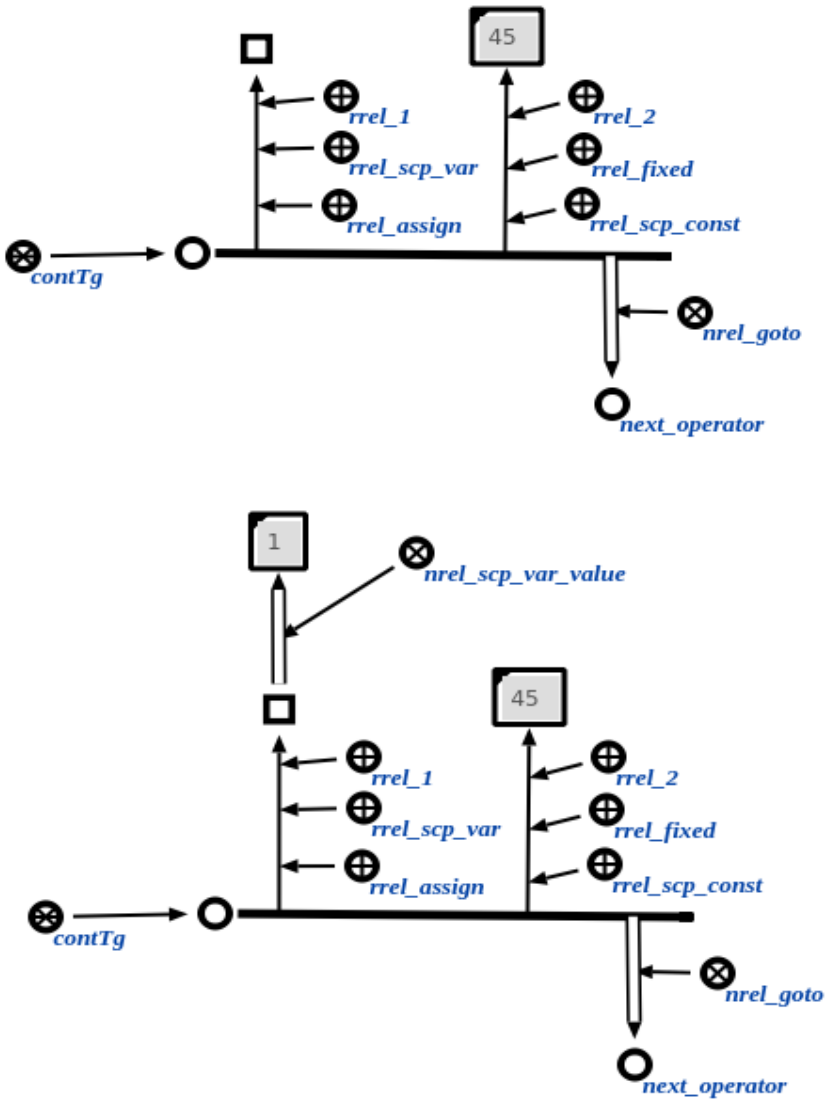



Рис. 3.60 Оператор вычисления тангенса

Операторы класса ***scp-оператор вычисления арксинуса числового содержимого sc-ссылки*** описывают действие вычисления арксинуса числового содержимого *sc-ссылки*, являющейся значением второго операнда.

Каждый оператор данного класса содержит два операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся арксинусом числа, задаваемого вторым операндом. В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то *sc-ссылка* будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся *sc-ссылки*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся аргументом функции арксинуса. При этом необходимо следить, чтобы численное содержимое данного операнда

идентифицировало число, попадающее в область определения функции арксинуса с точки зрения классической алгебры.

Предполагается, что результат вычисления соответствует значению угла в радианах.

```
scp_operator_example_contASin (*
        <- contASin;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [0.5];;
        => nrel_goto: next_operator;;
*);;
```

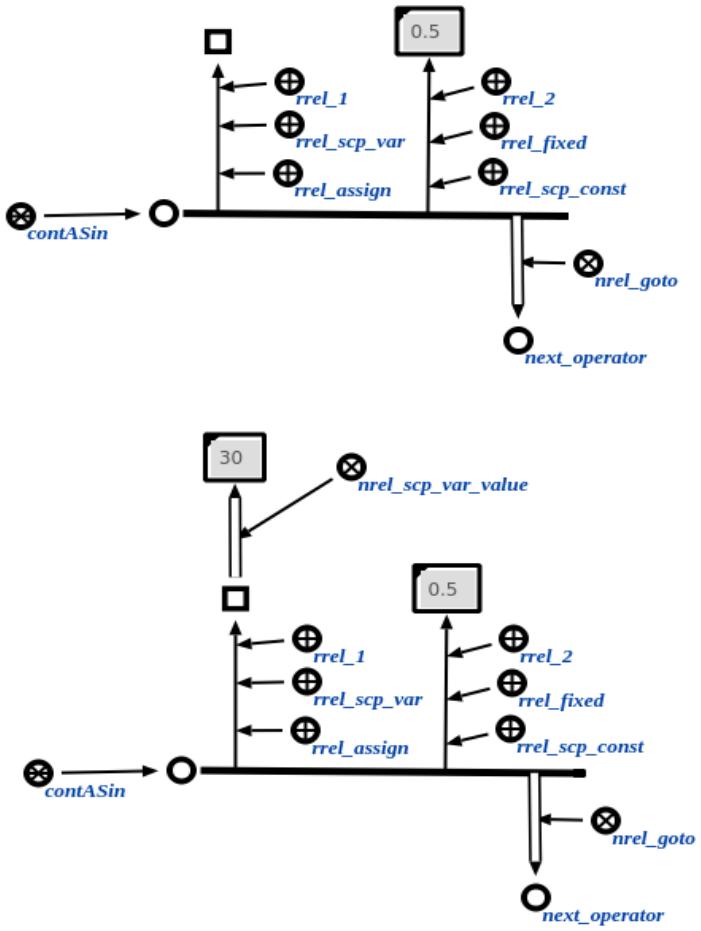



Рис. 3.61 Оператор вычисления арксинуса

Операторы класса ***scp-оператор вычисления арккосинуса числового содержимого sc-ссылки*** описывают действие вычисления арккосинуса числового содержимого *sc-ссылки*, являющейся значением второго операнда.

Каждый оператор данного класса содержит два операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся арксинусом числа, задаваемого вторым операндом. В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то *sc-*

*ссылка* будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся *sc-ссылки*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся аргументом функции арккосинуса. При этом необходимо следить, чтобы численное содержимое данного операнда идентифицировало число, попадающее в область определения функции арккосинуса с точки зрения классической алгебры.

Предполагается, что результат вычисления соответствует значению угла в радианах.

```
scp_operator_example_contACos (*
        <- contACos;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [0.5];;
        => nrel_goto: next_operator;;
*);;
```

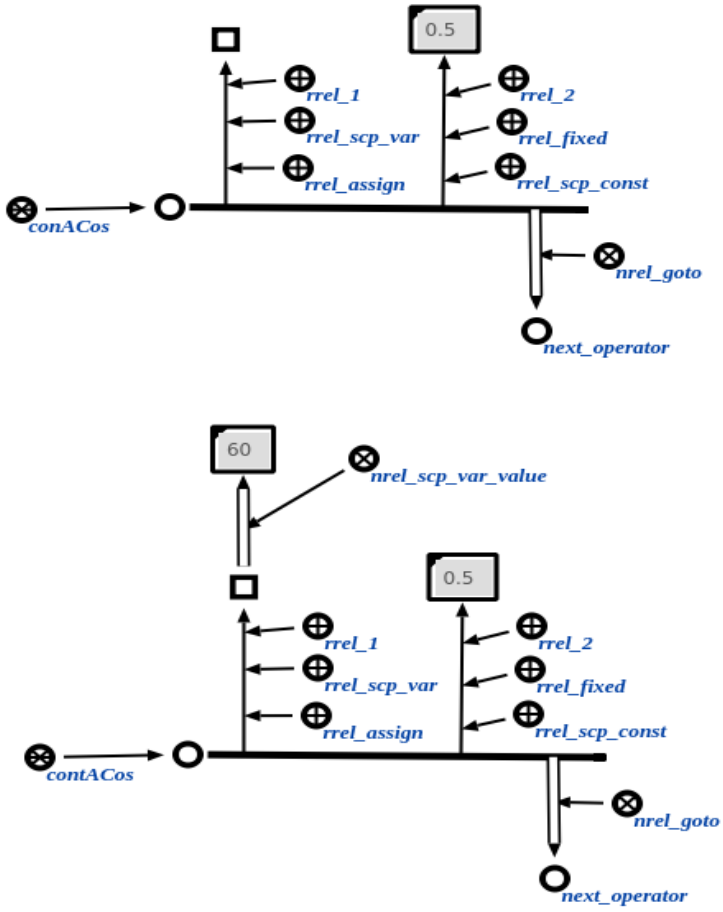



Рис. 3.62 Оператор вычисления арккосинуса

Операторы класса ***scp-оператор вычисления арктангенса числового содержимого sc-ссылки*** описывают действие вычисления арктангенса числового содержимого *sc-ссылки*, являющейся значением второго операнда.

Каждый оператор данного класса содержит два операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся арктангенсом числа, задаваемого вторым операндом. В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то *sc-ссылка* будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся *sc-ссылки*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, идентифицирующая число, являющееся аргументом функции арктангенса. При этом необходимо следить, чтобы численное содержимое данного операнда идентифицировало число, попадающее в область определения функции арктангенса с точки зрения классической алгебры.

Предполагается, что результат вычисления соответствует значению угла в радианах.

```
scp_operator_example_contATg (*
        <- contATg;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [1];;
        => nrel_goto: next_operator;;
*);;
```

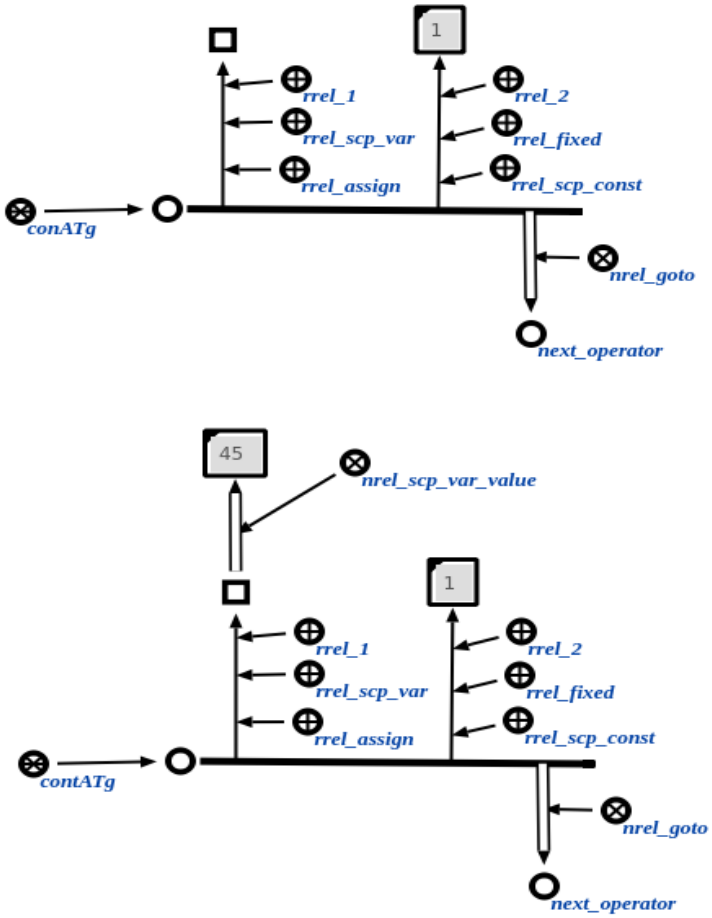



Рис. 3.63 Оператор вычисления арктангенса

Операторы класса ***scp-оператор копирования содержимого sc-ссылки*** описывают действие копирования произвольного содержимого одной *sc-ссылки* в содержимое другой *sc-ссылки*.

Каждый оператор данного класса содержит два операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. Значением данного операнда является *sc-ссылка*, в которую будет записано содержимое в результате копирования. В случае, если данный операнд помечен как *scp-операнд со свободным значением'*, то *sc-ссылка* будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся *sc-ссылки*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, из которой будет взято содержимое для копирования.

```
scp_operator_example_contAssign (*
        <- contAdd;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [30];;
        => nrel_goto: next_operator;;
*);;
```

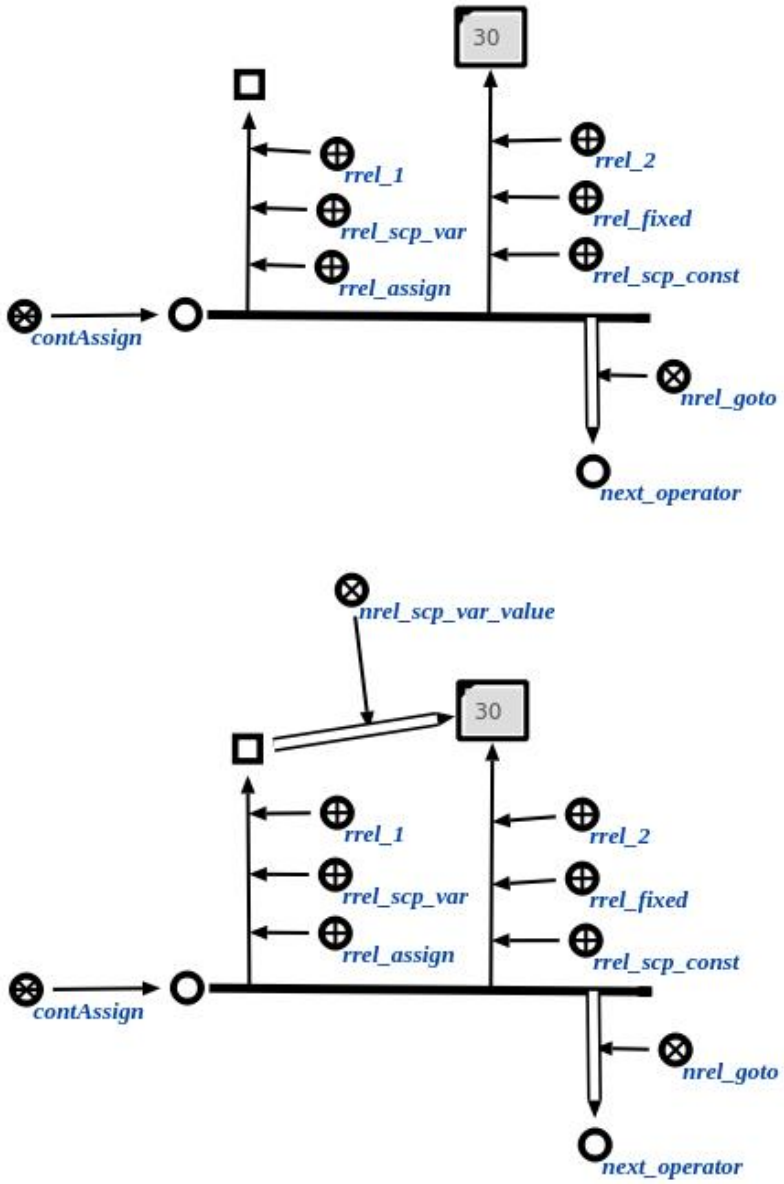



Рис. 3.64 Оператор копирования содержимого sc-ссылки (Пример 1)

```
scp_operator_example_contAssign (*
        <- contAssign;;
        -> rrel_1: rrel_fixed: rrel_scp_const: el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [Hello world!];;
        => nrel_goto: next_operator;;
*);;
```

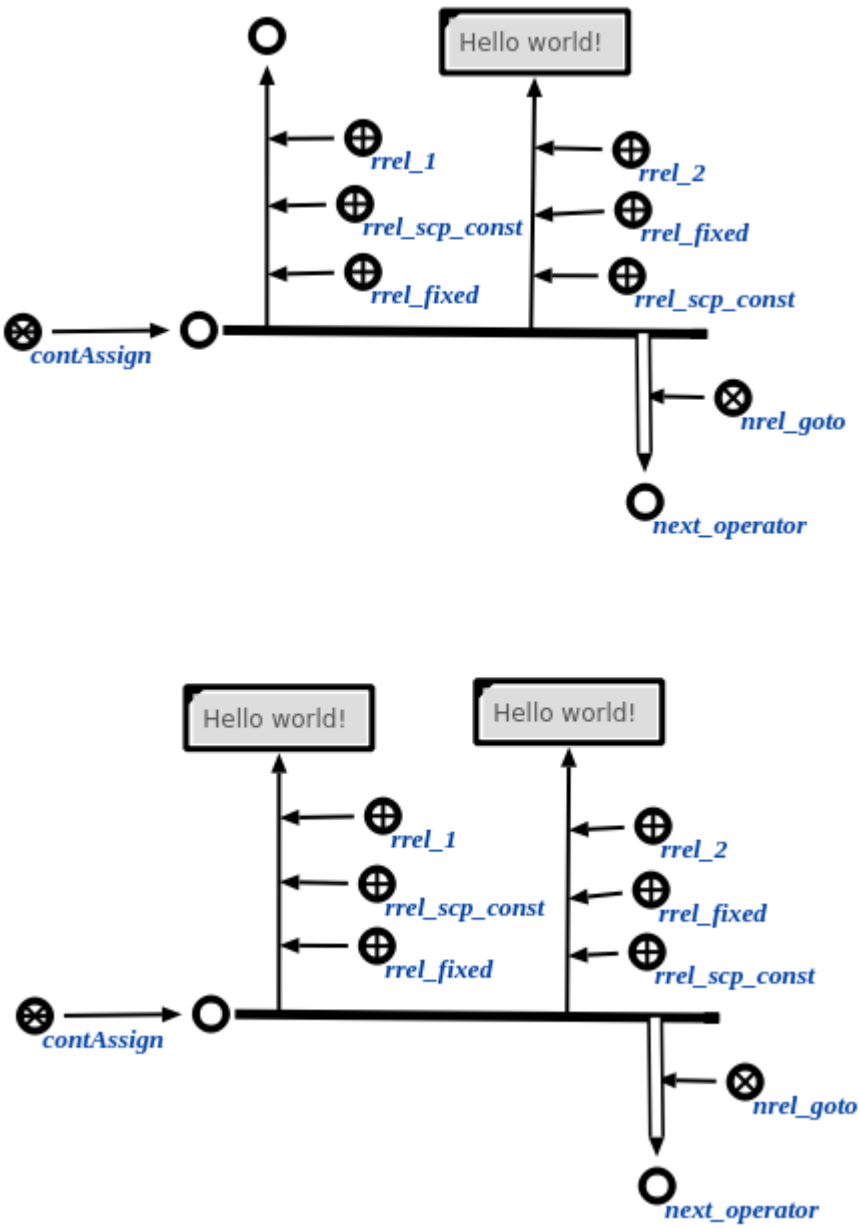



Рис. 3.65 Оператор копирования содержимого sc-ссылки (Пример 2)

Операторы класса ***scp-оператор удаления содержимого sc-ссылки*** описывают действие удаления произвольного содержимого некоторой *sc-ссылки*.

Каждый оператор данного класса содержит один операнд. Данный операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, содержимое которой будет удалено. После выполнения оператора указанная *sc-ссылка* будет считаться *sc-ссылкой*, для которой не сформировано содержимое, то есть не существует соответствующего ей файла. При этом сама *sc-ссылка* не удаляется.

```
scp_operator_example_contErase (*
        <- contErase;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        => nrel_goto: next_operator;;
*);;
```

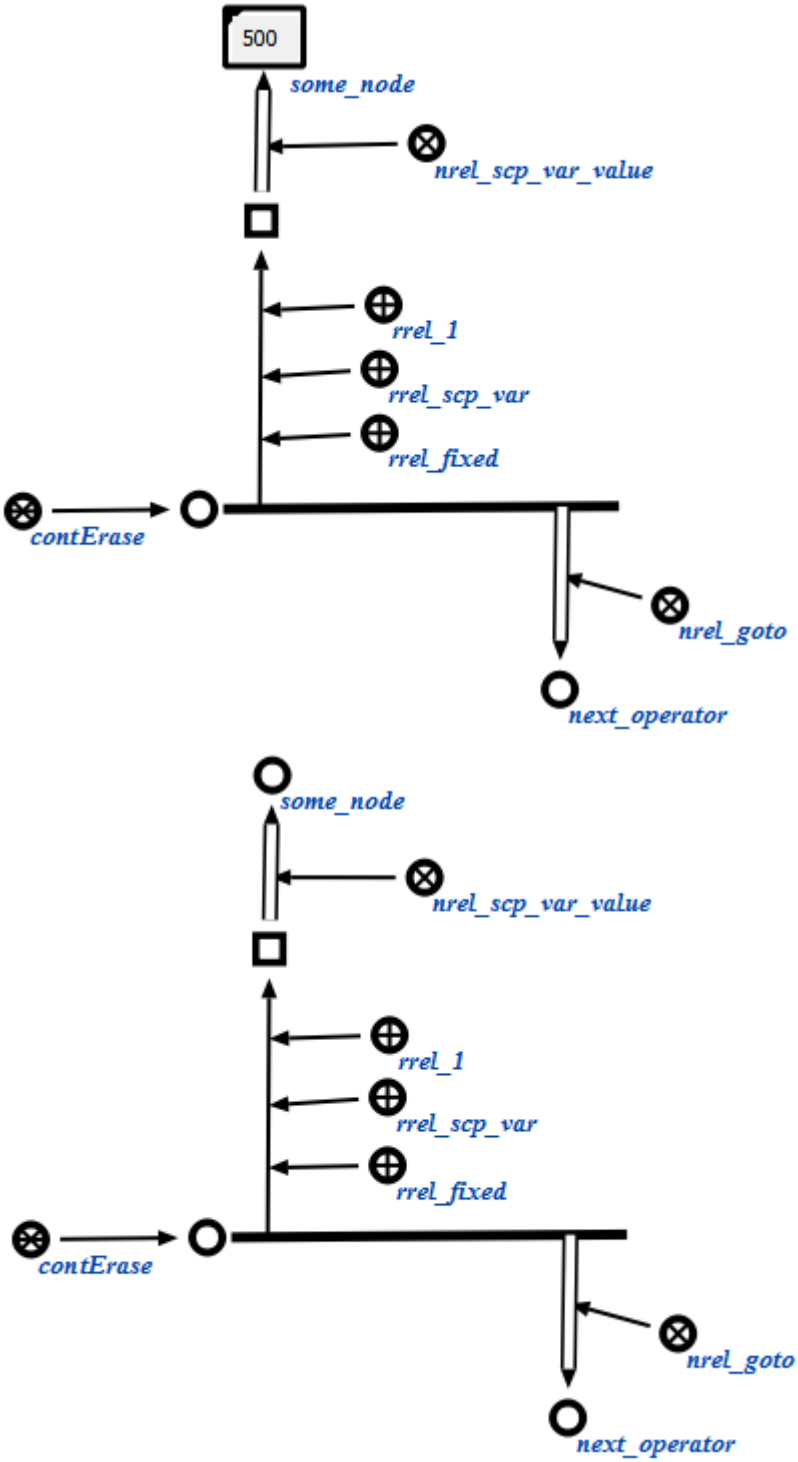



Рис. 3.66 Оператор удаления содержимого sc-ссылки (Пример 1)

```
scp_operator_example_contErase (*
        <- contErase;;
        -> rrel_1: rrel_fixed: rrel_scp_const: [Hello world];;
        => nrel_goto: next_operator;;
*);;
```

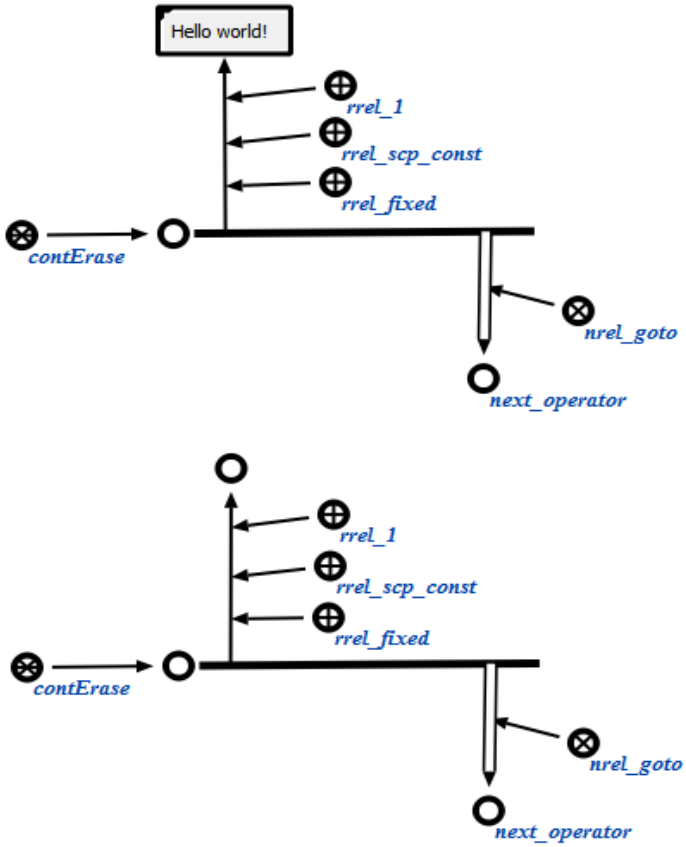



Рис. 3.67 Оператор удаления содержимого sc-ссылки (Пример 1)

### 3.6 Scp-операторы управления событиями

Операторы класса ***scp-оператор управления событиями*** описывают действия, связанные с *sc-событиями*.

Операторы класса не предполагают ветвления программы в зависимости от выполнения дополнительных условий, поэтому указание следующих операторов осуществляется только при помощи отношения *следующий оператор**, без подмножеств.

В данном классе *scp-операторов* выделяется один оператор:

- ***scp-оператор ожидания события.***

Операторы класса ***scp-оператор ожидания события*** описывают действие ожидания возникновения в *sc-памяти* одного из базовых типов *sc-событий*, связанных с каким-либо *sc-элементом*. Если при выполнении *scp-программы* встретился *scp-оператор* данного класса, выполнение *scp-программы* приостанавливается до тех пор, пока в *sc-памяти* не произойдет *sc-событие*,

описываемое значениями операндов. При этом учитываются только события, произошедшие после того момента, как при выполнении *scp-программы* встретился s*cp-оператор* данного класса, события, произошедшие ранее, не учитываются.

Каждый оператор данного класса содержит два операнда.

Первый операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является один из базовых типов *sc-событий*, т.е. одно из подмножеств множества *sc-событий*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-элемент*, с которым непосредственно связан указанный тип *sc-события*, т.е. элемент, из которого выходит либо в который входит генерируемая либо удаляемая *sc-дуга*, либо *sc-ссылка*, содержимое которой было изменено и т.п.

```
scp_operator_example_sys_wait (*
        <- sys_wait;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_var: _el2;;
        => nrel_goto: next_operator;;
*);;
```

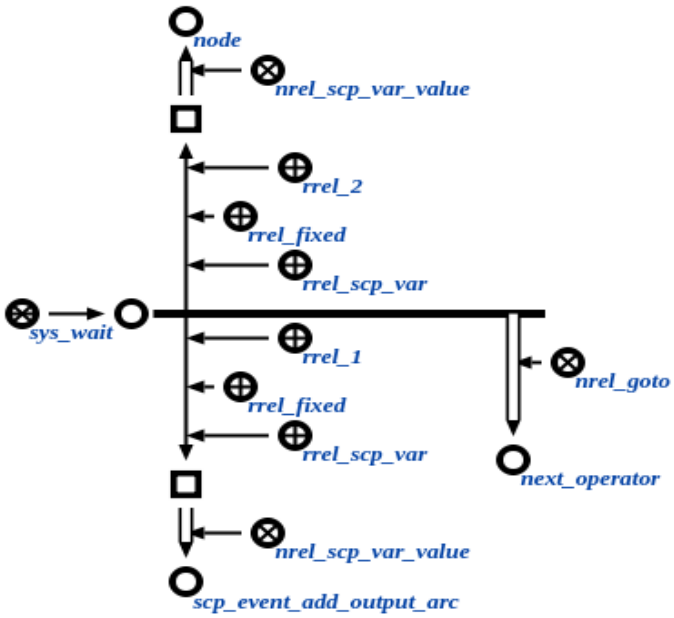



Рис. 3.68 Оператор ожидания события

### 3.7 Scp-операторы управления scp-процессами

Операторы класса ***scp-оператор управления scp-процессами*** описывают действия, связанные с вызовом подпрограмм (т.е. созданием *scp-процессов*), а также управления выполнением *scp-программ*.

Операторы класса не предполагают ветвления программы в зависимости от выполнения дополнительных условий, поэтому указание следующих операторов осуществляется только при помощи отношения *следующий оператор**, без подмножеств.

Данный класс *scp-операторов* разбивается на следующие подклассы:

- ***scp-оператор асинхронного вызова подпрограммы;***
- ***scp-оператор ожидания завершения выполнения scp-программы;***
- ***scp-оператор ожидания завершения выполнения множества scp-программ;***
- ***scp-оператор завершения выполнения программы.***

Под ***scp-процессом*** понимается некоторая структура в *sc-памяти*, однозначно описывающая конкретный акт выполнения некоторой *scp-программы* для заданных исходных данных. Если *scp-программа* описывает алгоритм решения какой-либо задачи в общем виде, то ***scp-процесс*** описывает конкретную реализацию данного алгоритма для заданных входных параметров. В зависимости от конкретной реализации *scp-интерпретатора* структура *scp-процесса* может различаться.

Операторы класса ***scp-оператор асинхронного вызова подпрограммы*** описывают действие вызова подпрограммы, то есть создания уникального *scp-процесса*, соответствующего указанной *scp-программе*. Вызов является асинхронным, то есть выполнение дочернего *scp-процесса* начинается сразу же после его создания, параллельно с родительским *scp-процессом*. Каждый оператор данного класса содержит три операнда.

Первый операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является знак scp-программы, вызов которой необходимо осуществить.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является множество параметров, передаваемых в вызываемую подпрограмму. Количество элементов данного множества должно совпадать с количеством параметров вызываемой *scp-программы*, все элементы упорядочиваются при помощи ролевых отношений *1'*, *2'*, *3'*. Элемент, помеченный одним из указанных ролевых отношений, соответствует параметру, помеченному тем же ролевым отношением. Элементы, соответствующие *in-параметрам'* должны быть помечены как *scp-операнд с заданным значением'*. Элементы, соответствующие *out-параметрам'*, могут иметь произвольный *тип значения scp-операнда'*.

Третий операнд должен быть помечен как *scp-операнд со свободным значением'*. Значением данного операнда является знак созданного *scp-*

*процесса*, который далее может быть использован для того, чтобы дождаться завершения созданного *scp-процесса*. Следует отметить, что каждый созданный *scp-процесс* существует в *sc-памяти* до того момента как был выполнен один из операторов ожидания завершения *scp-процессов*. В связи с этим при помощи соответствующих *scp-операторов* необходимо обеспечить в конечном итоге ожидание завершения всех созданных во время выполнения *scp-программы* *scp-процессов* (необязательно одновременно и необязательно в рамках одной *scp-программы*). Нарушение данного правила ведет к засорению *sc-памяти* конструкциями, описывающими созданные *scp-процессы*.

```
scp_operator_example_call (*
        <- call;;
        -> rrel_1: rrel_fixed: rrel_scp_const: proc_search_all_output;;
        -> rrel_2: rrel_fixed: rrel_scp_const: scp_operator_example_call_params
(*
                -> rrel_1: rrel_fixed: rrel_scp_var: _param1;;
                -> rrel_2: rrel_fixed: rrel_scp_var: _param2;;
*);;
        -> rrel_3: rrel_assign: rrel_scp_var: _desrc;;
        => nrel_goto: next_operator;;
*);;
```

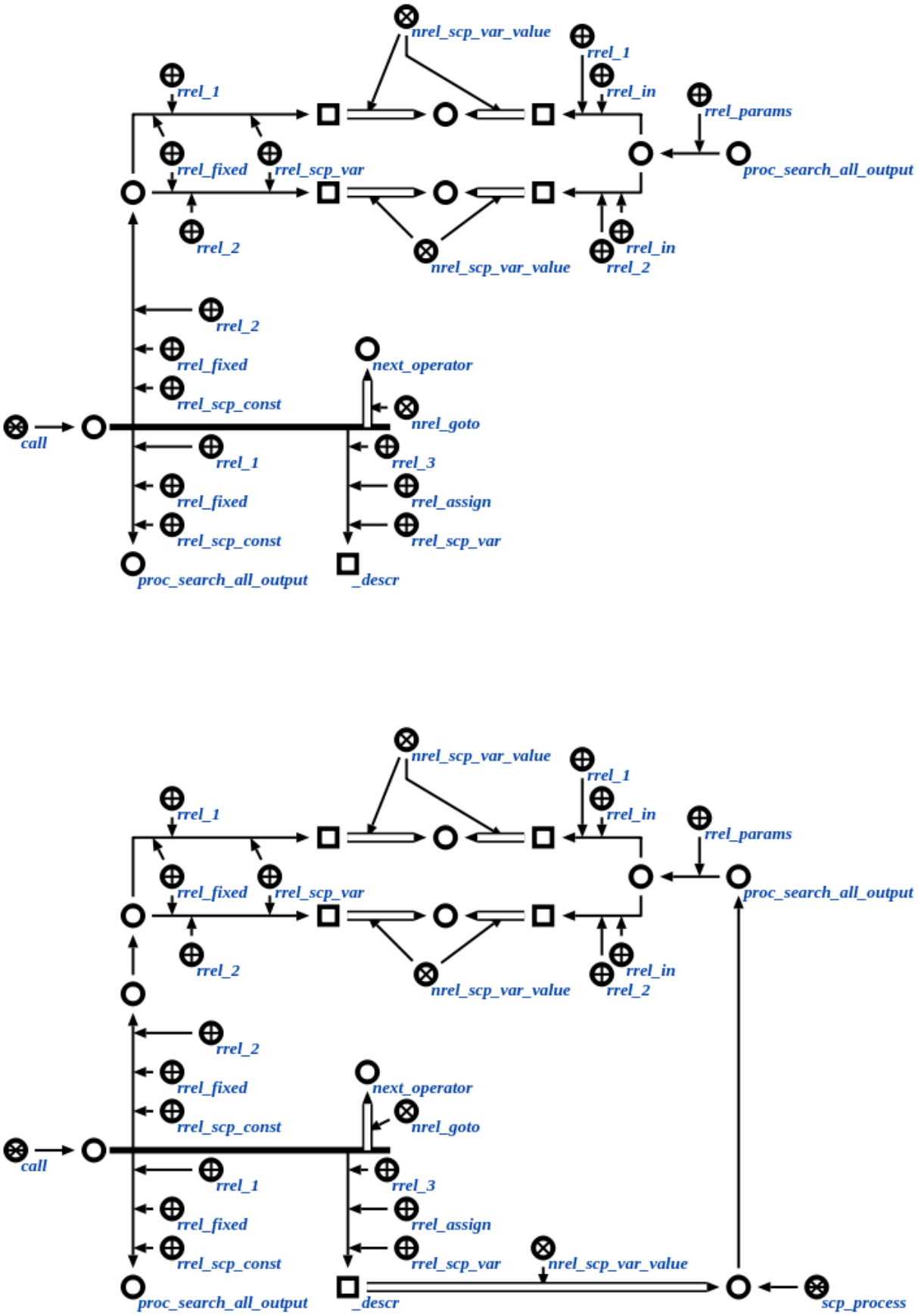



Рис. 3.69 Оператор асинхронного вызова подпрограмм

Операторы класса ***scp-оператор ожидания завершения выполнения scp-программы*** описывают действие ожидания завершения выполнения вызванной *scp-программы*, то есть *scp-процесса*, созданного при выполнении *scp-оператора асинхронного вызова подпрограммы*. Если *scp-процесс* завершился до того, как началось выполнение *scp-оператора* данного класса, то будет выполнен *следующий scp-оператор**, в противном случае *scp-программа* приостановит свое выполнение до того, как ожидаемый *scp-процесс* не завершится. После выполнения *scp-оператора* данного класса конструкция, описывающая заданный *scp-процесс* будет удалена из *sc-памяти*. В связи с этим запрещается использование нескольких *scp-операторов* данного класса для знака одного и того же *scp-процесса*.

Каждый оператор данного класса содержит один операнд, значением которого является знак *scp-процесса*, завершения которого необходимо дождаться. Операнд должен быть помечен как *scp-операнд с заданным значением'*.

```
scp_operator_example_waitReturn (*
        <- waitReturn;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _descr;;

        => nrel_goto: next_operator;;
*);;
```

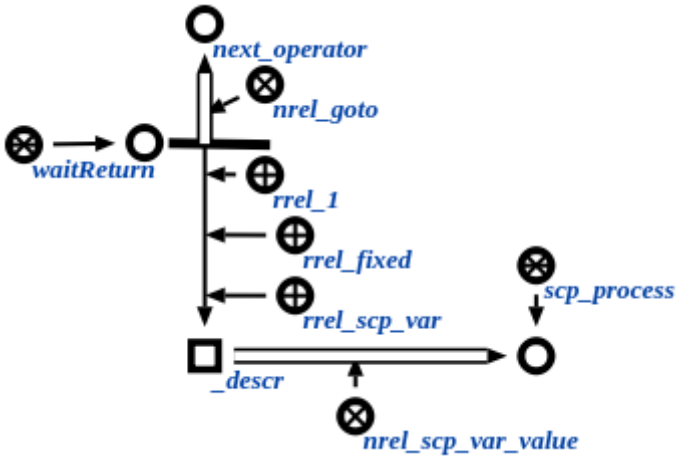



Рис. 3.70 Оператор ожидания завершения выполнения scp-процесса

Операторы класса ***scp-оператор ожидания завершения выполнения множества scp-процессов*** описывают действие ожидания завершения выполнения множества вызванных *scp-программ*, то есть *scp-процессов*, созданных при выполнении *scp-операторов асинхронного вызова подпрограммы*.

Каждый оператор данного класса содержит один операнд, значением которого является знак множества *scp-процессов*, завершения которых необходимо дождаться. Операнд должен быть помечен как *scp-операнд с заданным значением'*.

Scp-процессы удаляются из множества по мере своего завершения, до тех пор, пока множество не станет пустым. После этого уничтожается знак самого множества и начинается выполнение *следующего scp-оператора**,

```
scp_operator_example_waitReturnSet (*
        <- waitReturnSet;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _wait_prog_set;;

        => nrel_goto: next_operator;;
*);;
```

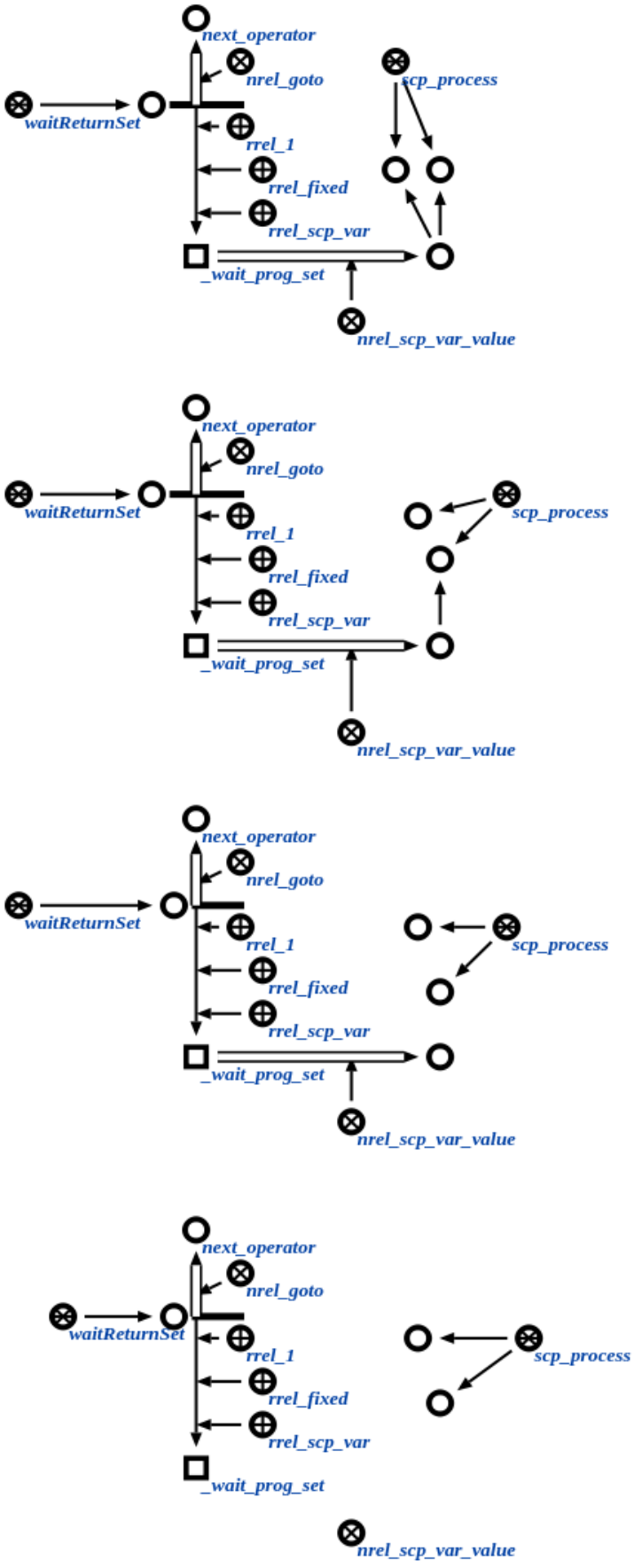



Рис. 3.71 Оператор ожидания завершения множества scp-процесса

Операторы класса ***scp-оператор завершения выполнения программы*** описывают действие завершения выполнения текущей *scp-программы*. Операторы данного класса не содержат операндов, и завершают выполнение той *scp-программы*, во множество *операторов'* которой они входят.

Каждой *scp-программе* могут соответствовать несколько операторов данного класса, каждый из которых может быть достигнут в различных случаях ветвления программы, но любой из указанных операторов завершает выполнение данной *scp-программы*. В связи с этим *scp-программа*, в которой оператор данного класса должен быть выполнен параллельно с каким-либо другим *scp-оператором,* может рассматриваться как некорректная.

### 3.8 Scp-операторы управления значениями операндов

Операторы класса ***scp-оператор управления значениями операндов*** описывают действия, связанные со связками отношения *значение**, но не затрагивающими конкретные *sc-элементы*, являющиеся значениями операндов данного класса операторов.

Данный класс *scp-операторов* разбивается на следующие подклассы:

- ***scp-оператор присваивания значения переменной;***
- ***scp-оператор удаления значения переменной.***

Операторы класса ***scp-оператор присваивания значения переменной*** описывают действие присваивания значения некоторой *scp-переменной'*.

Каждый оператор данного класса содержит два операнда.

Первый операнд может иметь произвольный *тип значения scp-операнда'*. После выполнения оператора значением данного операнда станет значение второго операнда, то есть будет сгенерирована новая связка отношения *значение**,соединяющая данный операнд и *sc-элемент*, являющийся значением второго операнда. Данный операнд должен быть помечен как *scp-переменная'*.

Второй операнд должен быть помечен как *scp-операнд с заданным значением'*. При этом данный операнд может быть как *scp-переменной'*, так и *scp-константой'*.

```
scp_operator_example_varAssign (*
        <- varAssign;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: some_node;;
        => nrel_goto: next_operator;;
*);;
```

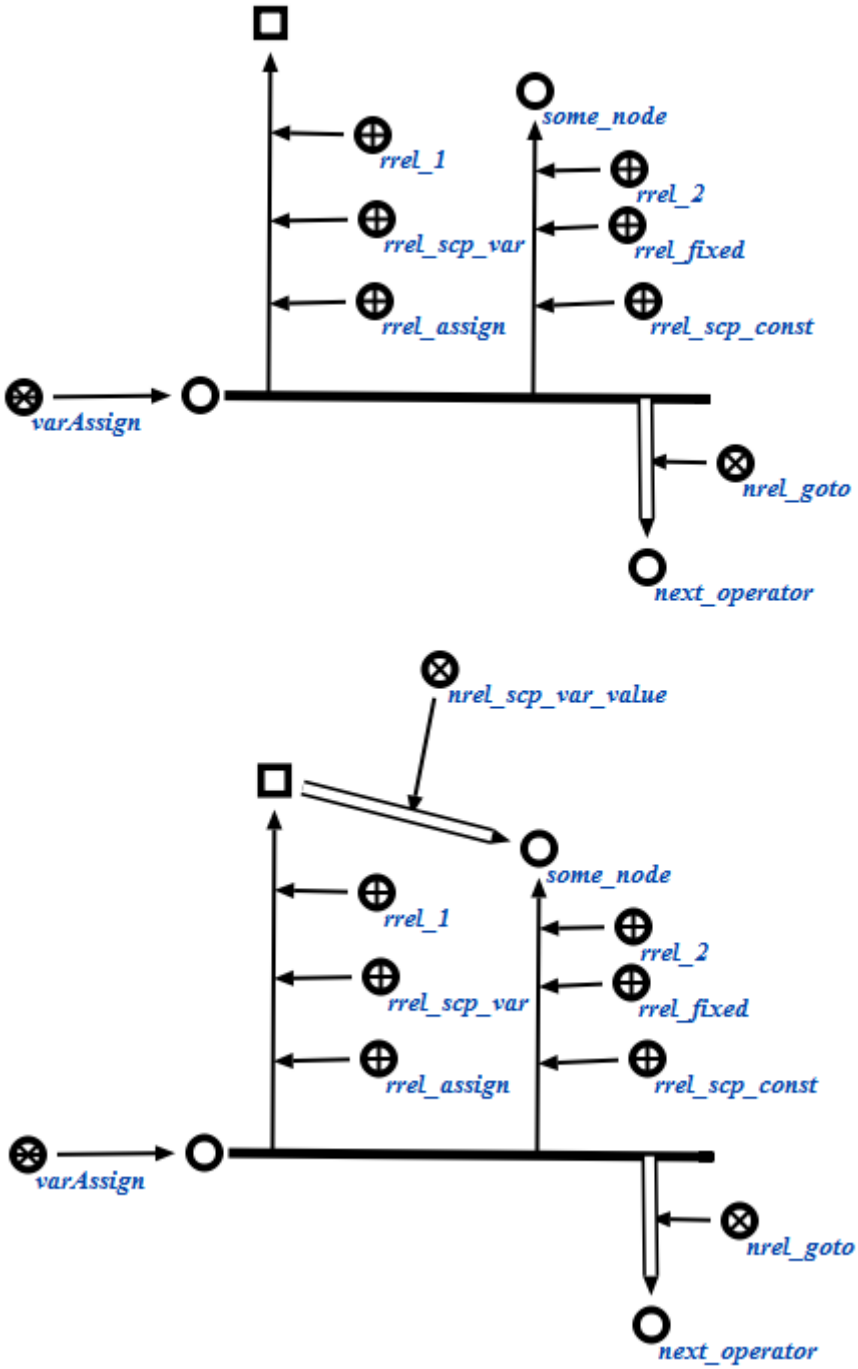



Рис. 3.72 Оператор присваивания значения scp-переменной (Пример 1)

```
scp_operator_example_varAssign (*
        <- varAssign;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_var: _some_node;;
        => nrel_goto: next_operator;;
*);;
```

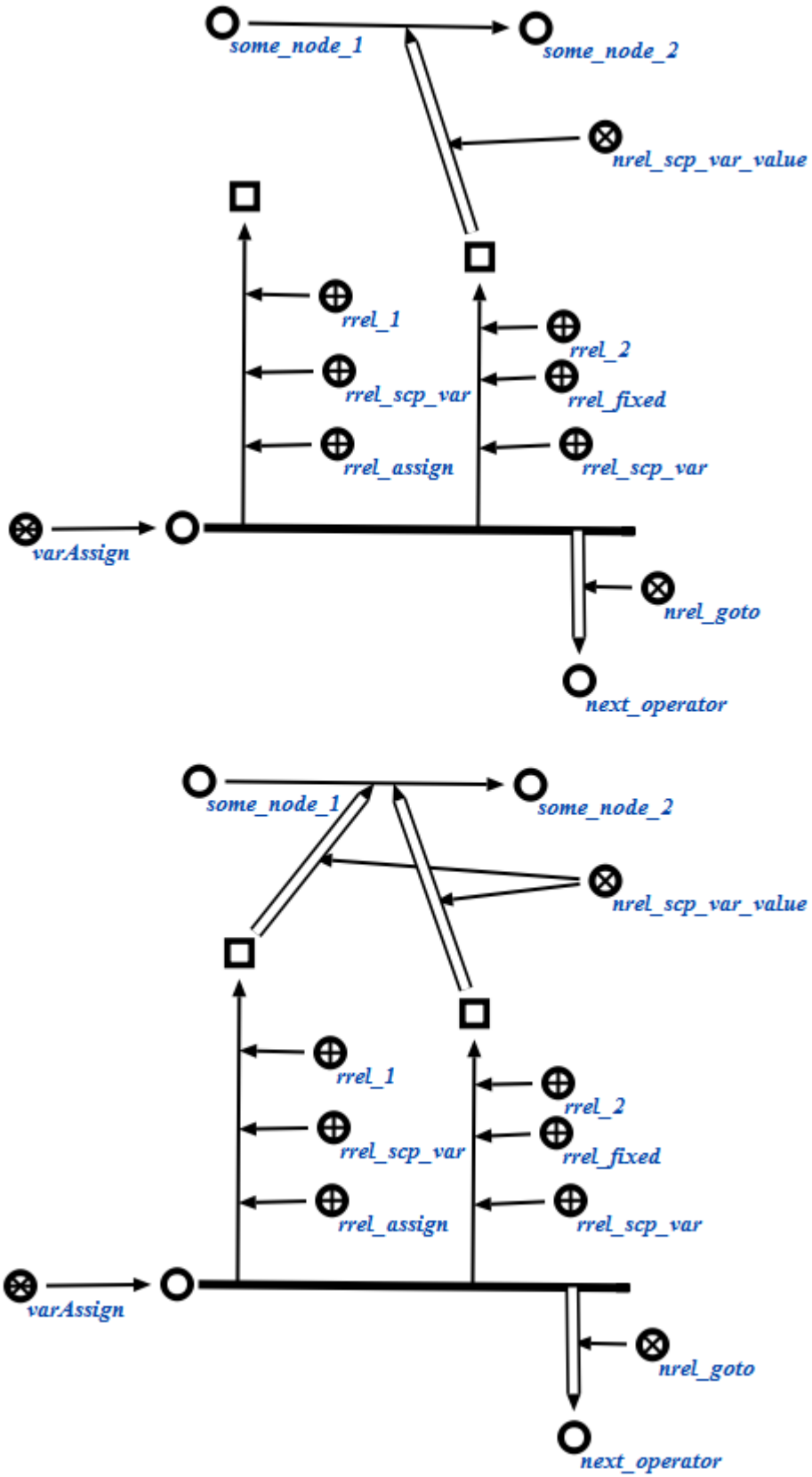



Рис. 3.73 Оператор присваивания значения scp-переменной (Пример 2)

Операторы класса ***scp-оператор удаления значения переменной*** описывают действие удаления связки отношения *значение** для некоторой *scp-переменной'*. Каждый оператор данного класса содержит один операнд, который должен быть помечен как *scp-операнд с заданным значением'* и как *scp-переменная'*. После выполнения оператора будет удалена связка отношения *значение**, связывавшая указанный операнд и его значение.

```
scp_operator_example_varErase (*
        <- varErase;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        => nrel_goto: next_operator;;
*);;
```

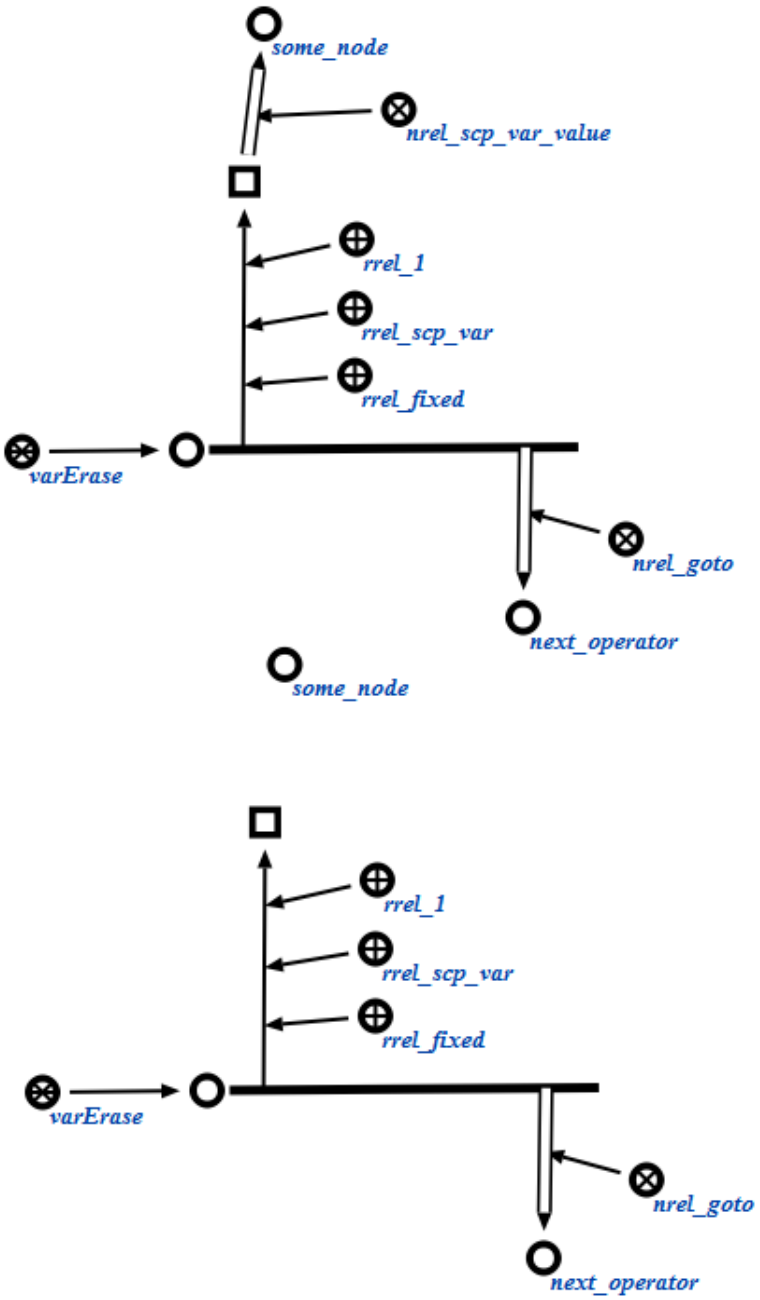



Рис. 3.74 Оператор удаления значения scp-переменной (Пример 1)

```
scp_operator_example_varErase (*
        <- varErase;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        => nrel_goto: next_operator;;
*);;
```

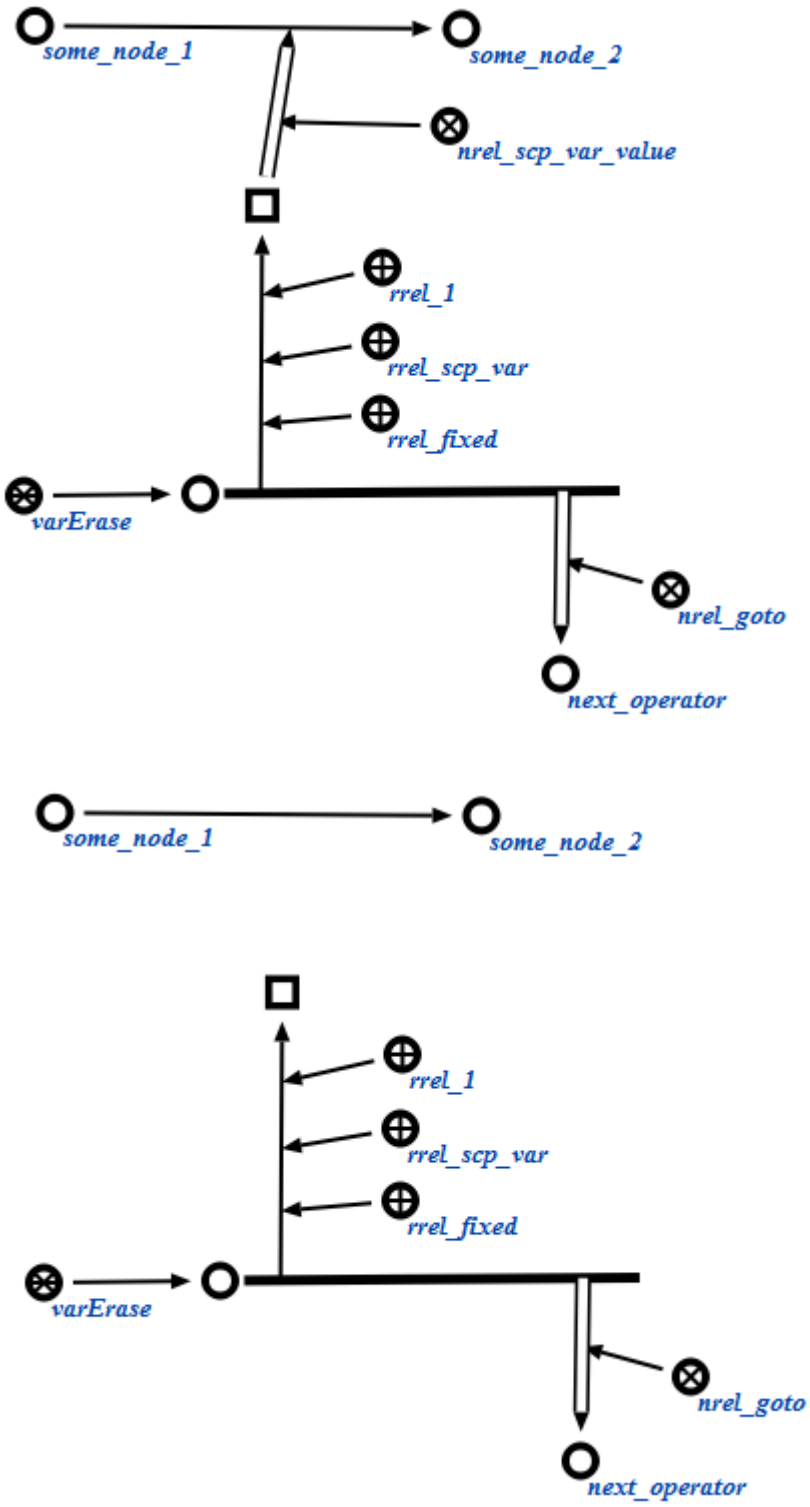



Рис. 3.75 Оператор удаления значения scp-переменной (Пример 2)

### 3.9 Scp-операторы вывода отладочной информации

Операторы класса ***scp-оператор вывода отладочной информации*** применяются при ручной отладке *scp-программ* в случае отсутствия необходимых для этого средств в используемой среде программирования. Также Операторы класса могут использованы для ведения системного протокола, хранимого вне *sc-памяти* на внешнем языке для выявления некоторых системных ошибок.

Для работы операторов данного класса используются стандартные средства операционной системы, обеспечивающие вывод текстовой информации в консоль.

Данный класс *scp-операторов* разбивается на следующие подклассы:

- ***scp-оператор вывода содержимого sc-ссылки;***
- ***scp-оператор вывода содержимого sc-ссылки с переходом на новую строку;***
- ***scp-оператор распечатки семантической окрестности sc-элемента.***

Операторы класса ***scp-оператор вывода содержимого sc-ссылки*** описывают действие вывода в консоль содержимого заданной *sc-ссылки*.

Каждый оператор данного класса содержит один операнд, который должен быть помечен как *scp-операнд с заданным значением'*. Значением данного операнда является *sc-ссылка*, содержимое которой должно быть сформировано и будет выведено в консоль. Особенности вывода данных определенного формата определяются возможностями платформы реализации.

```
scp_operator_example_print (*
        <- print;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        => nrel_goto: next_operator;;
*);;
scp_operator_example2_print (*
        <- print;;
        -> rrel_1: rrel_fixed: rrel_scp_const: [Hello, World!];;
        => nrel_goto: next_operator;;
*);;
```

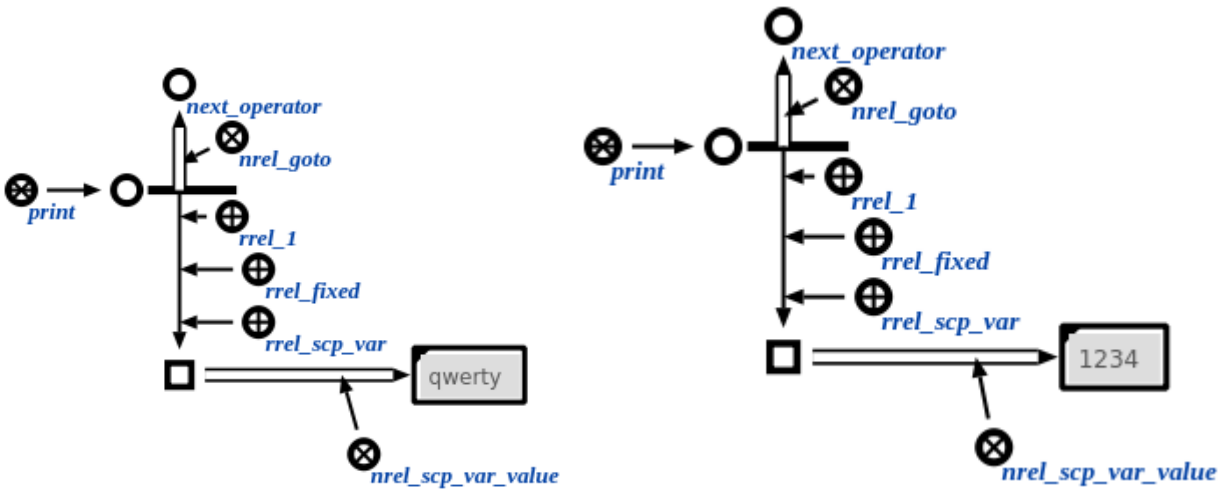



Рис. 3.76 Оператор вывода содержимого sc-ссылки

Действия, описываемые операторами класса ***scp-оператор вывода содержимого sc-ссылки с переходом на новую строку*** полностью аналогичны действиям, выполняемым *scp-операторами вывода содержимого sc-ссылки*, но после вывода в консоль содержимого *sc-ссылки* осуществляется переход на новую строку.

```
scp_operator_example_printNl (*
        <- printNl;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        => nrel_goto: next_operator;;
*);;
```

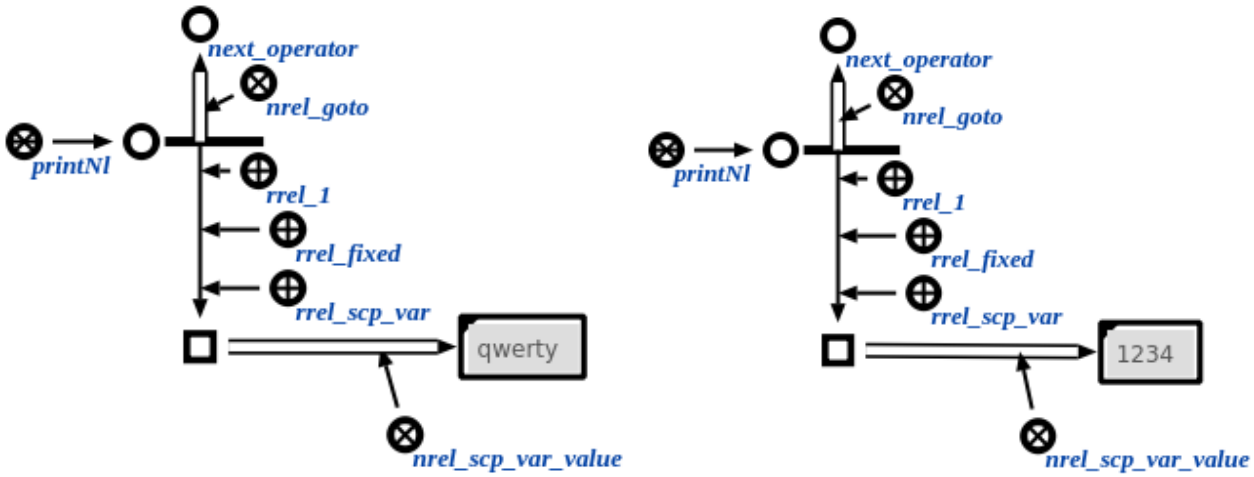



Рис. 3.77 Оператор вывода содержимого sc-ссылки с переходом на новую строку

Операторы класса ***scp-оператор распечатки семантической окрестности sc-элемента*** описывают действие вывода в консоль семантической окрестности единичного радиуса для заданного *sc-элемента*.

Для вывода информации используется формат, использующий примитивы, аналогичные обозначениям *sc-коннекторов* в *SCs-коде*.

Каждый оператор данного класса содержит один операнд, который должен быть помечен как *scp-операнд с заданным значением'*.

```
scp_operator_example_printEl (*
        <- printEl;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        => nrel_goto: next_operator;;
*);;
```

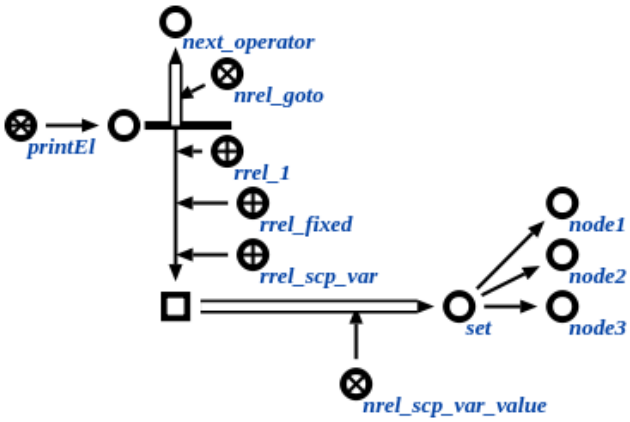



Рис. 3.78 Оператор распечатки семантической окрестности sc-элемента

### 3.9 Scp-операторы работы со строками

Операторы класса **scp-оператор конкатенации строкового содержимого sc-ссылок** описывают действие конкатенации строкового содержимого двух sc-ссылок, являющихся значениями второго и третьего операндов.

Каждый оператор данного класса содержит три операнда.

Первый операнд может иметь произвольный тип значения scp-операнда'. Значением данного операнда является sc-ссылка, идентифицирующая строку, являющейся результатом конкатенации строкового содержимого второго и третьего операнда. В случае, если данный операнд помечен как scp-операнд со свободным значением', то sc-ссылка будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся sc-ссылки.

Второй операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, к которой присоединяется содержимое третьего операнда.

Третий операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, которая присоединяется ко второму операнду.

```
scp_operator_example_stringConcat (*
        <- contStringConcat;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [Hello, ];;
        -> rrel_3: rrel_fixed: rrel_scp_const: [World!!!];;
```

```
        => nrel_goto: next_operator;;
*);;
```

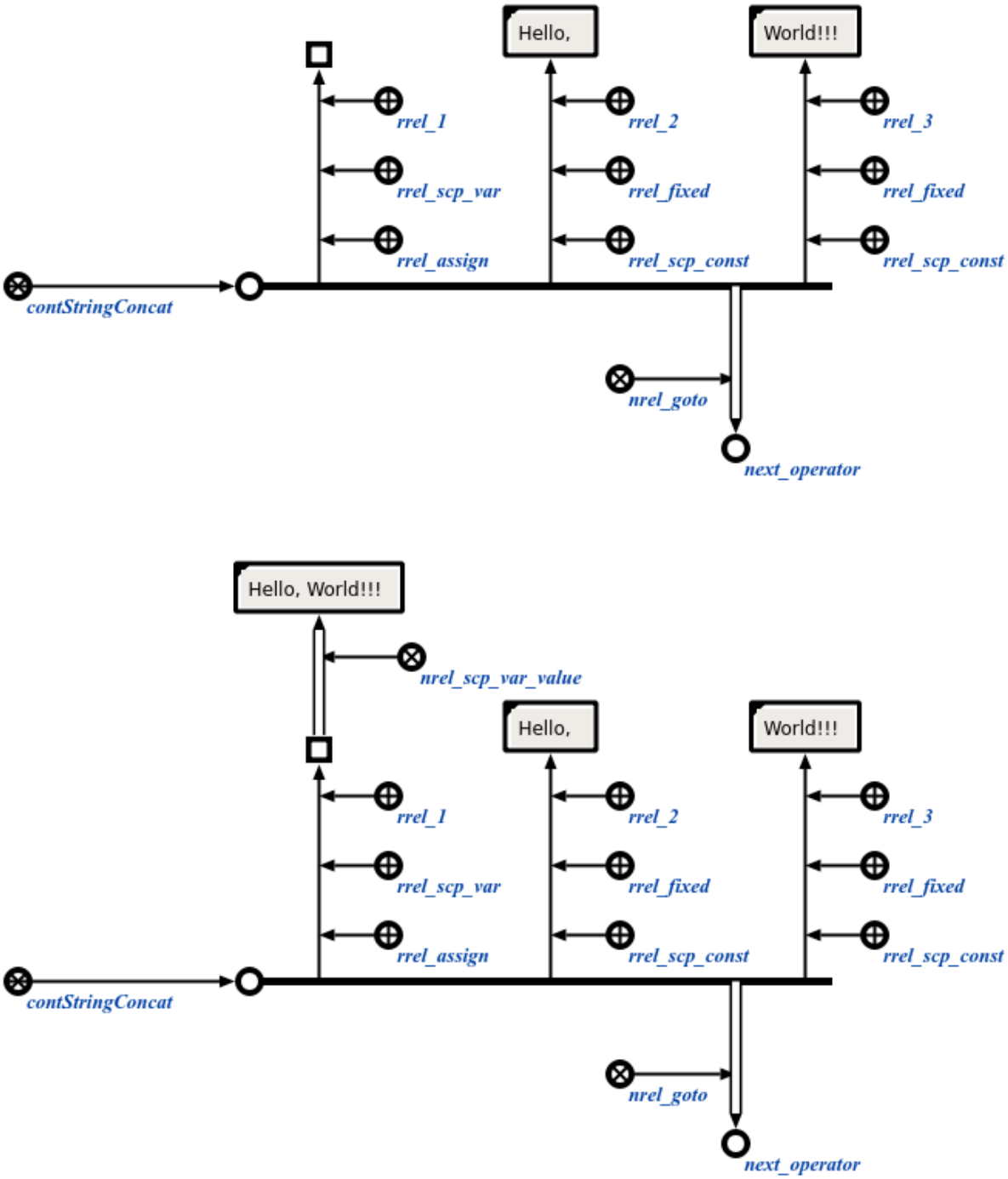



Рис. 3.79 Оператор конкатенации строк

Операторы класса **scp-оператор проверки совпадения конца строкового содержимого sc-ссылки со строковом содержимым другой sc-ссылки** описывают операции проверки на совпадение конечной части содержимого sc-ссылки, являющейся значением первого операнда, с содержимым sc-ссылки, являющейся значением второго операнда.

Каждый оператор данного класса содержит два операнда.

Первый операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, в которой осуществляется поиск.

Второй операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, поиск которой осуществляется в конце содержимого первого операнда.

```
scp_operator_example_stringEndsWith (*
        <- strinEndsWith;;
        -> rrel_1: rrel_fixed: rrel_scp_const: [test string];;
        -> rrel_2: rrel_fixed: rrel_scp_const: [ring];;
        => nrel_then: then_operator;;
        => nrel_else: else_operator;;
*);;
```

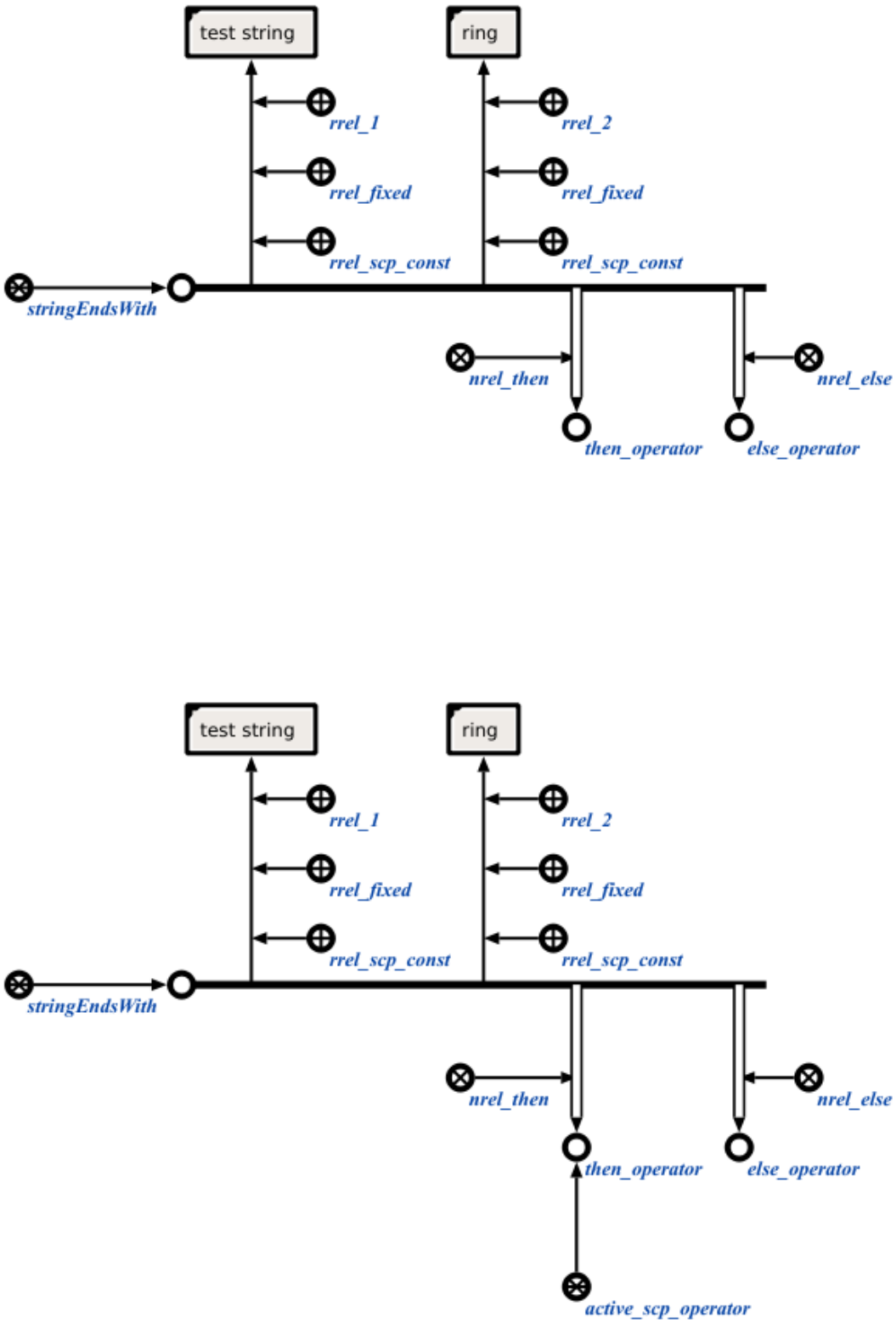



Рис. 3.80 Оператор проверки совпадения конца строкового содержимого sc-ссылки со строковом содержимым другой sc-ссылки

Операторы класса **scp-оператор проверки равенства строковых содержимых sc-ссылок** описывают действие проверки того, равны ли между собой строковые содержимые двух sc-ссылок, являющихся значениями операндов. Успешным выполнение считается, если строковые содержимые

равны. Каждый оператор данного класса содержит два операнда, которые должны быть помечены как scp-операнд с заданным значением'. Каждый из операндов может быть как scp-константой', так и scp-переменной'.

```
scp_operator_example_stringIfEq (*
        <- stringIfEq;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_var: _el1;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```

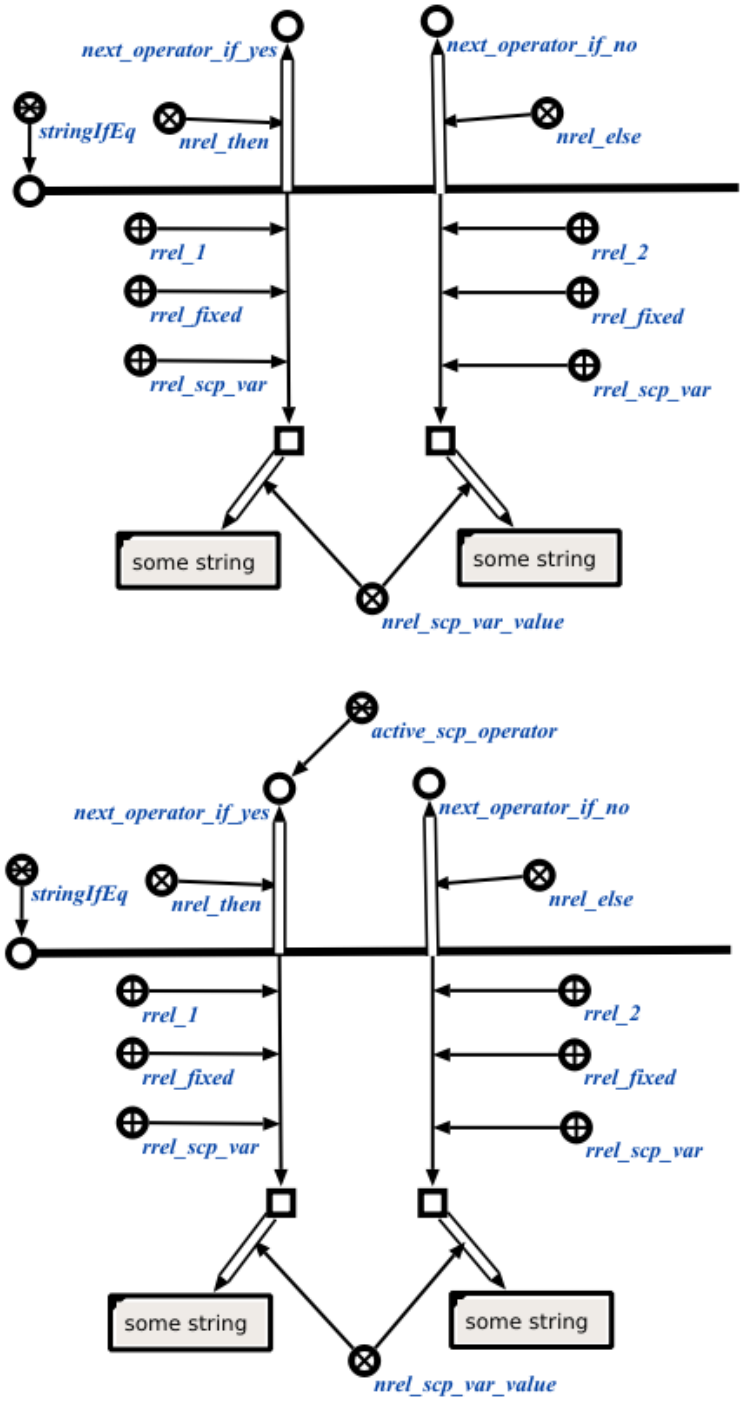



Рис. 3.81 Оператор проверки равенства строковых содержимых sc-ссылок

Операторы класса **scp-оператор лексикографического сравнения строковых содержимых sc-ссылок** описывают действие сравнения между собой строковых содержимых двух sc-ссылок, являющихся значениями операндов. Успешным выполнение считается, значение первого операнда строго больше значения второго операнда с точки зрения лексикографического порядка. Каждый оператор данного класса содержит два операнда, которые должны быть помечены как scp-операнд с заданным значением'. Каждый из операндов может быть как scp-константой', так и scp-переменной'.

```
scp_operator_example_stringIfGr (*
        <- stringIfGr;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_var: _el1;;
        => nrel_then: next_operator_if_yes;;
        => nrel_else: next_operator_if_no;;
*);;
```

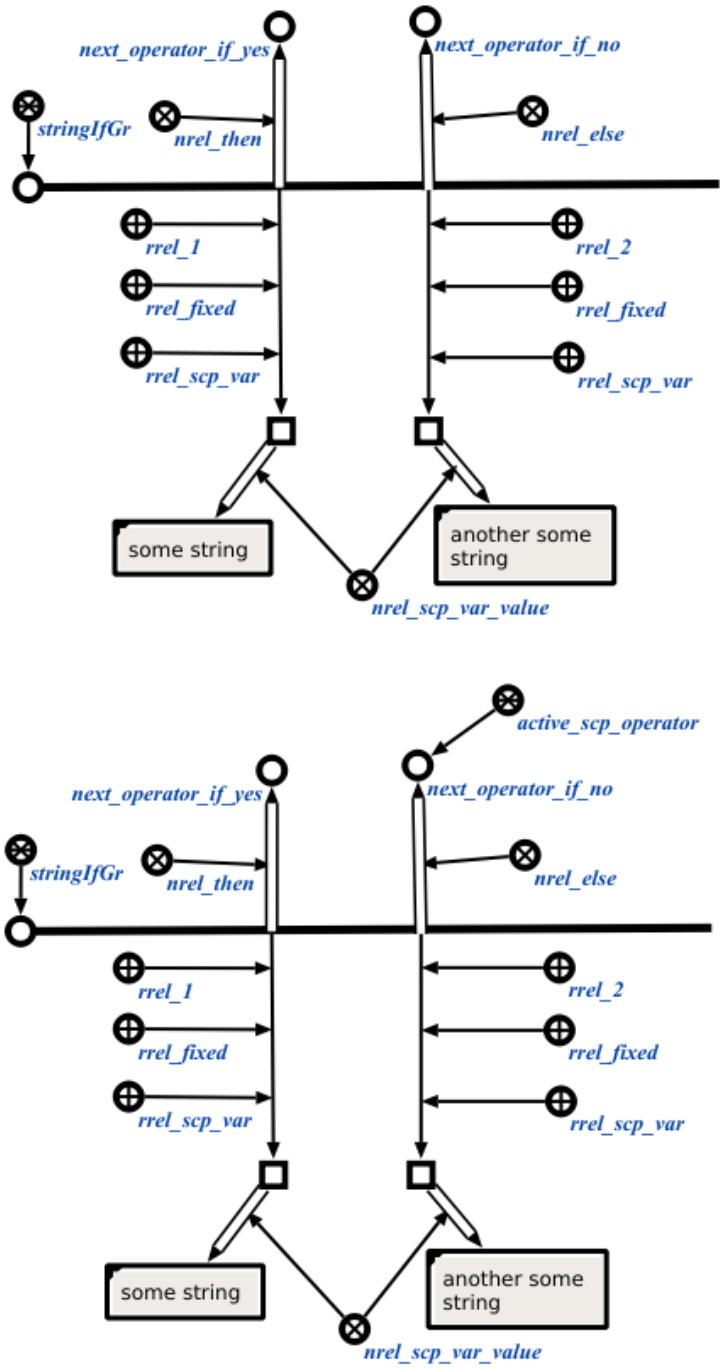



Рис. 3.82 Оператор лексикографического сравнения строковых содержимых sc-ссылок

Операторы класса **scp-оператор вычисления длины строкового содержимого sc-ссылки** описывают действие вычисления длины строкового содержимого sc-ссылки, являющейся значением второго операнда.

Каждый оператор данного класса содержит два операнда.

Первый операнд может иметь произвольный тип значения scp-операнда'. Значением данного операнда является sc-ссылка, идентифицирующая число, являющееся длиной строкового содержимого, задаваемого вторым операндом. В случае, если данный операнд помечен как scp-операнд со свободным значением', то sc-ссылка будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся sc-ссылки.

Второй операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, длина которой вычисляется.

```
scp_operator_example_stringLen (*
        <- stringLen;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [string];;
        => nrel_goto: .proc_operator_goto;;
*);;
```

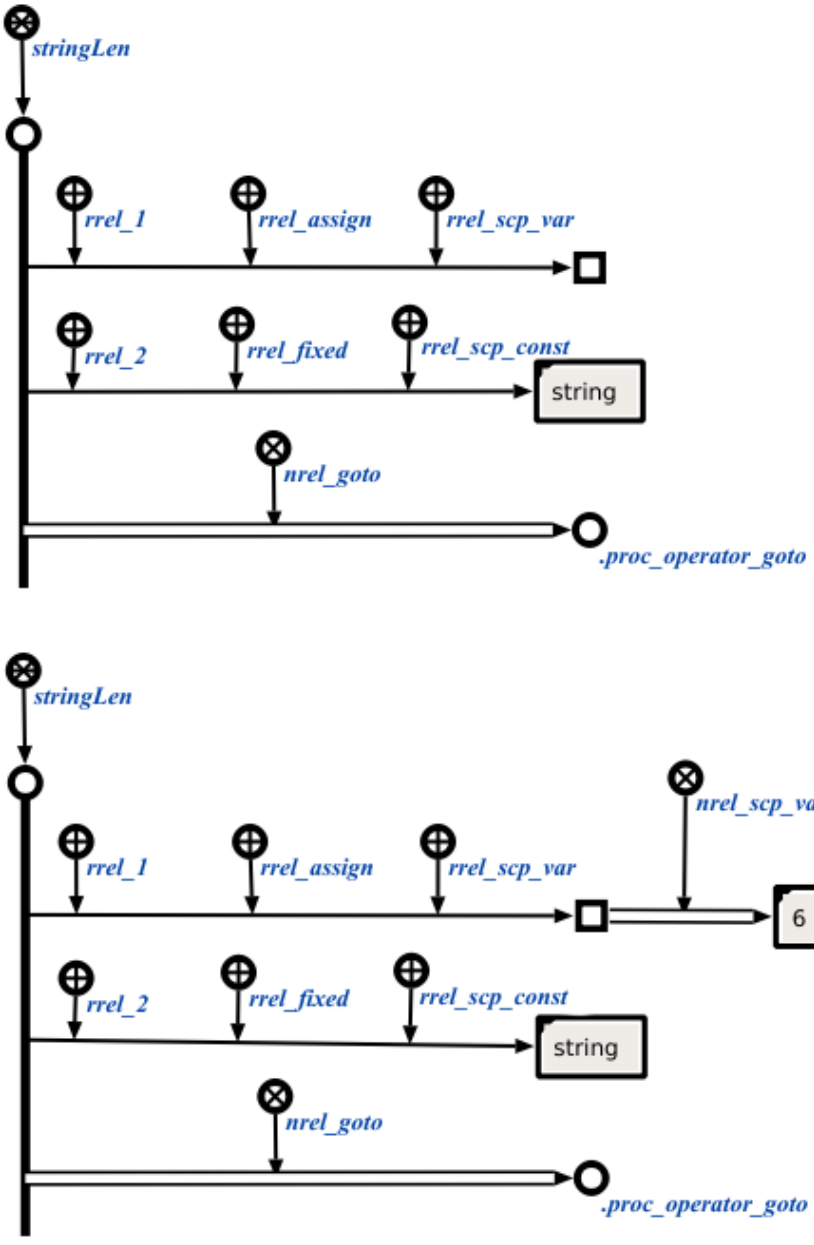



Рис. 3.83 Оператор вычисления длины строкового содержимого sc-ссылки

Операторы класса **scp-оператор замены определенной части строкового содержимого sc-ссылки на содержимое указанной sc-ссылки** описывают действие поиска строкового содержимого sc-ссылки, являющейся значением третьего операнда, внутри содержимого sc-ссылки, являющейся значением

второго операнда, и замены найденной подстроки строковым содержимым sc-ссылки, являющейся значением четвертого операнда.

Каждый оператор данного класса содержит четыре операнда.

Первый операнд может иметь произвольный тип значения scp-операнда'. Значением данного операнда является sc-ссылка, идентифицирующая строку, являющейся результатом поиска и замены найденной подстроки указанной строкой, значения которых определены стркоовым содержимым третьего и четвертого операнда, в строковом содержимом второго операнда. В случае, если данный операнд помечен как scp-операнд со свободным значением', то sc-ссылка будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся sc-ссылки.

Второй операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, в которой осуществляется поиск и замены.

Третий операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, вхождения которой в содержимое второго операнда требуется заменить.

Четвертый операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, на которую требуется заменить вхождения содержимого третьего операнда в содержимое второго.

```
scp_operator_example_stringReplace (*
        <- stringReplace;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [test example];;
        -> rrel_3: rrel_fixed: rrel_scp_const: [example];;
        -> rrel_4: rrel_fixed: rrel_scp_const: [string];;
        => nrel_goto: next_operator;;
*);;
```

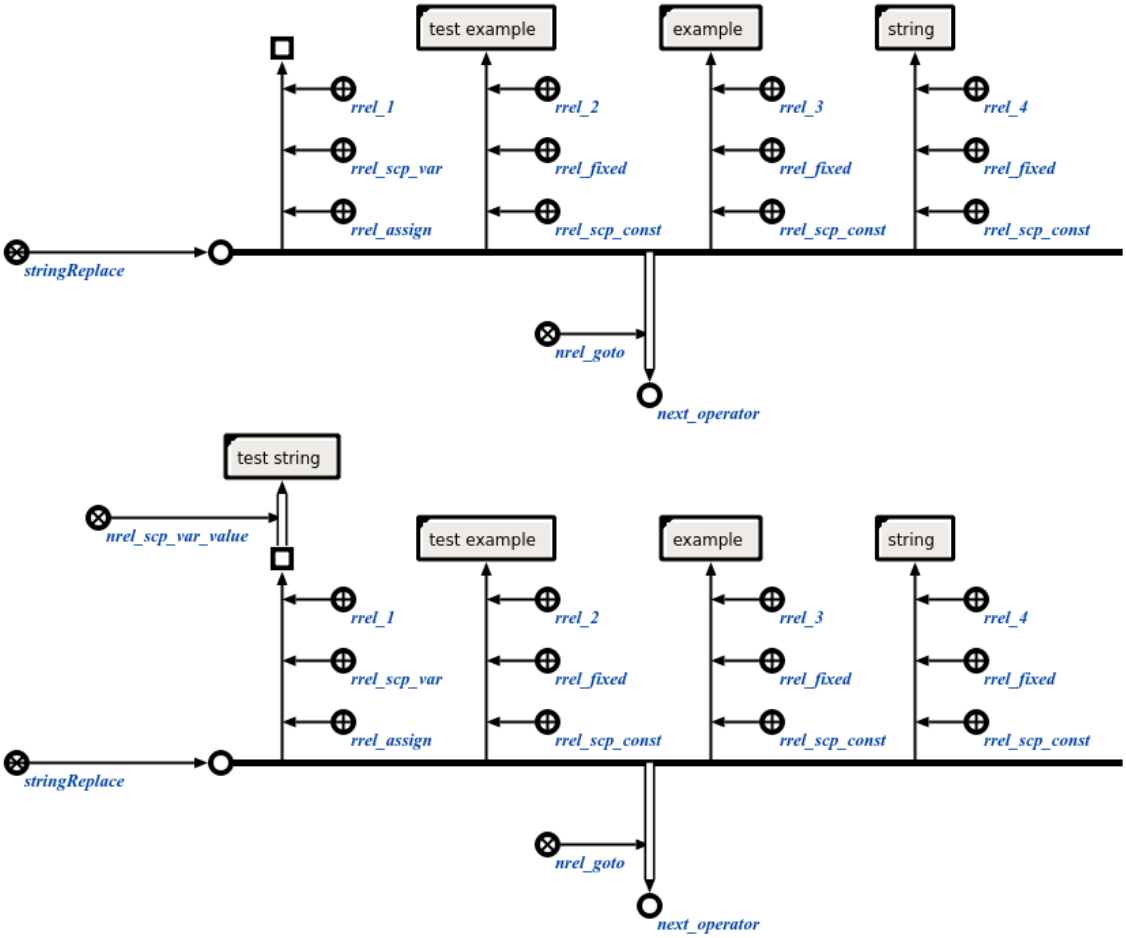



Рис. 3.84 Оператор замены определенной части строкового содержимого sc-ссылки на содержимое указанной sc-ссылки

Операторы класса **scp-оператор получения части строкового содержимого sc-ссылки по индексам** описывают действие получения части строкового содержимого sc-ссылки, являющейся значением второго операнда, по двум значениям числового содержимого двух sc-ссылок, являющихся значениями третьего и четвертого операнда.

Каждый оператор данного класса содержит четыре операнда.

Первый операнд может иметь произвольный тип значения scp-операнда'. Значением данного операнда является sc-ссылка, идентифицирующая строку, часть строкового содержимого второго операнда, ограниченную индексами. В случае, если данный операнд помечен как scp-операнд со свободным значением', то sc-ссылка будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся sc-ссылки.

Второй операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, часть которой требуется получить.

Третий операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая число, являющееся индексом - левой границей запрашиваемой подстроки.

Четвертый операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая число, являющееся индексом - правой границей запрашиваемой подстроки.

```
scp_operator_example_stringSlice (*
        <- stringSlice;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [test example];;
        -> rrel_3: rrel_fixed: rrel_scp_const: [0];;
        -> rrel_4: rrel_fixed: rrel_scp_const: [3];;
        => nrel_goto: next_operator;;
*);;
```

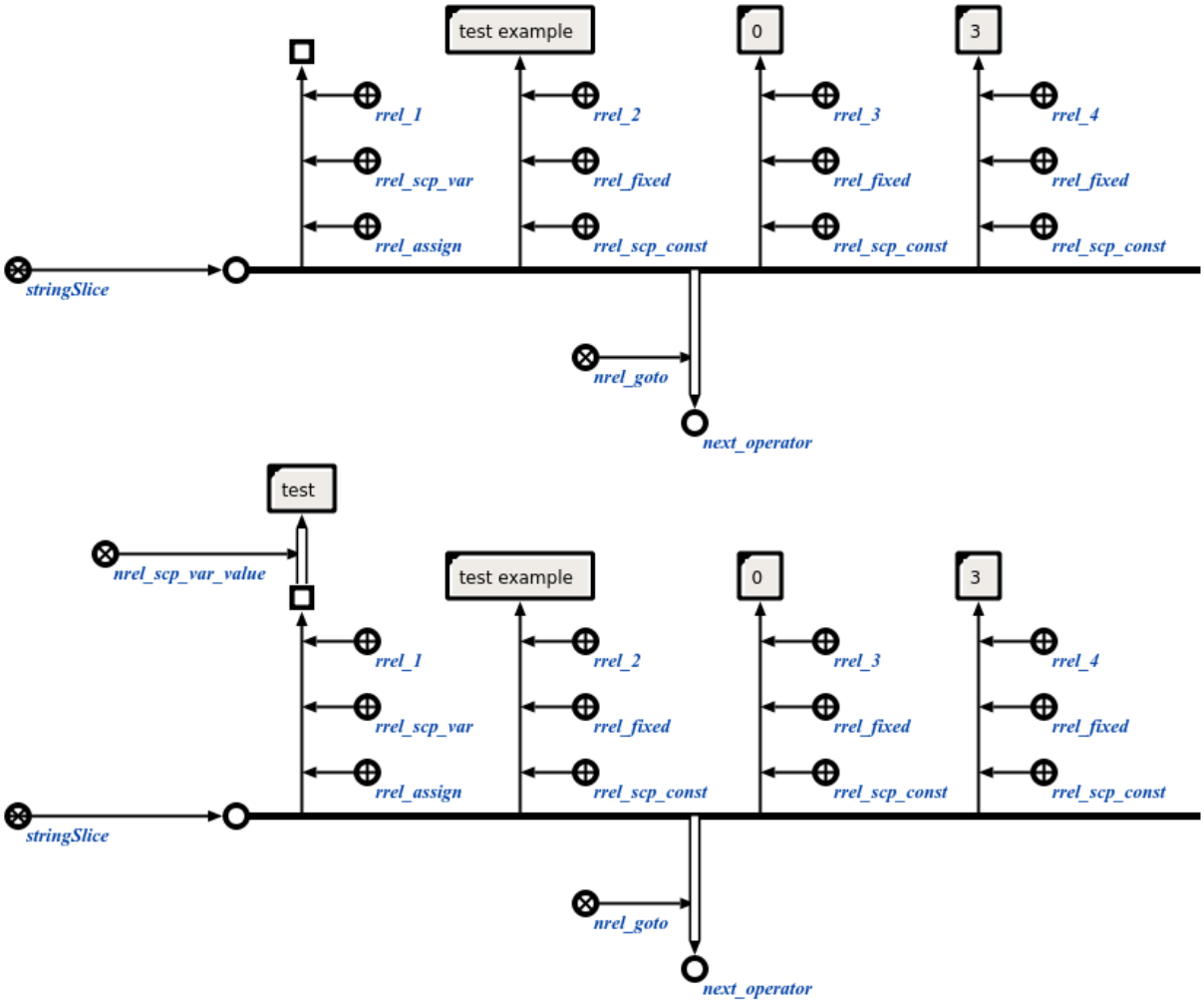



Рис. 3.85 Оператор получения части строкового содержимого sc-ссылки по индексам

Операторы класса **scp-оператор разбиения строки на подстроки** описывают действие разбиения строкового содержимого sc-элемента на подстроки, разделенные разделителем.

Каждый оператор данного класса содержит три операнда.

Первый операнд может иметь произвольный тип значения scp-операнда'. Значением данного операнда является sc-ссылка, идентифицирующая множество подстрок строки, задаваемой вторым операндом. В случае, если данный операнд помечен как scp-операнд со свободным значением', то sc-ссылка будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся sc-ссылки.

Второй операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, которая разбивается на подстроки.

Третий операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, которая выступает как разделитель подстрок второго операнда.

```
scp_operator_example_stringSplit (*
        <- stringSplit;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [some_test_example];;
        -> rrel_3: rrel_fixed: rrel_scp_const: [_];;
        => nrel_goto: next_operator;;
*);;
```

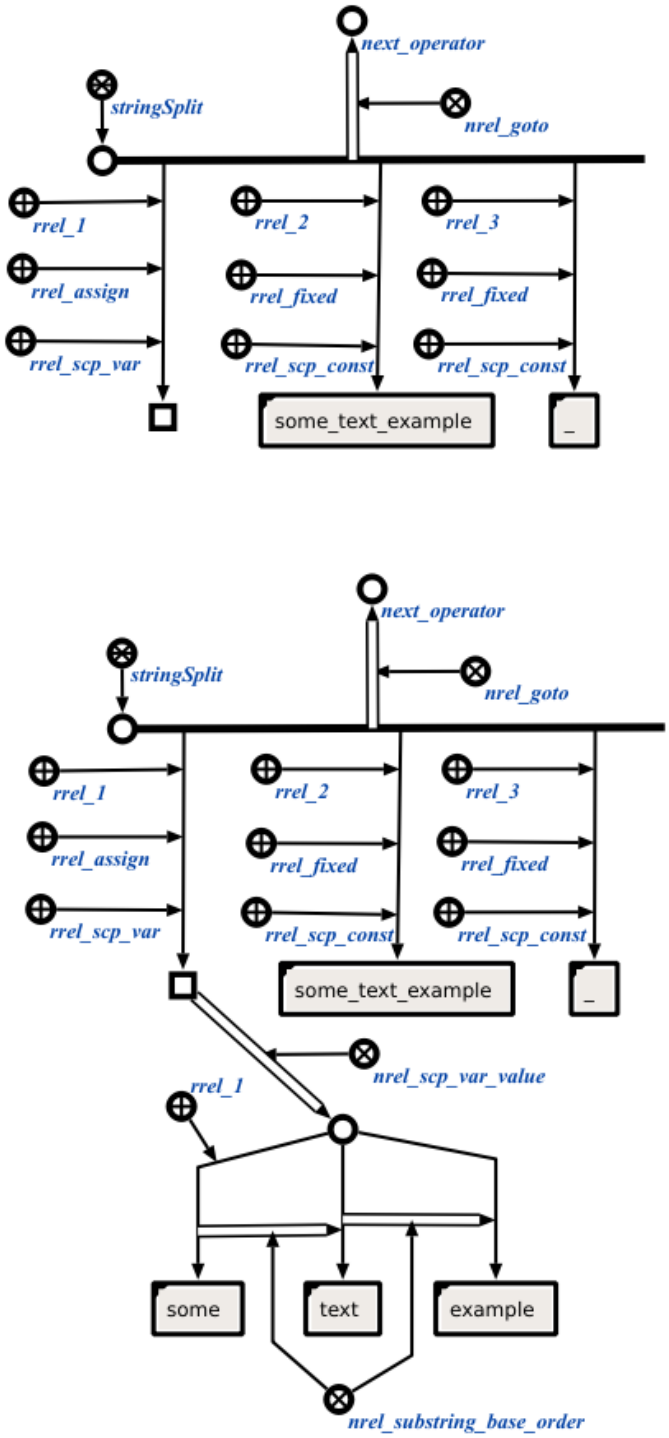



Рис. 3.86 Оператор разбиения строки на подстроки

Операторы класса **scp-оператор проверки совпадения начальной части строкового содержимого sc-ссылки со строковом содержимым другой sc-ссылки** описывают операции проверки на совпадение начальной части содержимого sc-ссылки, являющейся значением первого операнда, с содержимым sc-ссылки, являющейся значением второго операнда.

Каждый оператор данного класса содержит два операнда.

Первый операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, в которой осуществляется поиск.

Второй операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, поиск которой осуществляется в начале содержимого первого операнда.

```
scp_operator_example_stringStartsWith (*
        <- stringStartsWith;;
        -> rrel_1: rrel_fixed: rrel_scp_const: [test string];;
        -> rrel_2: rrel_fixed: rrel_scp_const: [test];;
        => nrel_then: then_operator;;
        => nrel_else: else_operator;;
*);;
```

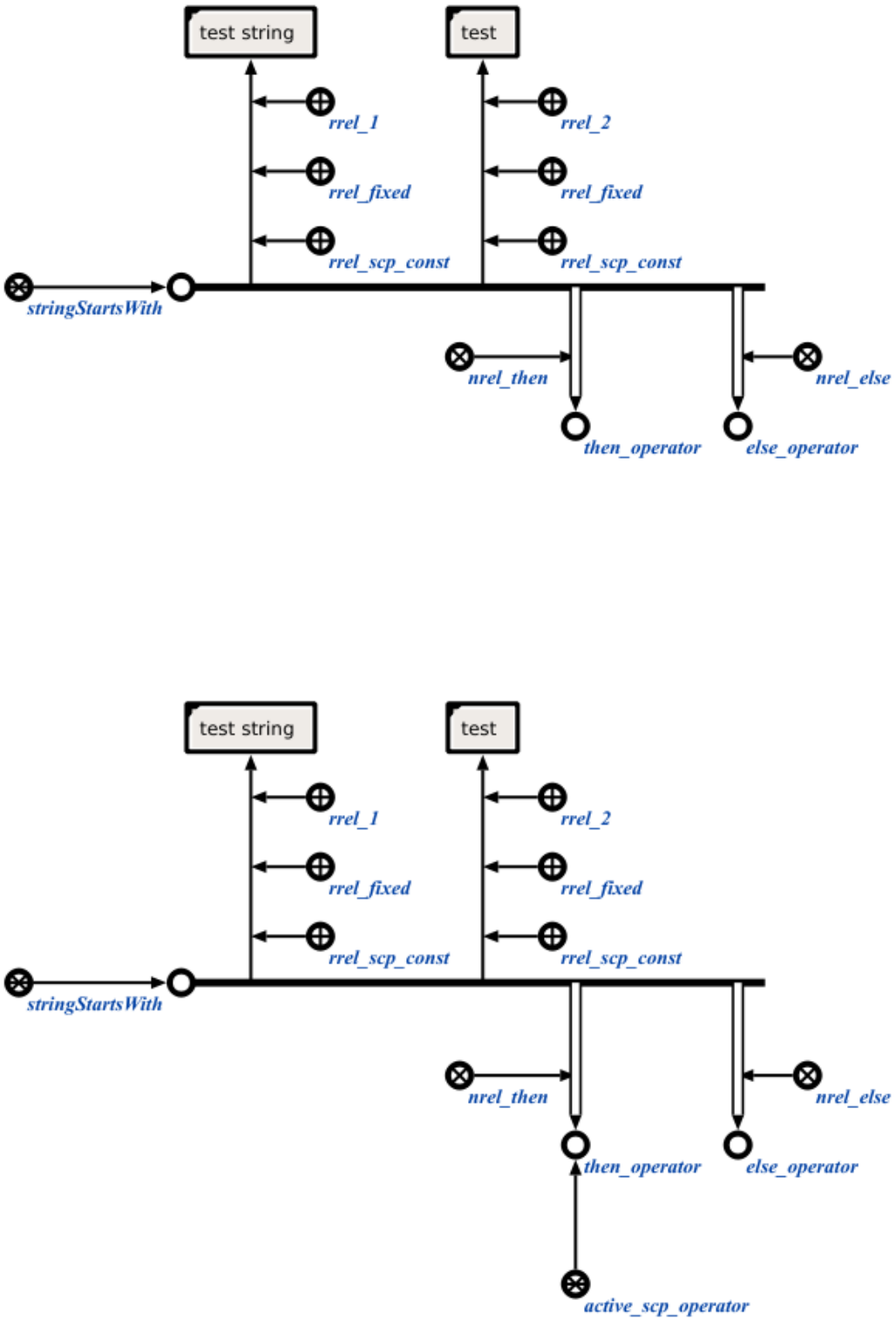



Рис. 3.87 Оператор проверки совпадения начальной части строкового содержимого sc-ссылки со строковом содержимым другой sc-ссылки

Операторы класса **scp-оператор поиска строкового содержимого sc-ссылки в строковом содержимом другой sc-ссылки** описывают операции поиска содержимого sc-ссылки, являющейся значением третьего операнда, в содержимом sc-ссылки, являющейся значением второго операнда.

Каждый оператор данного класса содержит три операнда.

Первый операнд может иметь произвольный тип значения scp-операнда'. Значением данного операнда является sc-ссылка, идентифицирующая число, являющееся позицией строкового содержимого, задаваемого третьим операндом, в строковом содержимом, задаваемым вторым операндом. В случае, если данный операнд помечен как scp-операнд со свободным значением', то sc-ссылка будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся sc-ссылки.

Второй операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, в которой осуществляется поиск подстроки.

Третий операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, поиск которой осуществляется в содержимом второго операнда.

```
scp_operator_example_stringSub (*
        <- stringSub;;
        -> rrel_1: rrel_assign: rrel_scp_var: _el1;;
        -> rrel_2: rrel_fixed: rrel_scp_const: [test string];;
        -> rrel_3: rrel_fixed: rrel_scp_const: [str];;
        => nrel_goto: next_operator;;
*);;
```

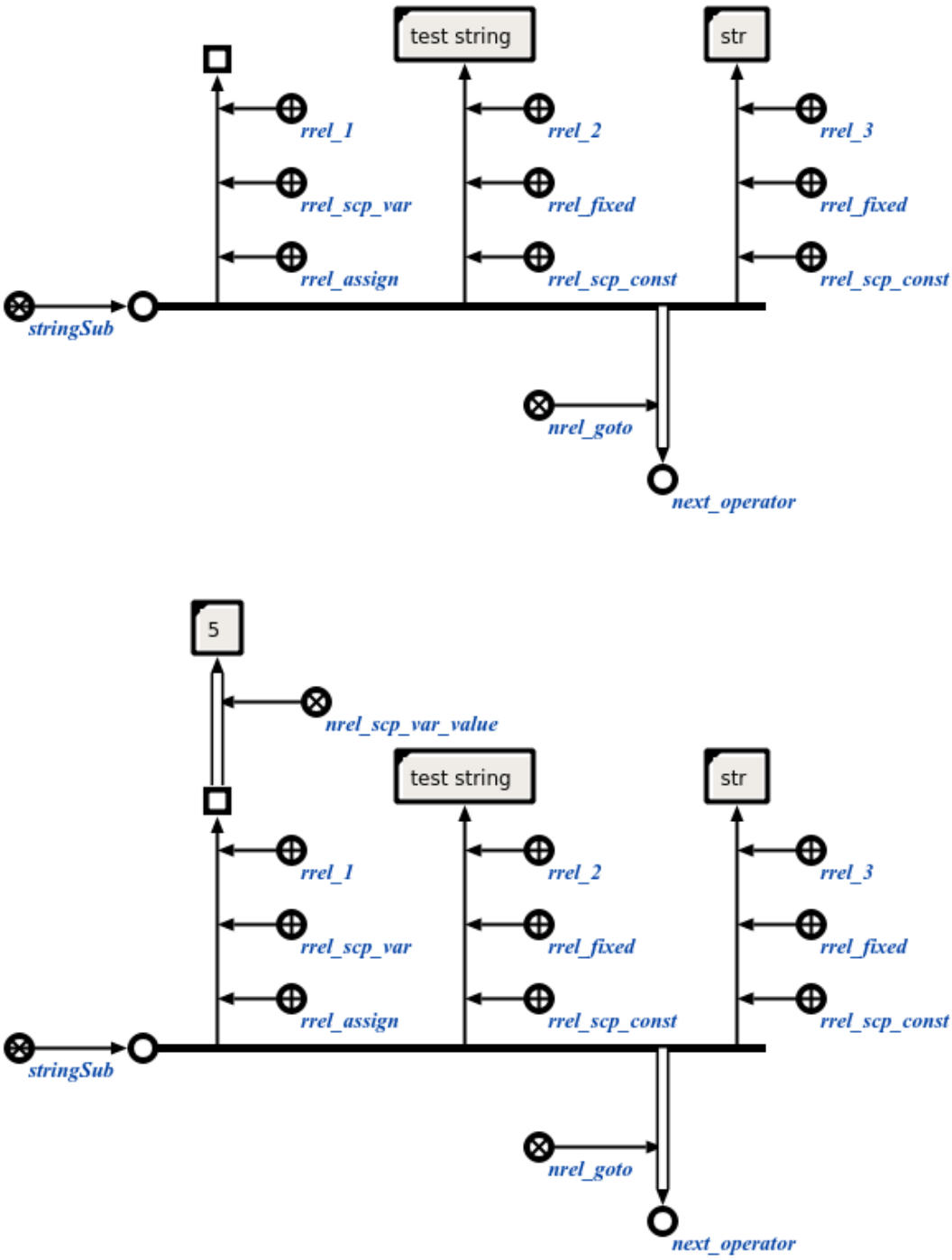



Рис. 3.88 Оператор поиска строкового содержимого sc-ссылки в строковом содержимом другой sc-ссылки

Операторы класса **scp-оператор перевода в нижний регистр строкового содержимого sc-ссылки** описывают действие перевода строкового содержимого в нижний регистр sc-ссылки, являющейся значением второго операнда.

Каждый оператор данного класса содержит два операнда.

Первый операнд может иметь произвольный тип значения scp-операнда'. Значением данного операнда является sc-ссылка, идентифицирующая строку, являющееся строкой в верхнем регистре, соответсвующей второму операнду. В

случае, если данный операнд помечен как scp-операнд со свободным значением', то sc-ссылка будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся sc-ссылки.

Второй операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, являющееся аргументом функции перевода строки в нижний регистр.

```
scp_operator_example_stringToLowerCase (*
        <- stringToLowerCase;;
        -> rrel_1: rrel_fixed: rrel_scp_var: _el1;;
        -> rrel_2: rrel_assign: rrel_scp_var: _el2;;
        => nrel_goto: next_operator;;
*);;
```

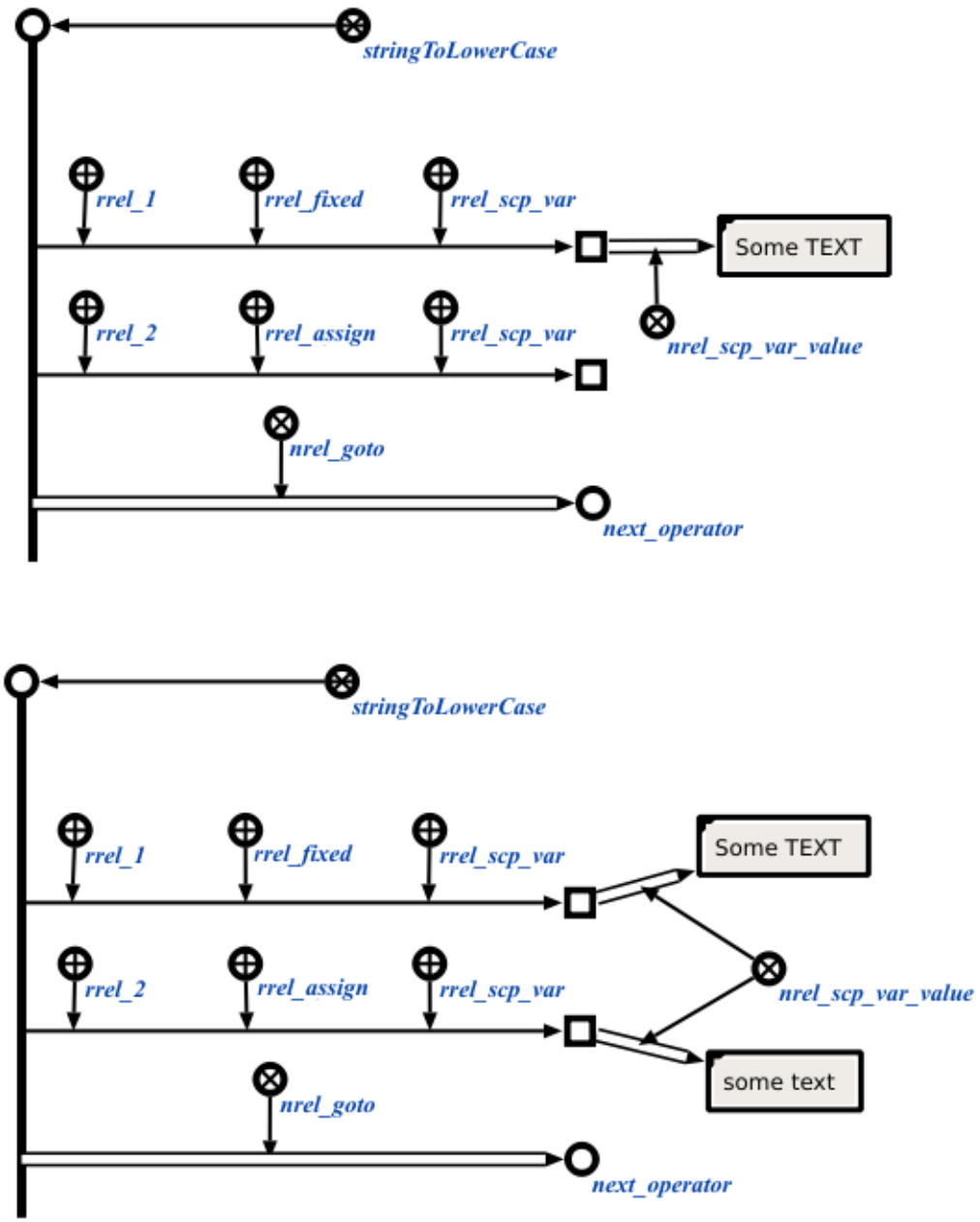



Рис. 3.89 Оператор перевода в нижний регистр строкового содержимого sc-ссылки

Операторы класса **scp-оператор перевода в верхний регистр строкового содержимого sc-ссылки** описывают действие перевода строкового содержимого в верхний регистр sc-ссылки, являющейся значением второго операнда.

Каждый оператор данного класса содержит два операнда.

Первый операнд может иметь произвольный тип значения scp-операнда'. Значением данного операнда является sc-ссылка, идентифицирующая строку, являющееся строкой в верхнем регистре, соответсвующей второму операнду. В случае, если данный операнд помечен как scp-операнд со свободным значением', то sc-ссылка будет сгенерирована, в противном случае будет изменено содержимое уже имеющейся sc-ссылки.

Второй операнд должен быть помечен как scp-операнд с заданным значением'. Значением данного операнда является sc-ссылка, идентифицирующая строку, являющееся аргументом функции перевода строки в верхний регистр.

```
scp_operator_example_stringToUpperCase (*
        <- stringToUpperCase;;
        -> rrel_1: rrel_fixed: rrel_scp_const: [Some TEXT];;
        -> rrel_2: rrel_assign: rrel_scp_var: _el2;;
        => nrel_goto: next_operator;;
*);;
```

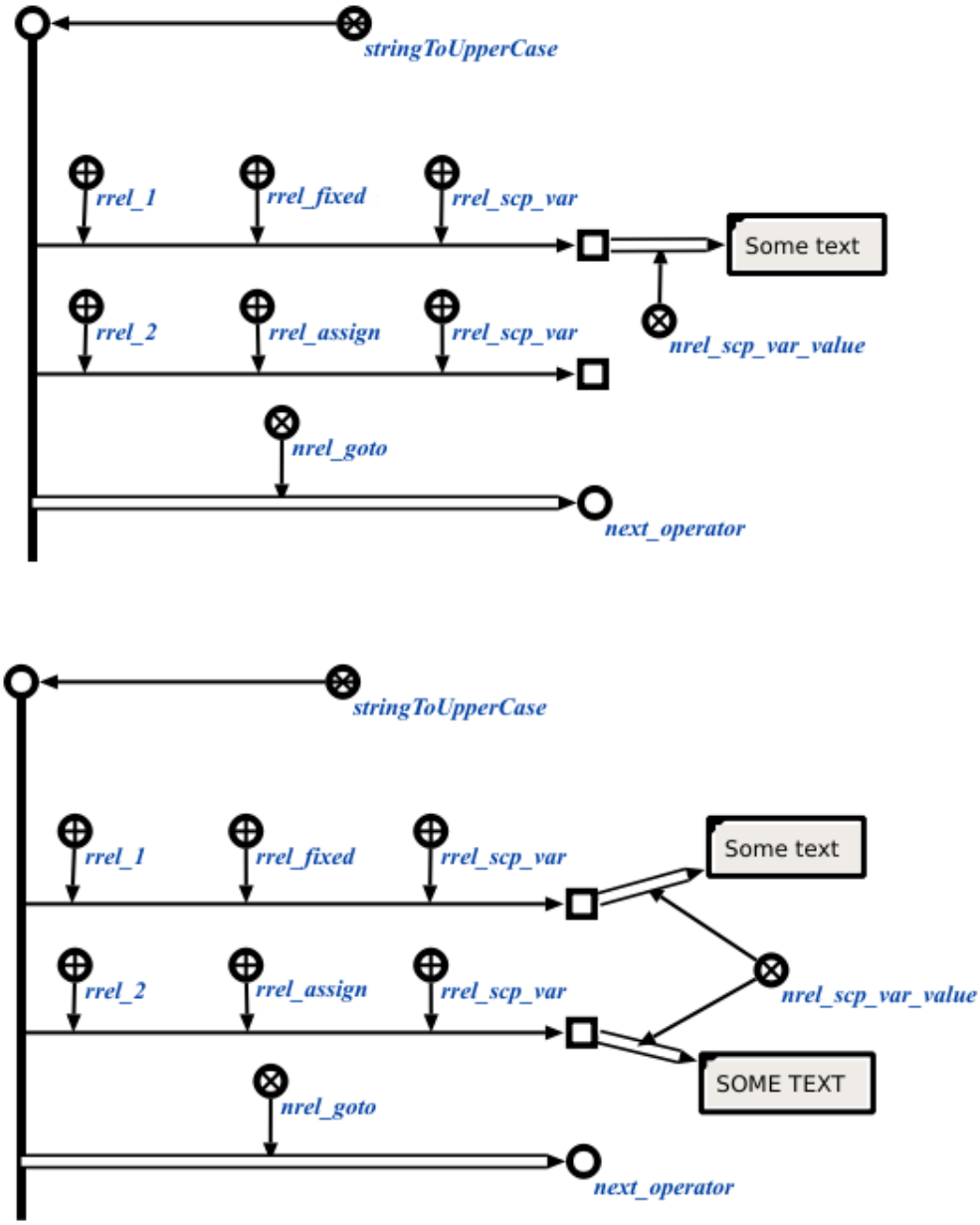



Рис. 3.90 Оператор перевода в верхний регистр строкового содержимого sc-ссылки

### 3.10 Scp-операторы синхронизации выполнения scp-программы

Операторы класса ***scp-оператор синхронизации выполнения scp-программы*** предназначены для синхронизации переходов между *scp-операторами* в рамках одной *scp-программы* и не описывают никаких непосредственных преобразований *sc-памяти*. Операторы данного класса могут использоваться, например, когда возникает необходимость дождаться завершения выполнения нескольких *scp-операторов*, выполнявшихся параллельно в рамках одной из веток *scp-программы*, порожденной каким-либо *scp-оператором проверки условий*.

```
scp_operator_example_synchronize (*
        <- syncronize;;
        => nrel_goto: next_operator;;
*);;
```

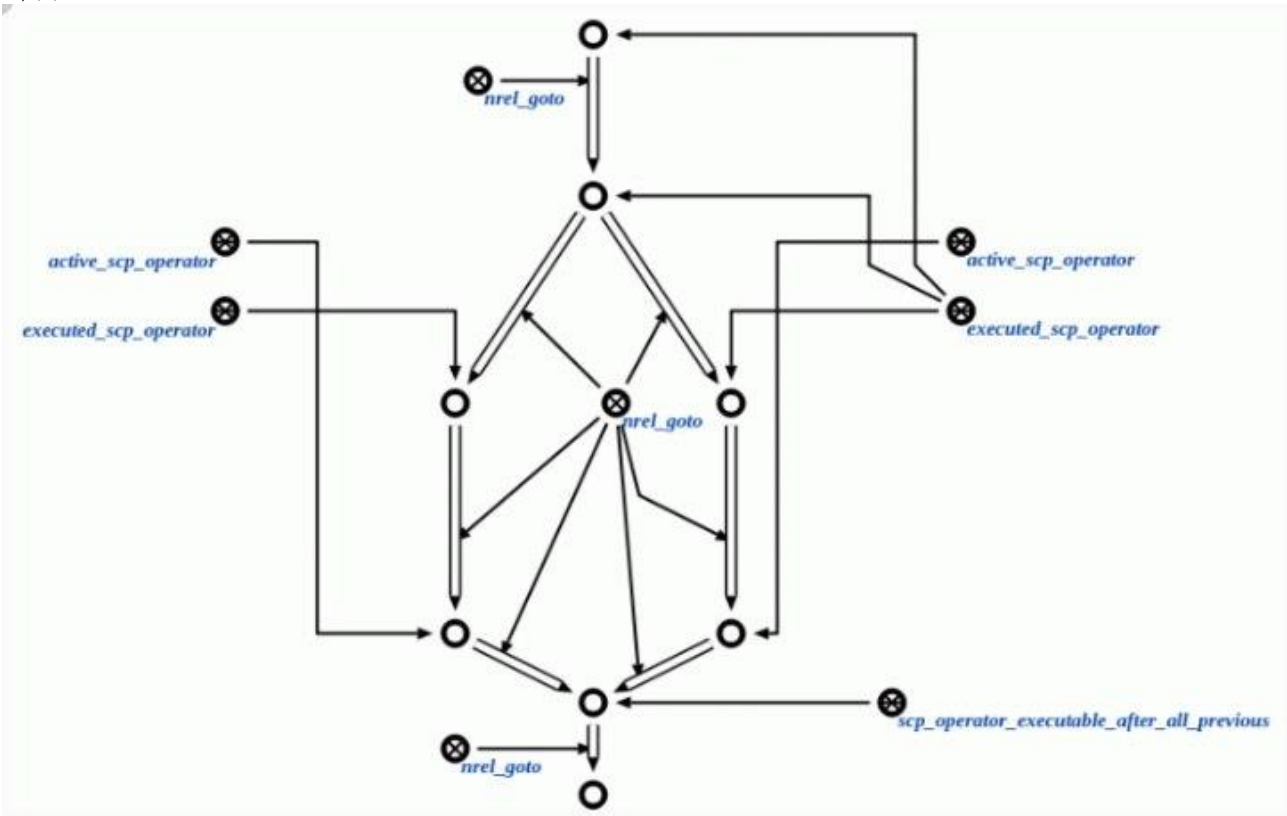



Рис. 3.91 Операторы класса scp-оператор синхронизации выполнения scp-программы, ч.1

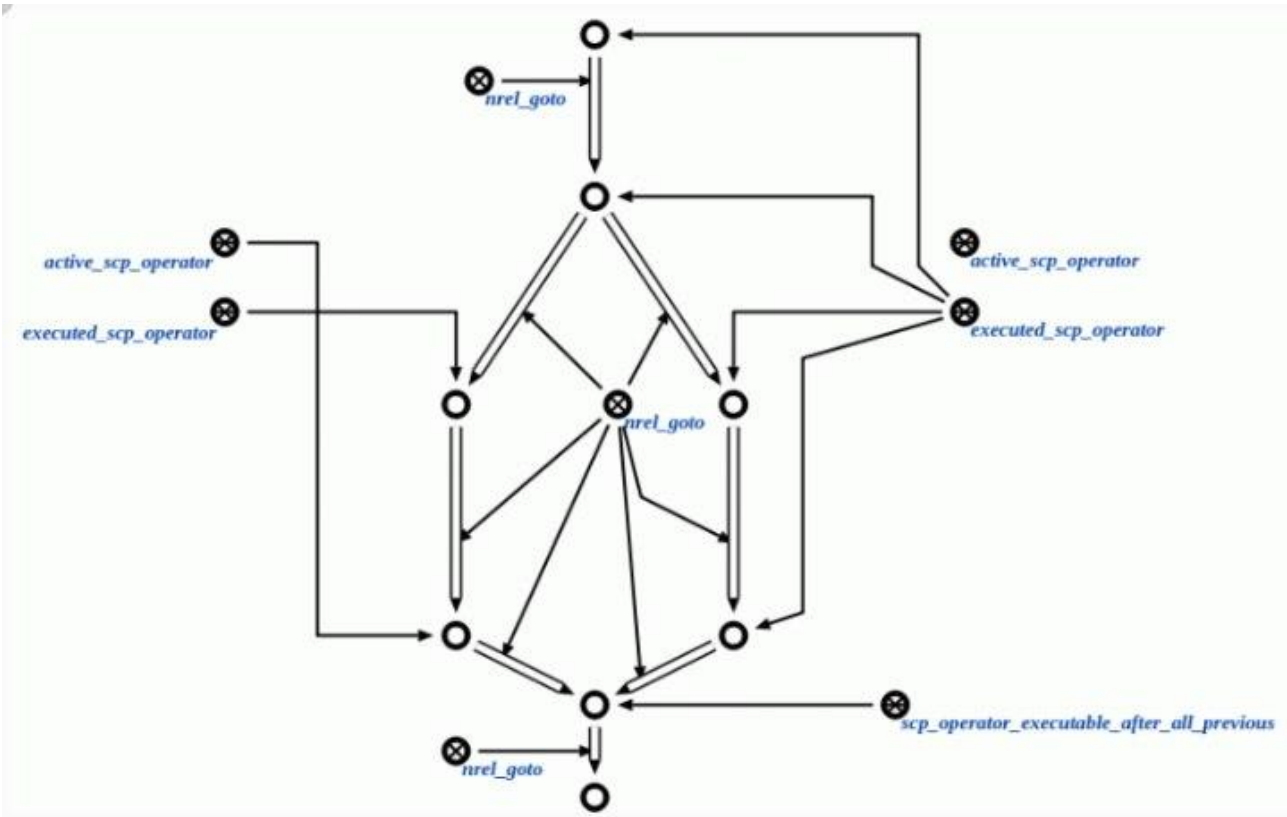


Рис. 3.92 Операторы класса scp-оператор синхронизации выполнения scp-программы, ч.2

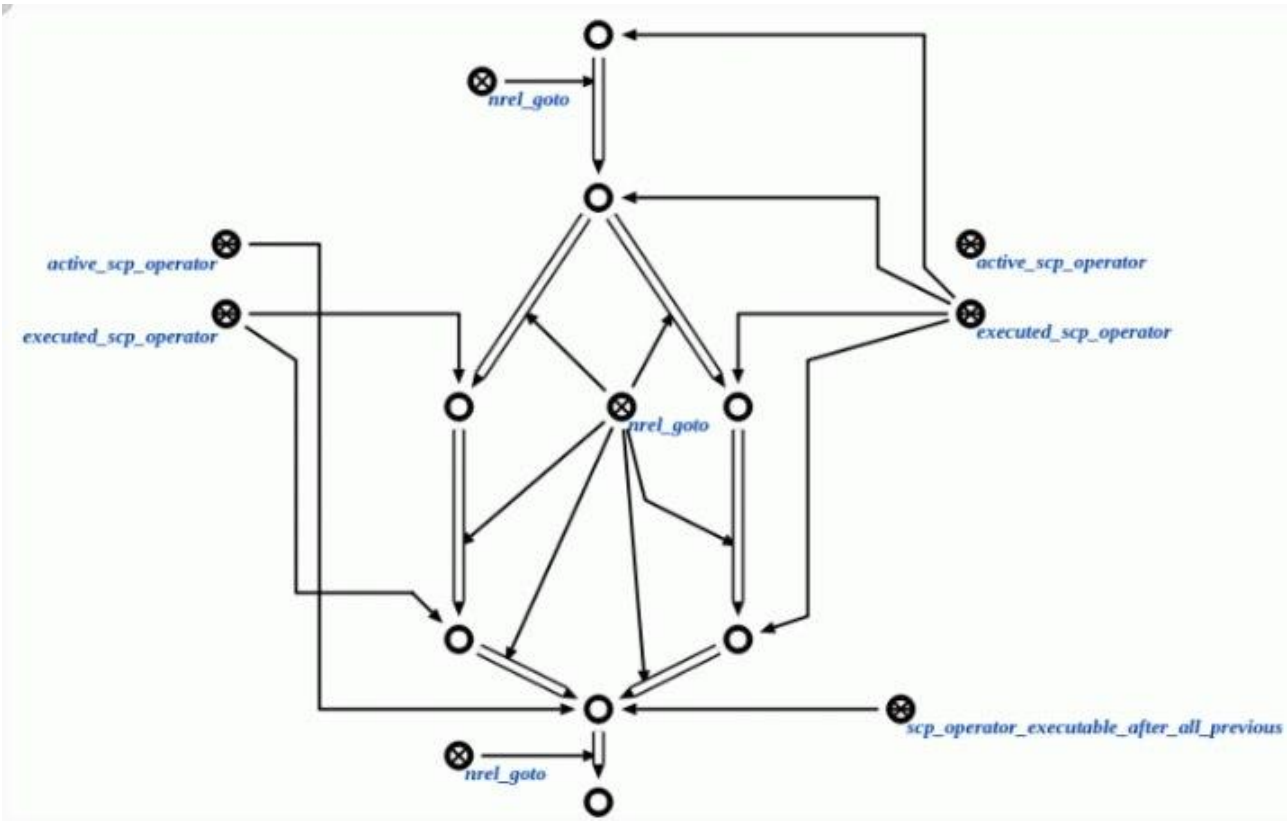



Рис. 3.93 Операторы класса scp-оператор синхронизации выполнения scp-программы, ч.3

# 4 РАЗРАБОТКА ПРОГРАММ ПРИ ПОМОЩИ СТАНДАРТНОГО РЕДАКТОРА

При программировании на *языке SCP* без использования специализированной среды, например, путем редактирования исходных текстов базы знаний в *SCs-коде* при помощи стандартного текстового редактора, необходимо пользоваться *системными идентификаторами** ключевых понятий языка, список которых приведен в таблице 2.

Таблица 2 – системные идентификаторы ключевых понятий языка SCP

| Основной русскоязычный идентификатор | Системный идентификатор |
|---|---|
| ***scp-программа*** | **scp_program** |
| ***scp-процедура*** | **scp_procedure** |
| ***агентная scp-программа*** | **agent_scp_program** |
| ***активная агентная scp-программа*** | **active_agent_scp_program** |
| ***параметры'*** | **rrel_params** |
| ***операторы'*** | **rrel_operators** |
| ***начальный оператор'*** | **rrel_init** |
| ***in-параметр'*** | **rrel_in** |
| ***out-параметр'*** | **rrel_out** |
| ***scp-константа'*** | **rrel_scp_const** |
| ***scp-переменная'*** | **rrel_scp_var** |
| ***значение переменной**** | **nrel_scp_var_value** |
| ***scp-операнд с заданным значением'*** | **rrel_fixed** |
| ***scp-операнд со свободным значением'*** | **rrel_assign** |
| ***константный sc-элемент'*** | **rrel_const** |
| ***переменный sc-элемент'*** | **rrel_var** |
| ***sc-узел'*** | **rrel_node** |
| ***sc-дуга'*** | **rrel_arc** |
| ***sc-ссылка'*** | **rrel_link** |
| ***sc-дуга общего вида'*** | **rrel_common** |
| ***sc-дуга принадлежности'*** | **rrel_access** |
| ***позитивная sc-дуга принадлежности'*** | **rrel_pos** |
| ***негативная sc-дуга принадлежности'*** | **rrel_neg** |
| ***нечеткая sc-дуга принадлежности'*** | **rrel_fuz** |

| | |
|---|---|
| ***временная sc-дуга принадлежности'*** | **rrel_temp** |
| ***постоянная sc-дуга принадлежности'*** | **rrel_perm** |
| ***sc-дуга основного вида'*** | **rrel_pos_const_perm** |
| ***формируемое множество'*** | **rrel_set** |
| ***формируемое множество 1', формируемое множество 2'…*** | **rrel_set_1, rrel_set_2…** |
| ***удаляемый sc-элемент'*** | **rrel_erase** |
| ***следующий оператор**** | **nrel_goto** |
| ***следующий оператор при успешном выполнении**** | **nrel_then** |
| ***следующий оператор при безуспешном выполнении**** | **nrel_else** |
| ***scp-оператор*** | **scp_operator** |
| ***1', 2', 3'…*** | **rrel_1, rrel_2, rrel_3 …** |
| ***scp-оператор, выполняемый после завершения хотя бы одного из предыдущих*** | **scp_operator_executable_after_one_of_previous** |
| ***scp-оператор, выполняемый после завершения всех предыдущих*** | **scp_operator_executable_after_all_previous** |
| ***scp-оператор генерации одноэлементной sc-конструкции*** | **genEl** |
| ***scp-оператор генерации трехэлементной sc-конструкции*** | **genElStr3** |
| ***scp-оператор генерации пятиэлементной sc-конструкции*** | **genElStr5** |
| ***scp-оператор генерации sc-конструкции по произвольному образцу*** | **sys_gen** |
| ***scp-оператор удаления одноэлементной sc-конструкции*** | **eraseEl** |
| ***scp-оператор удаления трехэлементной sc-конструкции*** | **eraseElStr3** |
| ***scp-оператор удаления пятиэлементной sc-конструкции*** | **eraseElStr5** |
| ***scp-оператор удаления множества*** | **eraseSetStr3** |

| | |
|---|---|
| *элементов трехэлементной sc-конструкции* | |
| *scp-оператор удаления множества элементов пятиэлементной sc-конструкции* | **eraseSetStr5** |
| *scp-оператор поиска трехэлементной sc-конструкции* | **searchElStr3** |
| *scp-оператор поиска пятиэлементной sc-конструкции* | **searchElStr5** |
| *scp-оператор поиска трехэлементной sc-конструкции с формированием множеств* | **searchSetStr3** |
| *scp-оператор поиска пятиэлементной sc-конструкции с формированием множеств* | **searchSetStr5** |
| *scp-оператор поиска sc-конструкции по произвольному образцу* | **sys_search** |
| *scp-оператор проверки типа sc-элемента* | **ifType** |
| *scp-оператор проверки наличия значения у переменной* | **ifVarAssign** |
| *scp-оператор проверки наличия содержимого у sc-ссылки* | **ifFormCont** |
| *scp-оператор проверки совпадения значений операндов* | **ifCoin** |
| *scp-оператор проверки равенства числовых содержимых sc-ссылок* | **ifEq** |
| *scp-оператор сравнения числовых содержимых sc-ссылок* | **ifGr** |
| *scp-оператор сложения числовых содержимых sc-ссылок* | **contAdd** |
| *scp-оператор вычитания числовых содержимых sc-ссылок* | **contSub** |
| *scp-оператор умножения числовых содержимых sc-ссылок* | **contMult** |
| *scp-оператор деления числовых* | **contDiv** |

| | |
|---|---|
| *содержимых sc-ссылок* | |
| ***scp-оператор возведения числового содержимого sc-ссылки в степень*** | **contPow** |
| ***scp-оператор вычисления логарифма числового содержимого sc-ссылки*** | **contLn** |
| ***scp-оператор вычисления синуса числового содержимого sc-ссылки*** | **contSin** |
| ***scp-оператор вычисления косинуса числового содержимого sc-ссылки*** | **contCos** |
| ***scp-оператор вычисления тангенса числового содержимого sc-ссылки*** | **contTg** |
| ***scp-оператор вычисления арксинуса числового содержимого sc-ссылки*** | **contASin** |
| ***scp-оператор вычисления арккосинуса числового содержимого sc-ссылки*** | **contACos** |
| ***scp-оператор вычисления арктангенса числового содержимого sc-ссылки*** | **contATg** |
| ***scp-оператор копирования содержимого sc-ссылки*** | **contAssign** |
| ***scp-оператор удаления содержимого sc-ссылки*** | **contErase** |
| ***scp-оператор ожидания события*** | **sys_wait** |
| ***scp-оператор асинхронного вызова подпрограммы*** | **call** |
| ***scp-оператор ожидания завершения выполнения scp-программы*** | **waitReturn** |
| ***scp-оператор ожидания завершения выполнения множества scp-программ*** | **waitReturnSet** |
| ***scp-оператор завершения выполнения программы*** | **return** |
| ***scp-оператор присваивания*** | **varAssign** |

| | |
|---|---|
| *значения переменной* | |
| *scp-оператор удаления значения переменной* | **varErase** |
| *scp-оператор вывода содержимого sc-ссылки* | **print** |
| *scp-оператор вывода содержимого sc-ссылки с переходом на новую строку* | **printNl** |
| *scp-оператор распечатки семантической окрестности sc-элемента* | **printEl** |
| *scp-оператор синхронизации выполнения scp-программы* | **syncronize** |
| *scp-оператор перевода в верхний регистр строкового содержимого sc-ссылки* | **stringToUpperCase** |
| *scp-оператор перевода в верхний регистр строкового содержимого sc-ссылки* | **stringToLowerCase** |
| *scp-оператор замены определенной части строкового содержимого sc-ссылки на содержимое указанной sc-ссылки* | **stringReplace** |
| *scp-оператор проверки совпадения конца строкового содержимого sc-ссылки со строковом содержимым другой sc-ссылки* | **stringEndsWith** |
| *scp-оператор проверки совпадения начальной части строкового содержимого sc-ссылки со строковом содержимым другой sc-ссылки* | **stringStartsWith** |
| *scp-оператор получения части строкового содержимого sc-ссылки по индексам* | **stringSlice** |
| *scp-оператор поиска строкового содержимого sc-ссылки в* | **stringSub** |

| | |
|---|---|
| ***строковом содержимом другой sc-ссылки*** | |
| ***scp-оператор вычисления длины строкового содержимого sc-ссылки*** | **stringLen** |
| ***scp-оператор разбиения строки на подстроки*** | **stringSplit** |
| ***scp-оператор лексикографического сравнения строковых содержимых sc-ссылок*** | **stringIfGr** |
| ***scp-оператор проверки равенства строковых содержимых sc-ссылок*** | **stringIfEq** |
| ***scp-оператор конкатенации строкового содержимого sc-ссылок*** | **contConcat** |

# 5 ПРИМЕРЫ SCP-ПРОГРАММ

Рассмотрим ряд примеров подпрограмм языка SCP, записанных в SCs-коде.

## 5.1 Программа формирования множества всех выходящих sc-дуг для заданного sc-элемента

Параметры программы:

_parameter – заданный sc-элемент

_answer – sc-элемент, обозначающий ответ

Текст программы:

```
scp_program -> proc_search_all_output (*
        -> rrel_params: .proc_search_all_output_params (*
                -> rrel_1: rrel_in: _parameter;;
                -> rrel_2: rrel_in: _answer;;
        *);;

        -> rrel_operators: .proc_search_all_output_operator_set (*

                -> rrel_init: .proc_search_all_output_operator1 (*
                 <- genEl;;
                 -> rrel_1: rrel_assign: rrel_const: rrel_node: rrel_scp_var:
_set_elem;;

                 => nrel_goto: .proc_search_all_output_operator2;;
                *);;

                -> .proc_search_all_output_operator2 (*
                 <- genElStr3;;
                 -> rrel_1: rrel_fixed: rrel_scp_var: _answer;;
                 -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var:
_arc1;;
                 -> rrel_3: rrel_fixed: rrel_scp_var: _parameter;;

                 => nrel_goto: .proc_search_all_output_operator3;;
                *);;

                -> .proc_search_all_output_operator3 (*
                 <- searchSetStr3;;
                 -> rrel_1: rrel_fixed: rrel_scp_var: _parameter;;
                 -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var:
_arc1;;
                 -> rrel_3: rrel_assign: rrel_scp_var: _elem;;

                 -> rrel_set_2: rrel_fixed: rrel_scp_var: _set_elem;;
                 -> rrel_set_3: rrel_fixed: rrel_scp_var: _set_elem;;

                 => nrel_then: .proc_search_all_output_operator4;;
                 => nrel_else: .proc_search_all_output_operator_return;;
                *);;

                /* Elements processing cycle */
                -> .proc_search_all_output_operator4 (*
                 <- searchElStr3;;
```

```
                 -> rrel_1: rrel_fixed: rrel_scp_var: _set_elem;;
                 -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var:
_arc1;;
                 -> rrel_3: rrel_assign: rrel_scp_var: _curr_elem;;

                 => nrel_then: .proc_search_all_output_operator5;;
                 => nrel_else: .proc_search_all_output_operator8;;
                *);;

                -> .proc_search_all_output_operator5 (*
                 <- eraseEl;;
                 -> rrel_1: rrel_fixed: rrel_erase: rrel_scp_var: _arc1;;

                 => nrel_goto: .proc_search_all_output_operator6;;
                *);;

                -> .proc_search_all_output_operator6 (*
                 <- searchElStr3;;
                 -> rrel_1: rrel_fixed: rrel_scp_const: system_element;;
                 -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var:
_arc1;;
                 -> rrel_3: rrel_fixed: rrel_scp_var: _curr_elem;;

                 => nrel_then: .proc_search_all_output_operator4;;
                 => nrel_else: .proc_search_all_output_operator6_1;;
                *);;

                -> .proc_search_all_output_operator6_1 (*
                 <- searchElStr3;;
                 -> rrel_1: rrel_fixed: rrel_scp_var: _answer;;
                 -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var:
_arc1;;
                 -> rrel_3: rrel_fixed: rrel_scp_var: _curr_elem;;

                 => nrel_then: .proc_search_all_output_operator4;;
                 => nrel_else: .proc_search_all_output_operator7;;
                *);;

                -> .proc_search_all_output_operator7 (*
                 <- genElStr3;;
                 -> rrel_1: rrel_fixed: rrel_scp_var: _answer;;
                 -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var:
_arc1;;
                 -> rrel_3: rrel_fixed: rrel_scp_var: _curr_elem;;

                 => nrel_goto: .proc_search_all_output_operator4;;
                *);;

                -> .proc_search_all_output_operator_print_result (*
                 <- printEl;;
                 -> rrel_1: rrel_fixed: rrel_scp_var: _answer;;

                 => nrel_goto: .proc_search_all_output_operator8;;
                *);;

                /* Garbage cleaning */
                -> .proc_search_all_output_operator8 (*
                 <- eraseEl;;
                 -> rrel_1: rrel_fixed: rrel_erase: rrel_scp_var: _set_elem;;

                 => nrel_goto: .proc_search_all_output_operator_return;;
                *);;
```

```
                -> .proc_search_all_output_operator_return (*
                 <- return;;
                *);;
        *);;
*);;
```

### 5.2 Программа формирования множества всех входящих sc-дуг для заданного sc-элемента

Параметры программы:

_parameter – заданный sc-элемент

_answer – sc-элемент, обозначающий ответ

Текст программы:

```
scp_program -> proc_search_all_input (*
        -> rrel_params: .proc_search_all_input_params (*
                -> rrel_1: rrel_in: _parameter;;
                -> rrel_2: rrel_in: _answer;;
        *);;

        -> rrel_operators: .proc_search_all_input_operator_set (*

                -> rrel_init: .proc_search_all_input_operator1 (*
                 <- genEl;;
                 -> rrel_1: rrel_assign: rrel_const: rrel_node: rrel_scp_var:
_set_elem;;

                 => nrel_goto: .proc_search_all_input_operator3;;
                *);;

                -> .proc_search_all_input_operator3 (*
                 <- searchSetStr3;;
                 -> rrel_1: rrel_assign: rrel_scp_var: _elem;;
                 -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var:
_arc1;;
                 -> rrel_3: rrel_fixed: rrel_scp_var: _parameter;;

                 -> rrel_set_1: rrel_fixed: rrel_scp_var: _set_elem;;
                 -> rrel_set_2: rrel_fixed: rrel_scp_var: _set_elem;;

                 => nrel_then: .proc_search_all_input_operator4;;
                 => nrel_else: .proc_search_all_input_operator_return;;
                *);;

                /* Elements processing cycle */
                -> .proc_search_all_input_operator4 (*
                 <- searchElStr3;;
                 -> rrel_1: rrel_fixed: rrel_scp_var: _set_elem;;
                 -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var:
_arc1;;
                 -> rrel_3: rrel_assign: rrel_scp_var: _curr_elem;;

                 => nrel_then: .proc_search_all_input_operator5;;
                 => nrel_else: .proc_search_all_input_operator7_5;;
                *);;

                -> .proc_search_all_input_operator5 (*
```

```
                 <- eraseEl;;
                 -> rrel_1: rrel_fixed: rrel_erase: rrel_scp_var: _arc1;;

                 => nrel_goto: .proc_search_all_input_operator6;;
                *);;

                -> .proc_search_all_input_operator6 (*
                 <- searchElStr3;;
                 -> rrel_1: rrel_fixed: rrel_scp_const: system_element;;
                 -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var:
_arc1;;
                 -> rrel_3: rrel_fixed: rrel_scp_var: _curr_elem;;

                 => nrel_then: .proc_search_all_input_operator4;;
                 => nrel_else: .proc_search_all_input_operator6_1;;
                *);;

                -> .proc_search_all_input_operator6_1 (*
                 <- searchElStr3;;
                 -> rrel_1: rrel_fixed: rrel_scp_var: _answer;;
                 -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var:
_arc1;;
                 -> rrel_3: rrel_fixed: rrel_scp_var: _curr_elem;;

                 => nrel_then: .proc_search_all_input_operator4;;
                 => nrel_else: .proc_search_all_input_operator7;;
                *);;

                -> .proc_search_all_input_operator7 (*
                 <- genElStr3;;
                 -> rrel_1: rrel_fixed: rrel_scp_var: _answer;;
                 -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var:
_arc1;;
                 -> rrel_3: rrel_fixed: rrel_scp_var: _curr_elem;;

                 => nrel_goto: .proc_search_all_input_operator4;;
                *);;

                -> .proc_search_all_input_operator7_5 (*
                 <- genElStr3;;
                 -> rrel_1: rrel_fixed: rrel_scp_var: _answer;;
                 -> rrel_2: rrel_assign: rrel_pos_const_perm: rrel_scp_var:
_arc1;;
                 -> rrel_3: rrel_fixed: rrel_scp_var: _parameter;;

                 => nrel_goto: .proc_search_all_input_operator8;;
                *);;

                -> .proc_search_all_input_operator_print_result (*
                 <- printEl;;
                 -> rrel_1: rrel_fixed: rrel_scp_var: _answer;;

                 => nrel_goto: .proc_search_all_input_operator8;;
                *);;

                /* Garbage cleaning */
                -> .proc_search_all_input_operator8 (*
                 <- eraseEl;;
                 -> rrel_1: rrel_fixed: rrel_erase: rrel_scp_var: _set_elem;;

                 => nrel_goto: .proc_search_all_input_operator_return;;
                *);;
```

```
                -> .proc_search_all_input_operator_return (*
                 <- return;;
                *);;
        *);;
*);;
```

## 6 СОДЕРЖАНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

Пояснительная записка по курсовой работе содержит описание процесса разработки программы решения поставленной теоретико-графовой задачи.

Пояснительная записка должна содержать следующие разделы:

**6.1 Введение.**

Во введении, как правило, освещаются основные положения теории графов, особенности предлагаемых средств разработки. Рассматриваются основные особенности графовых языков программирования, в частности языка SCP.

**6.2 Постановка задачи и разработка алгоритма решения.**

В разделе уточняется постановка задачи, выделяются основные понятия, знание которых необходимо для решения поставленной задачи. Для каждого выделенного понятия приводится определение со ссылками на литературу, примерами по возможности. Подробно рассматривается алгоритм решения поставленной задачи без привязки к какому-либо традиционному варианту представления графовых конструкций в памяти компьютера. Приводятся примеры практического применения теоретико-графовой задачи, рассматриваемой в рамках курсовой работы.

**6.3 Постановка задачи и разработка алгоритма решения.**

В разделе приводится описание нескольких (как минимум пяти) тестовых примеров графовых конструкций, удовлетворяющих условию поставленной задачи, на которых будет производиться отладка реализованного алгоритма решения и демонстрация его работоспособности.

**6.4 Реализация алгоритма решения задачи на основе библиотеки для работы с семантической памятью.**

В разделе приводится описание реализации алгоритма решения поставленной теоретико-графовой задачи на основе библиотеки для работы с семантической памятью.

Указываются ключевые узлы, введенные пользователем для решения конкретной задачи, использованные компоненты, спроектированные сторонними разработчиками.

Описываются проблемы, возникшие в процессе решения задачи, особенности используемых средств проектирования.

**6.5 Реализация алгоритма решения задачи с помощью языка SCP.**

В данном разделе приводится описание реализации алгоритма решения поставленной теоретико-графовой задачи при помощи языка программирования SCP, ориентированного на обработку семантических сетей.

Описывается процесс формализации тестовых примеров при помощи текстового варианта представления семантических сетей – языка SCs. [5], [6]

Описываются проблемы, возникшие в процессе решения задачи, особенности используемых средств проектирования.

## СПИСОК ВОЗМОЖНЫХ ВАРИАНТОВ ТЕМ КУРСОВОЙ РАБОТЫ

1. Определить вид графа:
   1.1. Дерево
   1.2. Ациклический граф
   1.3. Полуэйлеров/эйлеров граф
   1.4. Гамильтонов граф
   1.5. Связный граф
   1.6. Сильно-связный граф
   1.7. Двусвязный граф
   1.8. Двудольный неориентированный граф
   1.9. Регулярный, реберно-регулярныйнеориентированный граф
   1.10. Симметричный, антисимметричный, частично симметричный орграф
   1.11. Транзитивный, антитранзитивный, частично транзитивный орграф
   1.12. Рефлексивный, антирефлексивный, частично рефлексивныйорграф
   1.13. Функциональный, контрафункциональныйорграф
   1.14. Односторонне связный орграф
   1.15. Кактус
   1.16. Турнир, транзитивный турнир
   1.17. Граф Паппа
   1.18. Планарный граф
   1.19. Транзитируемый граф
2. Определить числовую характеристику графа:
   2.1. Радиус
   2.2. Диаметр
   2.3. Средний диаметр
   2.4. Полустепени захода/исхода и средние полустепени всех вершин в орграфе
   2.5. Минимальную степень/среднюю степень/максимальную степень ребра в неориентированном графе
   2.6. Число вершинной связности
   2.7. Число рёберной связности)
   2.8. Среднее и максимальное расстояние между центральными вершинаминеориентиро-ванного графа
   2.9. Число хорд неориентированного графа
   2.10. Минимальное и среднее расстояние между периферийными вершинами неориентированного графа
   2.11. Окружение орграфа
   2.12. Обхват орграфа
   2.13. Число компонентов связности неориентированного графа
   2.14. Число Хадвигера для неориентированного графа
   2.15. Определить толщину неориентированного графа
   2.16. Индекс компонент относительно простой цепи в неориентированном графе

3. Операции над графами:
   3.1. Найти декартово произведение двух неориентированных графов
   3.2. Найти декартову сумму двух неориентированных графов
   3.3. Найти прямое (тензорное) произведение двух неориентированных графов
   3.4. Найти сильное произведение двух неориентированных графов
   3.5. Найти композицию двух неориентированных графов
   3.6. Найти модульное произведение двух неориентированных графов
   3.7. Найти большое модульное произведение двух неориентированных графов
   3.8. Найти объединение множества неориентированных графов
   3.9. Найти пересечение множества неориентированных графов
   3.10.
   3.11. Найти дополнение и фактор-дополнение неориентированного графа
   3.12. Найти граф инциденций неориентированного графа
   3.13. Найти реберный граф для неориентированного графа
   3.14. Найти граф смежностей для неориентированного графа
   3.15. Найти тотальный граф для неориентированного графа
   3.16. Найти граф замыкания неориентированного графа
   3.17. Найти граф конденсации для орграфа
   3.18. Найти граф каркасов для неориентированного графа
4. Поиск в графе:
   4.1. Определить эксцентриситет каждой вершины в неориентированном графе
   4.2. Найти эйлеров цикл в графе
   4.3. Найти гамильтонов цикл
   4.4. Найти компоненты связности в неориентированном графе
   4.5. Найти сильные компоненты связности в орграфе
   4.6. Найти вершины с указанной степенью
   4.7. Найти минимальный остов в неориентированном графа
   4.8. Найти доли неориентированного графа
   4.9. Найти тупики/антитупики в ориентированном графе
   4.10. Найти максимальный простой разрез
   4.11. Найти минимальный простой разрез
   4.12. Найти множество рёбер, удаление которых приводит к увеличению числа компонентов связности ориентированного графа
   4.13. Найти точки сочленения неориентированного графа
   4.14. Найти мосты в неориентированном графе
   4.15. Найти множество вершин, удаление которых приводит к увеличению числа компонентов связности ориентированного графа
   4.16. Найти циклы указанной длины
   4.17. Найти максимальный путь между заданными вершинами
   4.18. Нахождение критических путей во взвешенном неориентированном графе

4.19. Найти множество вершин, удаление которых приводит к увеличению числа компонентов связности неориентированного графа
4.20. Найти простые цепи указанной длины
4.21. Найти вершины с указанной полустепенью
4.22. Найти все доминирующие вершины
4.23. Найти центры графа
4.24. Найти все периферийные вершины
4.25. Найти множество рёбер, удаление которых приводит к увеличению числа компонентов связности неориентированного графа
4.26. Найти звёзды с заданным числом листьев
4.27. Найти дерево кратчайших путей
4.28. Найти n-фактор для указанного графа
4.29. Поиск подграфов в неориентированном графе, изоморфных графу-образцу
4.30. Сформировать множество различных суграфов неориентированного графа
4.31. Сформировать множество различных подграфов неориентированного графа

5. Приведение графа к указанному виду:
5.1. Найти минимальное множество вершин неориентированного графа, удаление которых позволяет сделать его деревом
5.2. Найти минимальное множество рёбер неориентированного графа, удаление которых позволяет сделать его деревом
5.3. Найти минимальное множество вершин неориентированного графа, удаление которых позволяет сделать его планарным
5.4. Найти минимальное множество рёбер графа, удаление которых позволяет сделать его планарным

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Голенков, В.В. Представление и обработка знаний в графодинамических ассоциативных машинах / В. В. Голенков, Д.В. Шункевич, И.Т. Давыденко и др.]; под ред. В.В. Голенкова – Минск, 2001.

[2] Голенков, В. В. Семантическая технология проектирования интеллектуальных решателей задач на основе агентно-ориентированного подхода / В. В. Голенков, Д.В.Шункевич, И.Т.Давыденко; - Программные системы и вычислительные методы. – 2013. – №1.

[3] Шункевич, Д. В., Модели и средства компонентного проектирования машин обработки знаний на основе семантических сетей. В сб.: Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2013): материалы Междунар.научн.-техн.конф. Минск, 21-23 февраля 2013 г.) – Минск: БГУИР, 2013. –С. 269-280

[4] Тарасов, В.Б. От многоагентных систем к интеллектуальным организациям / В.Б. Тарасов; – М. :Изд-во УРСС, 2002.

[5] Гулякина, Н. А. Методика проектирования семантической модели интеллектуальной справочной системы, основанная на семантических сетях / Н. А. Гулякина, И.Т.Давыденко, Д.В.Шункевич; - Программные системы и вычислительные методы. – 2013. – №1.

[6] Давыденко, И. Т., Технология компонентного проектирования баз знаний на основе унифицированных семантических сетей. В сб.: Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2013): материалы Междунар.научн.-техн.конф. Минск, 21-23 февраля 2013 г.) – Минск: БГУИР, 2013. –С. 185-190

[7] Оре О. Теория графов. - 2-е изд.. - М.: Наука, 1980. - С. 336.

[8] Кормен Т. Х. и др. Часть VI. Алгоритмы для работы с графами // Алгоритмы: построение и анализ = Introduction to Algorithms. - 2-е изд. М.: Вильямс, 2006. - С. 1296.

[9] Нечипуренко, М. И. Алгоритмы и программы решения задач на графах и сетях / М.И. Нечипуренко, В.К. Попков, С.М. Майнагашев и др. Новосибирск: Наука. Сиб. отд-ние, 1990. 515 с.

[10] Кузнецов, О.П. Дискретная математика для инженера. Изд. 6, стереотипное. / О.П. Кузнецов; - СПБ, Лань, 2009.