4. Базовый язык программирования для графодинамических ассоциативных машин

Данный раздел посвящен рассмотрению языка программирования, обеспечивающего эффективную интерпретацию всевозможных графодинамических моделей, т.е. языка, используемого для записи микропрограмм операций интерпретируемых графодинамических моделей [342] (***ПредсИОЗвГАМ-2001кн***).

Реализацию произвольной графодинамической формальной модели условно можно разбить на три этапа:

- приведение реализуемой графодинамической модели к каноническому виду, т.е. построение эквивалентной sc-модели;
- запись микропрограмм операций (правил вывода) для канонического вида реализуемой формальной модели (sc-модели) на некотором языке программирования;
- техническая разработка интерпретатора или компилятора указанного языка программирования, на котором записываются микропрограммы операций реализуемых формальных моделей.

Язык программирования, предназначенный для записи микропрограмм операций реализуемых sc-моделей, должен быть:

- языком процедурного программирования, так как микропрограммы абстрактных машин носят процедурный характер;
- языком параллельного и асинхронного программирования, так как реализация различных операций формальной модели в общем случае носит параллельный и асинхронный характер. Процесс реализации каждой операции также может носить параллельный и асинхронный характер, что существенно повышает производительность интеллектуальных систем. Необходимость в этом возникает при реализации сложных операций, к которым, в частности, относится большинство операций логического вывода в моделях обработки знаний;
- языком программирования, хорошо приспособленным к переработке sc-конструкций, т.е. графовых конструкций, принадлежащих языку SC. Такая приспособленность, в частности, требует наличия средств ассоциативного доступа к элементам перерабатываемой графовой конструкции.

В качестве языка процедурного программирования, предназначенного для эффективной интерпретации графодинамических моделей различного вида, предлагается язык SCP (Semantic Code Programming). Язык SCP относится к классу графовых языков программирования. Особенностью этого языка является то, что не только данные, но и сами программы, написанные на языке SCP, представляются в виде sc-конструкций. Язык SCP является языком параллельного асинхронного программирования. Это необходимо для того, чтобы обеспечить адекватную интерпретацию не только последовательных, но и параллельных, не только синхронных, но и асинхронных формальных моделей.

Данный раздел может быть использован в качестве учебного пособия по дисциплине «Конструирование программ и языки программирования в традиционных и интеллектуальных компьютерах» специальности «Искусственный интеллект».

4.1. Принципы языка SCP (Semantic Code Programming)

Язык SCP относится к классу процедурных языков параллельного программирования, ориентированных на переработку нечисловой информации.

Особенностями языка SCP по сравнению с другими языками указанного класса являются:

- приспособленность к переработке нечисловых структур, имеющих мощные выразительные возможности, т.е. обеспечивающих описание любых сложноструктурированных предметных областей;
- использование структурно перестраиваемой графодинамической ассоциативной памяти. Как тексты языка SCP (scp-процедуры), так и перерабатываемые данные являются графовыми конструкциями, хранимыми в графовой структурно перестраиваемой ассоциативной памяти (это означает, что язык SCP относится к классу графовых языков);
- ориентация на переработку непосредственно семантических сетей, а не структур (например, списковых), с помощью которых семантические сети кодируются (это означает, что язык SCP описывает переработку информации непосредственно на уровне семантического кодирования);

- возможность описания (с помощью scp-процедур) механизмов решения задач (логических операций, операций вывода), имеющих различный уровень сложности и поддерживающих самые различные стратегии решения задач;
- высокий потенциал распараллеливания процессов переработки информации в графовой структурно перестраиваемой ассоциативной памяти (как на уровне реализации различных логических операций, так и на уровне реализации каждой из этих операций);
- возможность через понятие sc-узла интегрировать процедуры над sc-конструкциями с процедурами, написанными на традиционных языках и описывающими переработку обычных информационных объектов. Это обеспечивается тем, что содержимым sc-узла может быть информационный объект любой природы.

Из всех вышеперечисленных отличительных особенностей языка SCP самой существенной является ориентация на принципиально иную модель памяти – ориентация на использование графодинамической ассоциативной памяти.

Ближайшими аналогами языка SCP являются:
- язык ВОЛНА (WAVE) П.С. Сапатого [366] (***Сапатый П.С.1986ст-ЯзыкВОЛНА-0***);
- язык передачи маркеров, используемый в качестве внутреннего языка параллельного компьютера маркерной обработки семантических сетей (marker-passing parallel computer) [539] (***Moldovan D.I..1985art-SNAP***).

Принципы языка SCP хорошо сочетаются с идеологией языков и моделей продукционного (ситуационного) программирования [210] (***Кауфман В.Ш.1993кн-ЯзыкиП***), к числу которых относятся нормальные алгоритмы Маркова [284] (***Марков А.А.1954ст-ТеориА***), язык РЕФАЛ [399] (***Турчин В.Ф.1968.пр-АлгорЯРФ***), всевозможные формальные грамматики [101] (***Гладкий А.В.1973кн-ФормаГиЯ***).

Тест языка SCP с формальной точки зрения есть определенным образом устроенная sc-конструкция, хранимая в sc-памяти и описывающая некоторую систему параллельных взаимодействующих процессов переработки другой sc-конструкции, хранимой в той же sc-памяти.

Язык SCP разрабатывается и реализуется в нескольких модификациях, имеющих одинаковый набор непосредственно выполняемых операций, но различный синтаксис. Система операций языка SCP описана в [116; 117] (***Голенков В.В..1994мо-ОписаСЯSCPas1 Голенков В.В..1994мо-ОписаСЯSCPas2***). В 1993 г. был реализован интерпретатор, обеспечивающий выполнение всех операций, включенных в текущую версию языка SCP. Для каждой из модификаций языка SCP дополнительно к указанному интерпретатору требуется реализовать конвертор из соответствующей формы представления scp-процедур в форму, поддерживаемую интерпретатором. Основная модификация языка SCP характеризуется графовым синтаксисом [113] (***Голенков В.В.,1995мо-ОписаЯSCP***). Другая его модификация, называемая SCPas (Semantic Code Programming assembler), имеет синтаксис языка ассемблерного типа. Каждая процедура на языке SCPas представляет собой последовательность scpas-операторов, содержащих код операции и перечень операндов, причем каждой операции языка SCP, непосредственно выполняемой интерпретатором, в языке SCPas соответствует свой код операции. Конвертор из SCPas в форму scp-программ, поддерживаемую интерпретатором, имеет достаточно простой вид. Исчерпывающее описание реализованной версии SCPas представлено в [119; 120; 121] (***Голенков В.В..1994пр-ОписаЯSCPas1; Голенков В.В..1994пр-ОписаЯSCPas2; Голенков В.В..1994пр-ОписаЯSCPas3***).

Другие модификации языка SCP характеризуются более удобным для программиста синтаксисом, что связано с относительно небольшим набором кодов операций и большим или меньшим (различным для разных модификаций) разнообразием операторных выражений. Конверторы из таких форм представления scp-процедур в форму, поддерживаемую интерпретатором языка SCP, имеют большое сходство с компиляторами традиционных и относительно простых алгоритмических языков высокого уровня, таких как Бейсик или Паскаль. Из модификаций рассматриваемого класса следует упомянуть SCPbs (Semantic Code Programming basic – базовый SCP) [122] (***Голенков В.В..1995мо-ОписаЯSCPbs***), а также SCPso (Semantic Code Programming source – исходный SCP). Язык SCPso отличается от всех остальных версий языка SCP более высокими выразительными возможностями, позволяющими создавать более компактные и наглядные программы.

4.2. Описание ядра языка SCP

4.2.1. Основные средства ядра языка SCP

Программы на языке SCP (scp-программы) представляются в виде специальным образом устроенных sc-конструкций, поэтому язык SCP следует считать подъязыком языка SC, т.е. одним из конкретных языков, построенных на базе SC.

Описание заголовка программы языка SCP

Язык представления данных для scp-программ полностью совпадает с языком SC. Данными scp-программ являются sc-конструкции произвольного вида и, в частности, конструкции любого графового языка, построенного на базе языка SC, например, конструкции логического языка SCL и даже конструкции самого языка SCP. SCP-программа может входить в состав данных по отношению к какой-либо другой scp-программе либо по отношению к самой себе. SCP-программа представляет собой соединение sc-конструкции, которая является декларативной частью (заголовком) этой scp-программы (см. scg-текст 4.2.1.1), а также sc-конструкций, являющихся операторами scp-программы. Соединение sc-конструкций предполагает склеивание синонимичных sc-элементов соединяемых sc-конструкций.

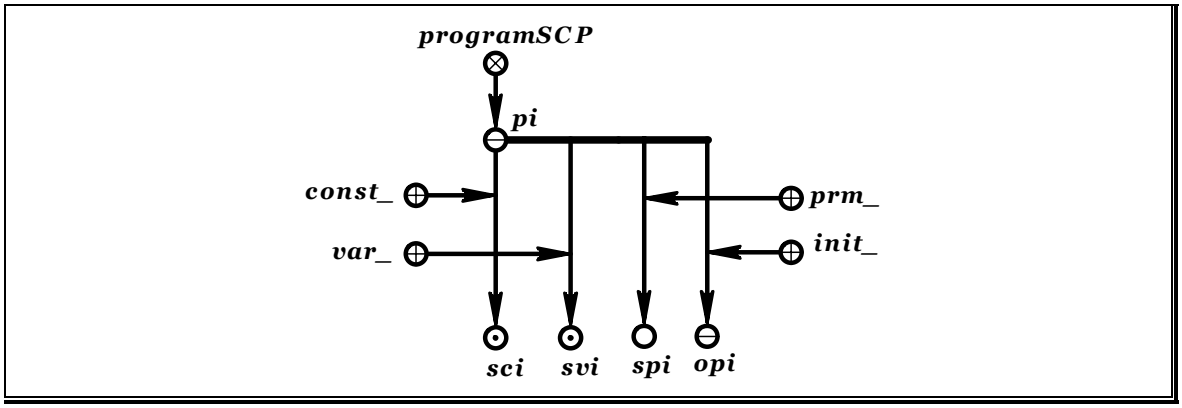
При представлении конструкций языка SCP (scp-операторов и заголовков scp-программ), как и при представлении конструкций любого другого графового языка, используется определенный набор ключевых узлов (см. подраздел 4.3), т.е. узлов, имеющих определенную семантику и обеспечивающих расшифровку произвольных конструкций соответствующего графового языка. Использование ключевых узлов в графовых языках программирования аналогично использованию ключевых слов в символьных языках программирования. Ключевыми узлами языка SCP являются также идентификаторы всех операторов этого языка.

Перейдем к рассмотрению синтаксиса языка SCP.

Заголовок scp-программы (обозначим его ***pi***) (см. scg-текст 4.2.1.1) состоит из следующих элементов:

- знака множества параметров scp-программы (обозначим его ***sy***);
- знака множества констант scp-программы (обозначим его ***sc***);
- знака множества программных переменных scp-программы (обозначим его ***sv***);
- знака первого scp-оператора scp-программы (обозначим его ***opi***);

SCg-текст 4.2.1.1. Заголовок scp-программы



Множество параметров scp-программы (***sv***) – это упорядоченное множество программных переменных этой scp-программы. Это множество упорядочено атрибутами *1_* , *2_* , *3_* и т.д. Параметры, значения которых scp-программа (***pi***) получает из вызвавшей её программы, уточняются атрибутам ***in_***. Параметры, значение которых scp-программа (***pi***) возвращает вызвавшей её программе, уточняется атрибутам ***out_***. Возвращение значений параметров происходит при вызове ***return***-операторов (см. пункт 4.2.7)

Множество констант scp-программы (scp-константы) – это множество константных sc-элементов. Значением scp-константы является узел, обозначающий scp-константу.

Множество программных переменных scp-программы (scp-переменных) – это множество переменных sc-элементов. Значением переменной scp-программы является sc-элемент, связанный бинарным ориентированным отношением с sc-узлом, обозначающим scp-переменную.

Примечание. Значением переменного sc-узла может быть только sc-узел, значением переменной sc-дуги может быть только sc-дуга, значением переменного sc-элемента неопределенного типа может быть как sc-узел, так и sc-дуга.

Описание структуры оператора языка SCP

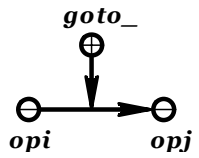
Операторы языка SCP (scp-операторы) разбиваются на типы. Множество типов scp-операторов в свою очередь разбивается на семейства типов scp-операторов. В ядре языка SCP выделено семь семейств типов scp-операторов. Перечислим идентификаторы ключевых узлов, каждый из которых обозначает соответствующее семейство scp-операторов:

- ***gen*** (семейство типов scp-операторов генерации sc-конструкций);
- ***erase*** (семейство типов scp-операторов удаления sc-конструкций);
- ***search*** (семейство типов scp-операторов ассоциативного поиска sc-конструкций);
- ***if*** (семейство типов scp-операторов проверки условий);
- ***change*** (семейство типов scp-операторов изменения свойств sc-элементов);
- ***manage*** (семейство типов scp-операторов управления scp-процессами).

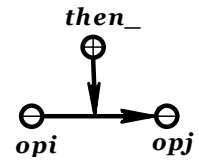
В языке SCP оператор формально определяется как кортеж операндов. Операторы языка SCP разных типов имеют разное количество операндов. В частности в некоторых scp-операторах операнды могут отсутствовать. По характеру передачи управления scp-операторы делятся на три категории:

- операторы, при реализации которых осуществляется безусловная передача управления следующему реализуемому оператору (см. scg-текст 4.2.1.2);
- операторы, при реализации которых осуществляется условная передача управления одному из двух операторов (см. scg-текст 4.2.1.3, scg-текст 4.2.1.4);
- операторы, завершающие реализацию scp-программ, т.е. операторы, при реализации которых не осуществляется передача управления одному из операторов этой же scp-программы.

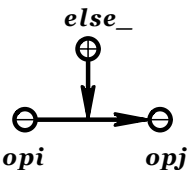
SCg-текст 4.2.1.2. Безусловный переход между scp-операторами



SCg-текст 4.2.1.3. Переход между scp-операторами, если истинно условие



SCg-текст 4.2.1.4. Переход между scp-операторами, если условие ложно



Если рассмотреть scp-операторы, входящие в состав одной scp-программы, то их также можно разбить на следующие категории:

- scp-оператор, которому управление от других scp-операторов не передается, т.е. это scp-оператор, который инициируется первым сразу после инициирования всей scp-программы. Напомним, что первый реализуемый scp-оператор (***opi***) scp-программы (***pi***) задается конструкцией ***pi*** —> ***init_*** **:** ***opi*** **;** ;
- scp-операторы, каждому из которых управление передается только от одного scp-оператора scp-программы;
- scp-операторы, каждому из которых управление передается от нескольких scp-операторов scp-программы. Наличие таких scp-операторов означает либо наличие циклов, либо слияние каких-либо альтернативных ветвей scp-программы.

Операнд scp-оператора scp-программы может быть или scp-константой, или scp-переменной этой scp-программы, т.е. входить в одно из этих sc-множеств. Операнд scp-оператора может уточняться одним или несколькими атрибутами.

Атрибуты операнда scp-оператора разбиваются на следующие группы:

- группа атрибутов порядка (***1_*** , ***2_*** , ***3_*** , ***4_*** и т.д.), которые определяют порядок операндов;
- группа атрибутов уточнения значения операнда (***fixed_***, ***assign_***), которые определяют, известно ли значение операнда (***fixed_***) или же его надо сформировать (***assign_***);
- группа атрибутов уточнения типа значения операнда (***const_*** , ***var_*** , ***pos_***, ***neg_***, ***fuz_***, ***node_***, ***elem_***, ***arc_***, ***meta_***), которые задают тип значения операнда.

Ниже приведем перечень основных атрибутов операнда с их семантикой;

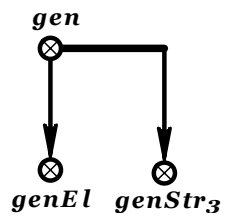
- ***fixed_*** - определяет, что значение операнда сформировано;
- ***assign_*** - определяет, что значение операнда не сформировано и его надо сформировать (за исключением тех случаев, когда операнд является scp-шаблоном);
- ***const_*** - уточняет, что значение операнда должно являться sc-константой;
- ***var_*** - уточняет, что значение операнда должно являться sc-переменной;
- ***pos_*** - уточняет, что значение операнда должно являться позитивной sc-дугой;
- ***neg_*** - уточняет, что значение операнда должно являться негативной sc-дугой;
- ***fuz_*** - уточняет, что значение операнда должно являться нечеткой sc-дугой;
- ***node_*** - уточняет, что значение операнда должно являться sc-узлом;
- ***elem_*** - уточняет, что значение операнда должно являться sc-элементом неопределенного типа;
- ***arc_*** - уточняет, что значение операнда должно являться sc-дугой;
- ***meta_*** - уточняет, что значение операнда должно являться sc-метапеременной;
- ***f_*** - уточняет, что значение операнда должно быть удалено;

4.2.2. Операторы генерации конструкций языка SC

Семейство scp-операторов генерации sc-конструкций (такие операторы будем также называть gen-операторами) разбивается на следующие типы scp-операторов:

- операторы генерации sc-элемента (genEl-операторы);
- операторы генерации sc-конструкций, состоящей из трёх элементов (genStr3-операторы).

Множество genEl-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор ***"genEl"***. Множество genStr3-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор ***"genStr3"***. Семейство типов gen-операторов в языке SCP обозначается ключевым узлом с идентификаторов ***"gen"***. Соотношение между указанными ключевыми узлами задается следующей sc-конструкцией:



Перейдем к более подробному рассмотрению указанных выше типов gen-операторов.

Операторы генерации sc-элемента (genEl-операторы, операторы типа *genEl*)

Операторы данного типа осуществляют генерацию sc-узла или sc-элемента неопределенного типа.

Запись scp-операторов типа ***genEl*** на языке SCs имеет следующий вид (этот вид описан на метаязыке Бэкуса-Наура, см. соответствующее приложение):

< *запись genEl –оператора* > ::=
genEl →
< *идентификатор genEl-оператора* > →
(* ***node_*** **:** ‖ ***elem_*** **:** ***)** — **/*** данные атрибуты уточняют тип генерируемого sc-элемента по признаку "узел – элемент неопределенного типа" ***/**

(* ***const_*** **:** ‖ ***var_*** **:** ‖ ***meta_*** **:** ***)** — **/*** данные атрибуты уточняют тип генерируемого sc-элемента по признаку "константа – переменная – метапеременная" ***/**

< *операнд* > **,** — **/*** операндом является программная переменная, значением которой становится генерируемый sc-элемент ***/**

goto_ **:** < *идентификатор scp-оператора* > **;** — **/*** здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора ***/**

<u>Результаты выполнения операторов типа ***genEl***:</u>

- Если значение операнда не вычислено (т.е. программной переменной, которая является операндом в текущий момент времени не присвоено никакого значения), то генерируется новый sc-элемент. Тип генерируемого элемента определяется атрибутами операнда:
  - ***node_*** - генерируется sc-узел;
  - ***elem_*** - генерируется sc-элемент неопределенного типа;
  - ***const_*** - генерируется константный sc-элемент;
  - ***var_*** - генерируется переменный sc-элемент;
  - ***meta_*** - генерируется метапеременный sc-элемент.

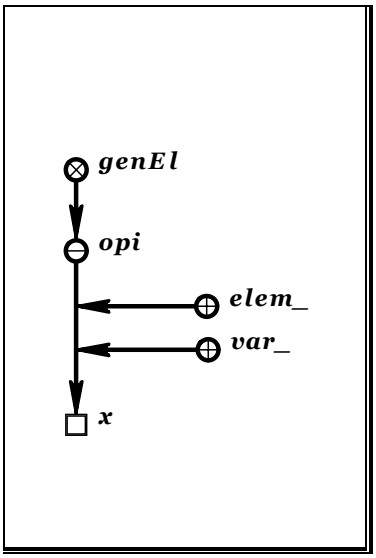
Сгенерированный sc-элеменет становится значением операнда (это означает, что проводится соответствующая бинарная ориентированная sc-дуга, соединяющая операнд с его значением).

- Если значение операнда вычислено и тип sc-элемента, являющегося значением операнда, соответствует типу, задаваемому атрибутом операнда, то в результате реализации оператора изменения состояние памяти не происходит. Иначе генерируется новый sc-элемент требуемого типа (см. выше) и он становится значением операнда (sc-дуга, соединяющая операнд с его предыдущим значением удаляется).

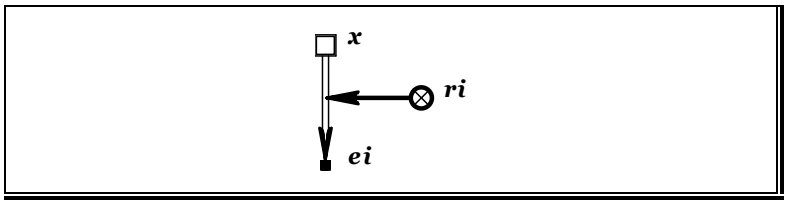
Примечание. В качестве операнда данного scp-оператора может использоваться только программная переменная.

Модификации scp-операторов типа ***genEl*** задаются комбинацией атрибутов, которыми помечен операнд. Всего для genEl-оператора существует 6 модификаций. Приведем некоторые из них:

Пример 4.2.2.1

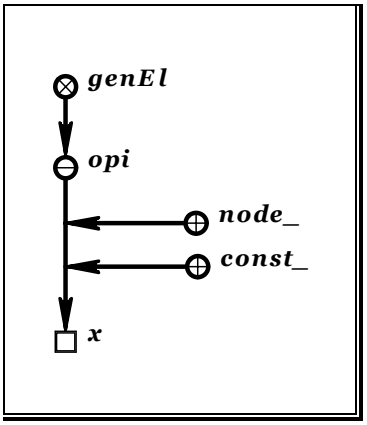


После успешного выполнения данного оператора (***opi***) будет сгенерирован переменный sc-элемент (***ei***) неопределенного типа; также будет сгенерирована дуга, связывающая операнд ***x*** с sc-элементом ***ei*** , и дуга принадлежности, связывающая указанную выше дугу со знаком соответствующего бинарного отношения (***ri***). Генерация этой дуги означает, что у операнда ***x*** сформировалось (было вычислено) значение:

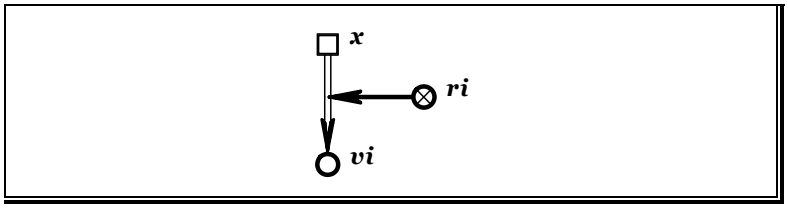


Здесь и далее ***ri*** – бинарное ориентированное отношение, связывающее программные переменные scp-программы, в которую входит оператор ***opi***, с их текущими значениями.

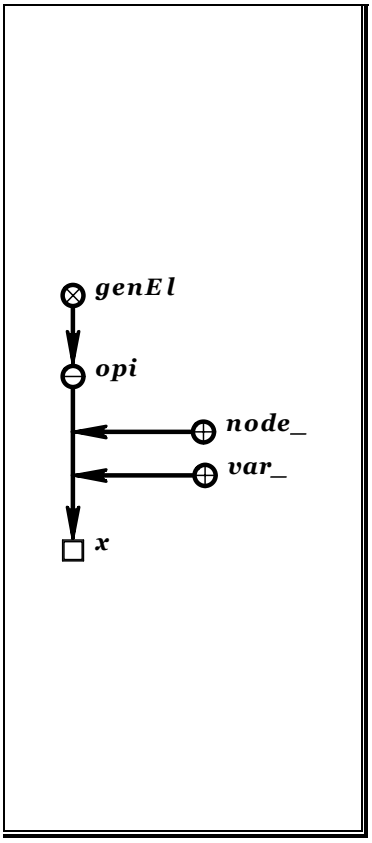
Пример 4.2.2.2



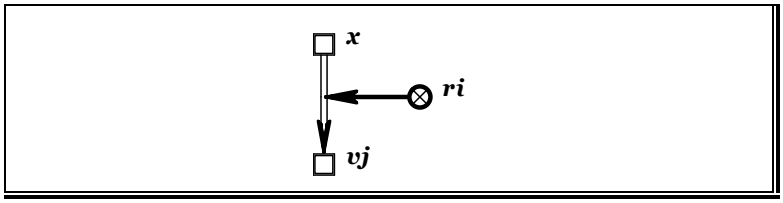
После успешного выполнения данного оператора (***opi***) будет сгенерирован константный sc-узел (***vi***), также будет сгенерированна дуга, связывающая операнд ***x*** с sc-узел ***vi*** , и дуга принадлежности, связывающая указанную выше дугу со знаком соответствующего бинарного отношения (***ri***). Генерация этой дуги означает, что у операнда ***x*** сформировалось (было вычислено) значение:



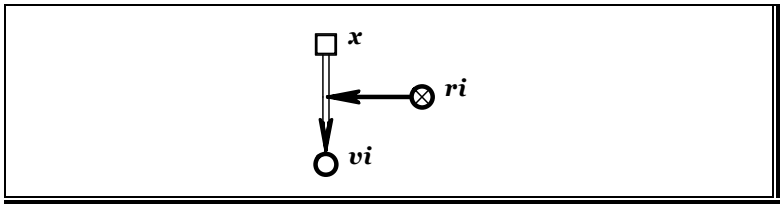
Пример 4.2.2.3



После успешного выполнения данного оператора (***opi***) будет сгенерирован переменный sc-узел (***vj***), также будет сгенерированна дуга, связывающая операнд ***x*** с sc-узел ***vj*** , и дуга принадлежности, связывающая указанную выше дугу со знаком соответствующего бинарного отношения (***ri***). Генерация этой дуги означает, что у операнда ***x*** сформировалось (было вычислено) значение:



Если перед началом выполнения данного оператора (***opi***), в памяти находится следующая конструкция:



То после успешного выполнения оператора ***opi*** дуга между операндом ***x*** и sc-узлом ***vi*** будет удалена.

Перечислим все модификации genEl-операторов:

- операторы генерации константного sc-узла;
- операторы генерации переменного sc-узла;
- операторы генерации метапеременного sc-узла;
- операторы генерации константного sc-элемента неопределенного типа;
- операторы генерации переменного sc-элемента неопределенного типа;
- операторы генерации метапеременного sc-элемента неопределенного типа.

Операторы генерации sc-конструкции, состоящей из трёх элементов (genStr3-операторы, операторы типа *genStr3*)

Операторы данного типа осуществляют генерацию sc-конструкции, состоящей из трех элементов, или каких-либо элементов, входящих в состав этой конструкции.

Запись scp-операторов типа ***genStr3*** на языке SCs имеет следующий вид:

< *запись genStr3 –оператора* > ::=
genStr3 —>
< *идентификатор genStr3-оператора* > —>
(* ***fixed_*** : ‖ ***assign_*** :

/* данные атрибуты уточняют, известно (вычислено) ли значение операнда или необходимо сгенерировать sc-узел и установить значение операнда данного scp-оператора */

(* ***const_*** : ‖ ***var_*** : ‖ ***meta_*** : *) *)

/* данные атрибуты уточняют тип генерируемого sc-элемента по признаку "константа – переменная – метапеременная" */

*1_* : ⪡ *операнд x1* ⪢ ,	/* операндом является scp-переменная, если указан атрибут ***assign_***, значением которой становится генерируемый sc-узел. Если указан атрибут ***fixed_***, то операндом может быть и scp-переменная и scp-константа */
(* ***fixed_*** : ‖ ***assign_*** :	/* данные атрибуты уточняют, известно (вычислено) ли значение операнда или необходимо сгенерировать sc-дугу и установить значение операнда данного scp-оператора */
(* ***const_*** : ‖ ***var_*** : ‖ ***meta_*** : *) *)	/* данные атрибуты уточняют тип генерируемого sc-элемента по признаку "константа – переменная – метапеременная" */
(* ***pos_*** : ‖ ***neg_*** : ‖ ***fuz_*** : *)	/* данные атрибуты уточняют тип генерируемой sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" */
*2_* : ⪡ *операнд x2* ⪢ ,	/* операндом является scp-переменная, если указан атрибут ***assign_***, значением, которой становится генерируемая sc-дуга. Если указан атрибут ***fixed_***, то операндом должна быть scp-переменная, а её значением sc-элемент неопределенного типа */
(* ***fixed_*** : ‖ ***assign_*** :	/* данные атрибуты уточняют, известно (вычислено) ли значение операнда или необходимо сгенерировать sc-элемент и установить значение операнда данного оператора */
(* ***node_*** : ‖ ***elem_*** : *)	/* данные атрибуты уточняют тип генерируемого sc-элемента по признаку "узел – элемент неопределенного типа" */
(* ***const_*** : ‖ ***var_*** : ‖ ***meta_*** : *) *)	/* данные атрибуты уточняют тип генерируемого sc-элемента по признаку "константа – переменная – метапеременная" */
*3_* : ⪡ *операнд x3* ⪢ ,	/* операндом является scp-переменная, если указан атрибут ***assign_***, значением которой становится генерируемый sc-элемент. Если указан атрибут ***fixed_***, то операндом может быть и scp-переменная, и scp-константа */
goto_ : ⪡ *идентификатор scp-оператора* ⪢ ;	/* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора */

<u>Результаты выполнения операторов типа ***genStr3***</u>:

- Если первый операнд помечен атрибутом ***assign_***, то будет сгенерирован новый sc-узел. Тип сгенерированного sc-узла определяется следующими атрибутами:
  - ***const_*** - генерируется константный sc-узел;
  - ***var_*** - генерируется переменный sc-узел;
  - ***meta_*** - генерируется метапеременный sc-узел.

  Сгенерированный sc-узел становится значением операнда (это означает, что проводится соответствующая бинарная ориентированная дуга, соединяющая операнд с его значением).
- Если первый операнд помечен атрибутом ***fixed_*** и его значением является sc-элемент неопределенного типа, то этот sc-элемент неопределенного типа будет преобразован в sc-узел. Тип sc-узла будет таким же, как тип sc-элемента неопределенного типа.
- Если третий операнд помечен атрибутом ***assign_***, то будет сгенерирован новый sc-элемент, тип которого определяется следующими атрибутами:
  - ***node_*** - генерируется sc-узел;
  - ***elem_*** - генерируется sc-элемент неопределенного типа;
  - ***const_, var_, meta_*** – генерируется соответственно константный, переменный или метапеременный sc-элемент.

Сгенерированный sc-элемент становится значением операнда (это означает, что проводится соответствующая бинарная ориентированная дуга, соединяющая операнд с его значением).

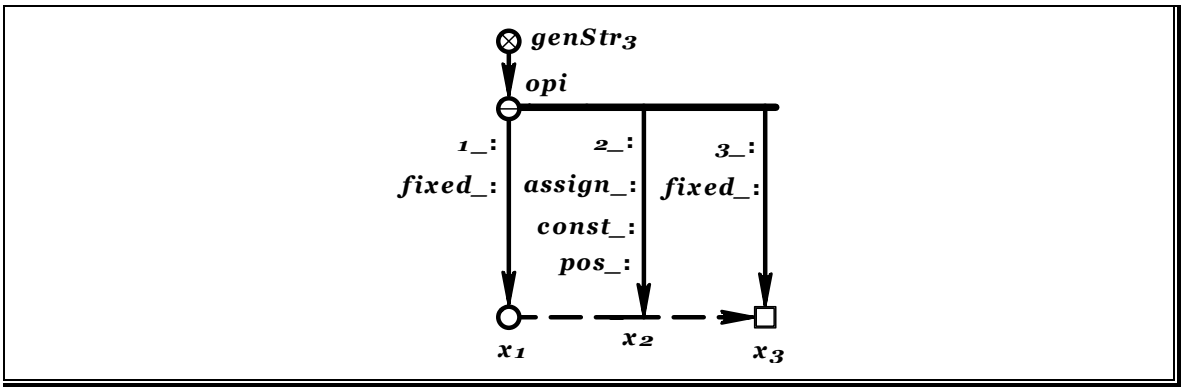
- Если второй операнд помечен атрибутом ***assign_***, то будет сгенерирована новая sc-дуга, выходящая из sc-узла, являющегося значением первого операнда, и входящая в sc-элемент, являющийся значением третьего операнда. Тип сгенерированной sc-дуги определяется следующими атрибутами:
  - ***pos_*** - генерируется позитивная sc-дуга;
  - ***neg_*** - генерируется негативная sc-дуга;
  - ***fuz_*** - генерируется нечеткая sc-дуга;
  - ***const_, var_, meta_*** – генерируется соответственно константная, переменная или метапеременная sc-дуга.

  Сгенерированная sc-дуга становится значением операнда (это означает, что проводится соответствующая бинарная ориентированная дуга, соединяющая операнд с его значением).
- Если второй операнд помечен атрибутом ***fixed_*** и его значением является sc-элемент неопределенного типа, то этот sc-элемент неопределенного типа будет преобразован в sc-дугу, выходящую из sc-узла, являющегося значением первого операнда, и входящая в sc-элемент, являющийся значением третьего операнда. Тип sc-дуги по признаку “константа – переменная – метапеременная” будет таким же, как и у sc-элемента неопределенного типа. Тип sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" определяется следующими атрибутами:
  - ***pos_*** - генерируется позитивная sc-дуга;
  - ***neg_*** - генерируется негативная sc-дуга;
  - ***fuz_*** - генерируется нечеткая sc-дуга.

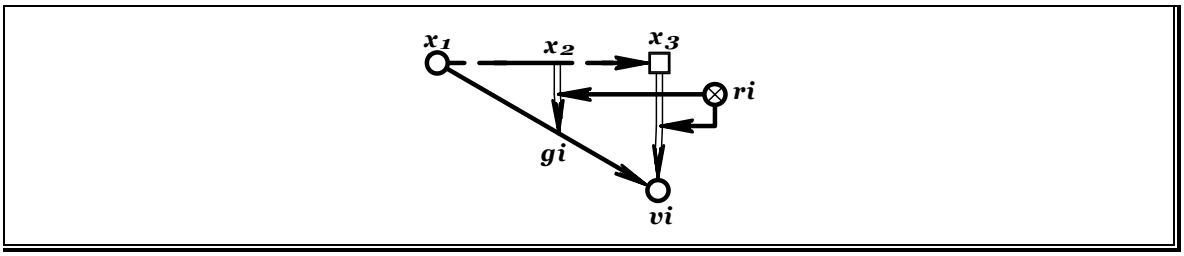
Примечание. Если первый операнд помечен атрибутом ***fixed_***, то его значением должен быть sc-узел или sc-элемент неопределенного типа. Если значением первого операнда является sc-элемент неопределенного типа, то после реализации данного scp-оператора этот sc-элемент неопределенного типа преобразуется в sc-узел. Если третий операнд помечен атрибутом ***fixed_***, то его значением может быть или sc-узел, или sc-дуга, или sc-элемент неопределенного типа.

Модификации scp-операторов типа ***genStr3*** задаются комбинацией атрибутов, которыми помечен операнд. Приведем некоторые из модификаций genStr3-оператора.

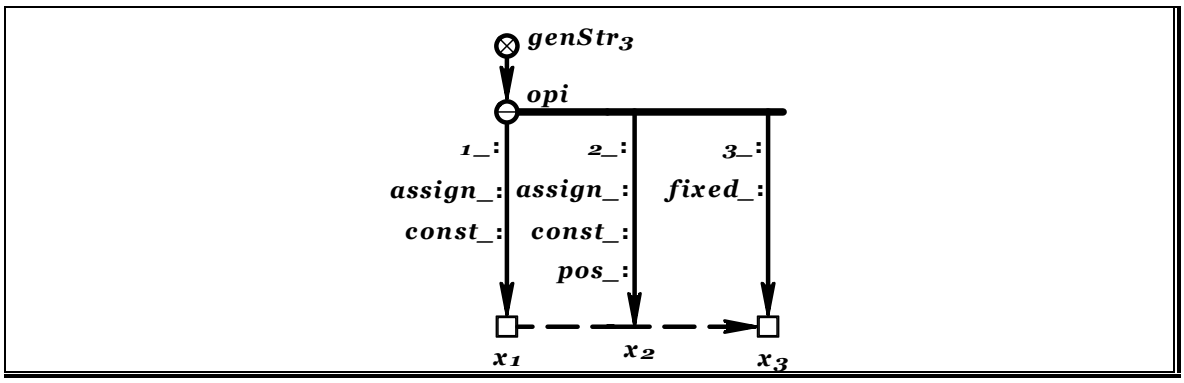
Пример 4.2.2.4



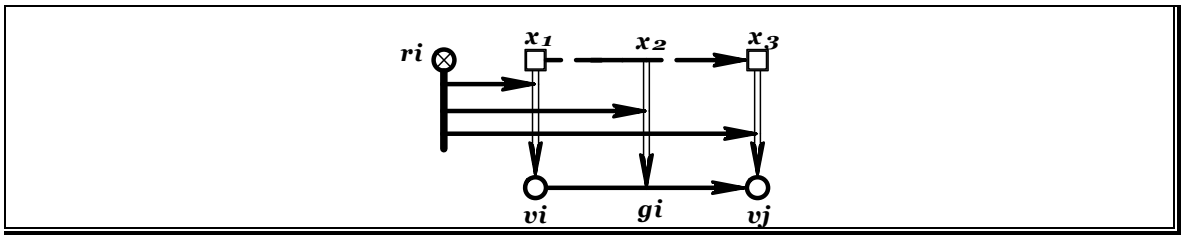
После успешного выполнения данного scp-оператора (***opi***) будет сгенерирована константная позитивная sc-дуга (***gi***), соединяющая значение операнда ***x1*** (так как операнд ***x1*** является scp-константой, то его значением является он сам) и sc-узел ***vi***, который является значением операнда ***x3***. Также будет сгенерированна sc-дуга, связывающая операнд ***x2*** с sc-дугой ***gi***, а также дуга принадлежности, связывающая указанную выше дугу со знаком соответствующего бинарного отношения (***ri***). Генерация этой дуги означает, что у операнда ***x2*** сформировалось (было вычислено) значение:



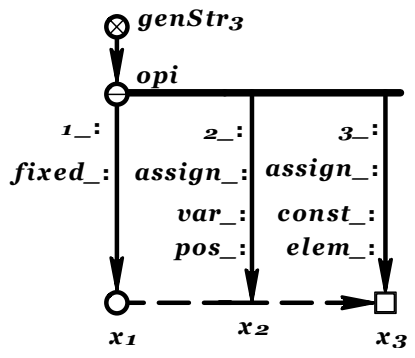
Пример 4.2.2.5



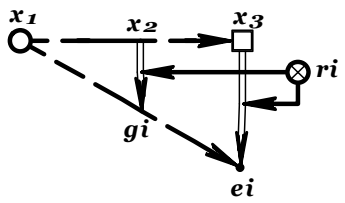
После успешного выполнения данного оператора (***opi***) будут сгенерированы два константных позитивных sc-узла (***vi*** и ***vj***) и соединяющая их дуга (***gi***). Также будут сгенерированны дуги, связывающие операнды ***x1***, ***x2*** и ***x3*** с sc-элементами ***vi***, ***gi*** и ***vj*** соответственно, а также дуги принадлежности, связывающие указанные выше дуги со знаком соответствующего бинарного отношения (***ri***). Генерация этих дуг означает, что у операндов ***x1***, ***x2*** и ***x3*** сформировались (вычислились) значения.



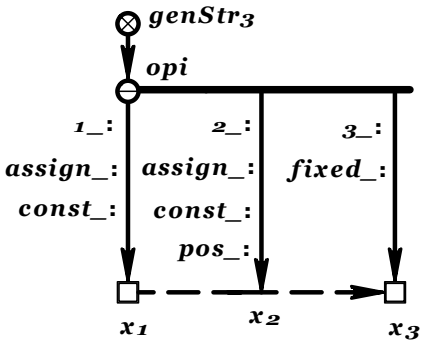
Пример 4.2.2.6



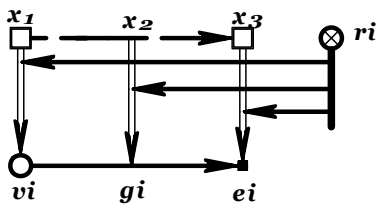
После успешного выполнения данного оператора (*opi*) будут сгенерированы константный sc-элемент неопределенного типа (*ei*) и переменная позитивная sc-дуга (*gi*), соединяющая операнд *x1* и сгенерированный sc-элемент *ei*. Также будут сгенерированы дуга связывающая операнд *x2* с sc-элементом *gi*, дуга связывающая операнд *x3* с sc-элементом *gi*, а также дуги принадлежности, связывающие указанные выше дуги со знаком соответствующего бинарного отношения (*ri*). Генерация этих дуг означает, что у операндов *x2* и *x3* сформировалось (вычислилось) значение:



Пример 4.2.2.7



После успешного выполнения данного оператора (*opi*) будут сгенерированы константный sc-узел *vi* и константная позитивная sc-дуга (*gi*), соединяющая sc-узел *vi* и sc-элемент неопределенного типа *ei*, который является значением операнда *x3*. Также будут сгенерированны дуга связывающая операнд *x2* с sc-узлом *vi*, дуга связывающая операнд *x2* с sc-элементом *gi*, а также дуги принадлежности, связывающие указанные выше дуги со знаком соответствующего бинарного отношения (*ri*). Генерация этих дуг означает, что у операндов *x1* и *x2* сформировалось (вычислилось) значение:

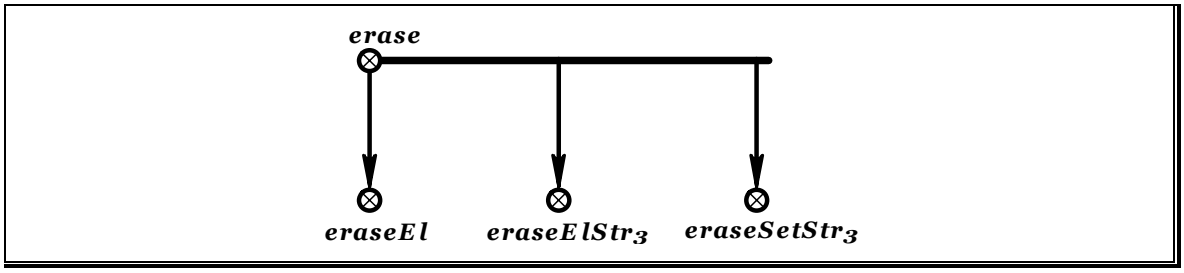


4.2.3. Операторы удаления конструкции языка SC

Семейство scp-операторов удаления sc-конструкций (такие операторы будем также называть erase-операторами) разбивается на следующие типы scp-операторов:

- операторы удаления указанного sc-элемента (eraseEl-операторы);
- операторы удаления указанных элементов sc-конструкции, состоящей из трех элементов (eraseElStr3-операторы);
- операторы удаления нескольких указанных элементов sc-конструкции, состоящей из трех элементов (eraseSetStr3-операторы).

Множество eraseEl-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор ***"eraseEl"***. Множество eraseElStr3-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор ***"eraseElStr3"***. Множество eraseSetStr3-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор ***"eraseSetStr3"***. Семейство типов erase-операторов в языке SCP обозначается ключевым узлом с идентификаторов ***"erase"***. Соотношение между указанными ключевыми узлами задается следующей sc-конструкцией:



Перейдем к более подробному рассмотрению указанных выше типов scp-операторов.

Операторы удаления указанного sc-элемента (eraseEl-операторы, операторы типа *eraseEl*)

Операторы данного типа осуществляют удаление sc-элемента.

Запись scp-операторов типа ***eraseEl*** на языке SCs имеет следующий вид:

< *запись eraseEl -оператора* > **:=**
eraseEl **→**
< *идентификатор eraseEl-оператора* > **→**
< *операнд* > **,** /* операндом является программная переменная, значение которой удаляется */

goto_ **:** < *идентификатор scp-оператора* > **;** /* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора */

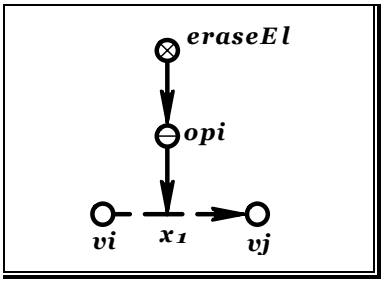
Результат выполнения операторов типа ***eraseEl***:

- удаляется sc-элемент, который является значением операнда, также удаляются все sc-дуги, входящие в этот элемент и выходящие из него (если значением операнда является sc-узел).

Примечание. При удаление sc-элемента также удаляется бинарное ориентированное отношение, связывающее этот sc-элемент с sc-узлом, обозначающим scp-переменную, которая является операндом данного scp-оператора. Таким образом, значение scp-переменной становится не вычисленным.

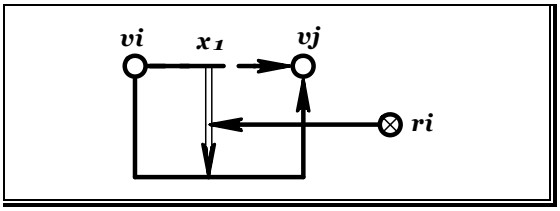
Существует только одна модификация scp-операторов типа ***eraseEl***. Приведем примеры использования данного оператора.

Пример 4.2.3.1

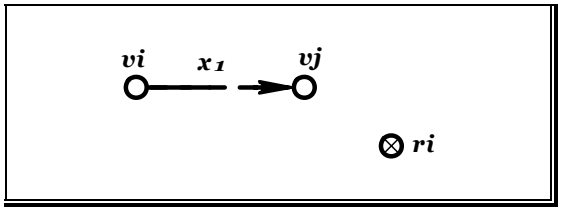


После успешного выполнения данного оператора (*opi*) будет удалена дуга, которая является значением операнда *x2*. Также будут удалены: дуга, связывавшая операнд *x2* с его значением, и дуга принадлежности, связывавшая вышеупомянутую дугу со знаком соответствующего бинарного отношения (*ri*).

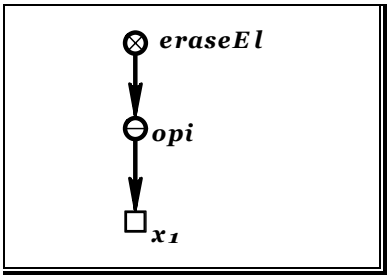
До



После

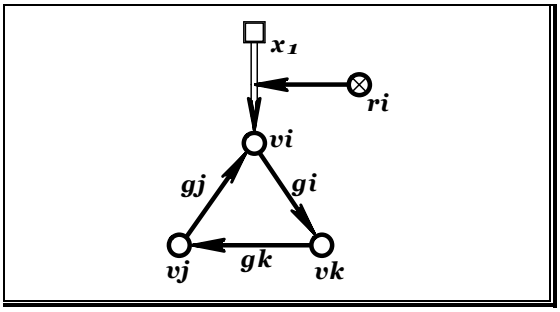


Пример 4.2.3.2

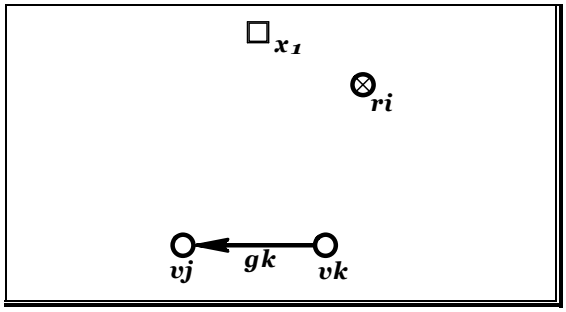


После успешного выполнения данного оператора (*opi*) будет удален узел *vi*, который является значением операнда *x1*. Также будут удалены входящие и выходящие из данного узла дуги (*gi* и *gj*), дуга, связывающая переменную *x1* с её значением, и дуга принадлежности, связывавшая эту дугу со знаком соответствующего бинарного отношения (*ri*).

До



После



Операторы удаления указанных элементов sc-конструкции, состоящей из трех элементов (eraseElStr3-операторы, операторы типа *eraseElStr3*)

Операторы данного типа осуществляют удаление sc-конструкции, состоящей из трех элементов, или каких-либо элементов, входящих в состав этой конструкции.

Запись scp-операторов типа *eraseElStr3* на языке SCs имеет следующий вид:

< *запись eraseElStr3 –оператора* > ::=
eraseElStr3 ⟶
< *идентификатор eraseElStr3-оператора* > ⟶
(* *fixed_* : ‖ /* данный атрибут уточняет, известно (вычислено) ли значение операнда данного scp-оператора */
[* (* *const_* : ‖ *var_* : ‖ *meta_* : *) *] *) /* данные атрибуты уточняют тип удаляемого (искомого) sc-узла по признаку "константа – переменная –

	метапеременная" */
[* *f_* : *]	/* данный атрибут означает, что значение операнда данного scp-оператора или найденный sc-узел должен быть удален */
*1_* : < *операнд x1* > ,	/* операндом является программная переменная, значение которой удаляется, если указан атрибут *f_* */
(* *fixed_* : ‖	/* данный атрибут уточняет, известно (вычислено) ли значение операнда данного scp-оператора */
[* (* *pos_* : ‖ *neg_* : ‖ *fuz_* : *) *]	/* данные атрибуты уточняют тип удаляемой (искомой) sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" */
[* (* *const_* : ‖ *var_* : ‖ *meta_* : *) *] *)	/* данные атрибуты уточняют тип удаляемой (искомой) sc-дуги по признаку "константа – переменная – метапеременная" */
[* *f_* : *]	/* данный атрибут означает, что значение операнда данного scp-оператора или найденная sc-дуга должна быть удалена */
*2_* : < *операнд x2* > ,	/* операндом является программная переменная, значение которой удаляется, если указан атрибут *f_* */
(* *fixed_* : ‖	/* данный атрибут уточняет, известно (вычислено) ли значение операнда данного scp-оператора */
(* [* *node_* : ‖ *elem_* : ‖ *arc_* : *]	/* данные атрибуты уточняют тип удаляемого (искомого) sc-элемента по признаку "узел – элемент неопределенного типа – дуга" */
[* (* *pos_* : ‖ *neg_* : ‖ *fuz_* : *) *] *)	/* данные атрибуты уточняют тип удаляемой (искомой) sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" */
[* (* *const_* : ‖ *var_* : ‖ *meta_* : *) *] *)	/* данные атрибуты уточняют тип удаляемой (искомой) sc-дуги по признаку "константа – переменная – метапеременная" */
[* *f_* : *]	/* данный атрибут означает, что значение операнда данного scp-оператора или найденная sc-дуга должна быть удалена */
*3_* : < *операнд x3* > ,	/* операндом является программная переменная, значение которой удаляется, если указан атрибут *f_* */
*goto_* : < *идентификатор scp-оператора* > ;	/* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора */

Результаты выполнения операторов типа ***eraseElStr3***:

- Если первый операнд не помечен атрибутом ***fixed_***, то будет искаться sc-узел. Тип искомого sc-узла уточняется следующими атрибутами:
  - ***const_*** - ищется константный sc-узел;
  - ***var_*** - ищется переменный sc-узел;
  - ***meta_*** - ищется метапеременный sc-узел.

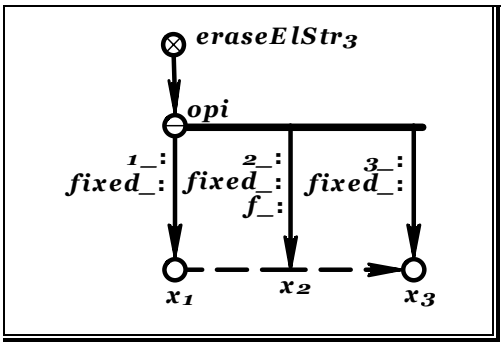
- Если искомый sc-узел найден и указан атрибут $f_$, то этот sc-узел удаляется.
- Если первый операнд помечен атрибутом $fixed_$, а также атрибутом $f_$, то значение операнда удаляется.
- Если второй операнд не помечен атрибутом $fixed_$, то будет искаться sc-дуга. Тип искомой sc-дуги уточняется следующими атрибутами:
  - $pos_$ - ищется позитивная sc-дуга;
  - $neg_$ - ищется негативная sc-дуга;
  - $fuz_$ - ищется нечеткая sc-дуга;
  - $const_$, $var_$, $meta_$ – ищется соответственно константная, переменная или метапеременная sc-дуга.
- Если искомая sc-дуга найдена и указан атрибут $f_$, то эта sc-дуга удаляется.
- Если второй операнд помечен атрибутом $fixed_$, а также атрибутом $f_$, то значение операнда удаляется.
- Если третий элемент не помечен атрибутом $fixed_$, то будет искаться sc-элемент, тип которого уточняется следующими атрибутами:
  - $node_$ - ищется sc-узел;
  - $elem_$ - ищется sc-элемент неопределенного типа;
  - $arc_$ - ищется sc-дуга. В этом случае для уточнения типа могут быть дополнительно использованы атрибуты $pos_$, $neg_$, $fuz_$ (семантику смотри выше);
  - $const_$, $var_$, $meta_$ – ищется соответственно константный, переменный или метапеременный sc-элемент.
- Если искомый sc-элемент найден и указан атрибут $f_$, то этот sc-элемент удаляется.
- Если третий операнд помечен атрибутом $fixed_$, а также атрибутом $f_$, то значение операнда удаляется.

Примечание 1. Хотя бы один операнд данного оператора должен быть помечен атрибутом $fixed_$.

Примечание 2. Если ни один операнд scp-оператора типа $eraseElStr3$ не помечен атрибутом $f_$, то в результате реализации такого оператора изменения состояния памяти не происходит.

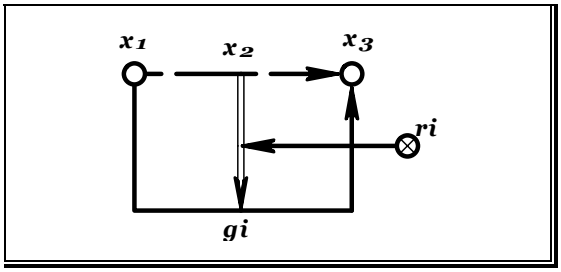
Модификации scp-операторов типа $eraseElStr3$ задаются комбинацией атрибутов, которыми помечен операнд. Приведем некоторые модификации eraseElStr3-оператора:

Пример 4.2.3.3

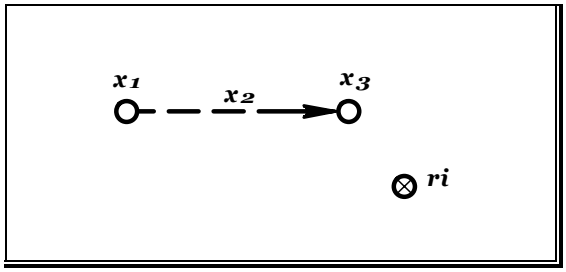


После успешного выполнения данного оператора (opi) будет удалена sc-дуга gi, которая является значением операнда $x2$. Также будут удалены: дуга, связывавшая операнд $x2$ с его значением, и дуга принадлежности, связывавшая данную дугу со знаком соответствующего бинарного отношения (ri). Удаление этой дуги означает, что у операнда $x2$ отсутствует (удалено) значение.

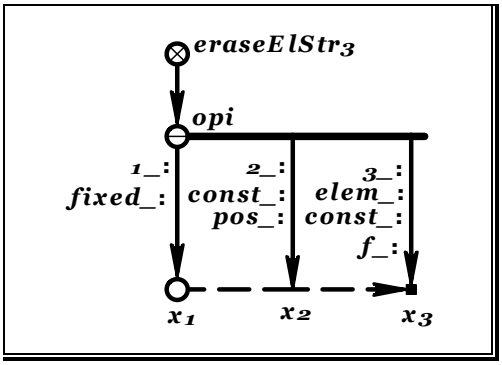
До



После

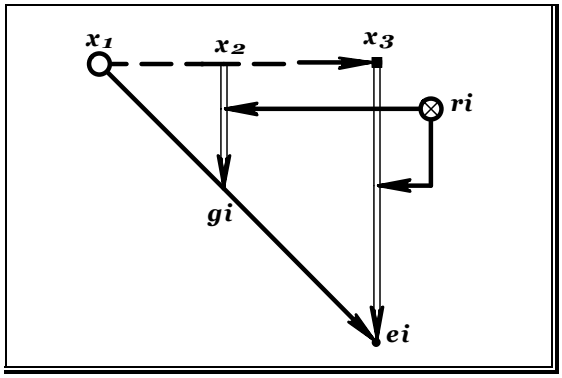


Пример 4.2.3.4

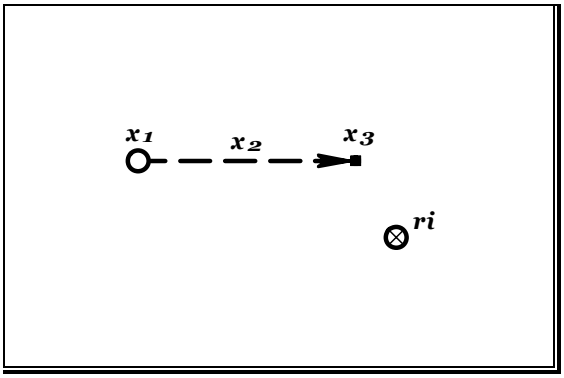


После успешного выполнения данного оператора (*opi*) будет удален константный sc-элемент неопределенного типа, который является значением операнда *x3* . Также будет удалена входящая в него константная позитивная sc-дуга, которая является значением операнда *x2*; дуги, связывавшие операнды *x2* и *x3* с их значениями; дуги принадлежности, связывавшие данные дуги со знаком соответствующего бинарного отношения (*ri*). Удаление этих дуг означает, что у операндов *x2* и *x3* отсутствуют (удалены) значения.

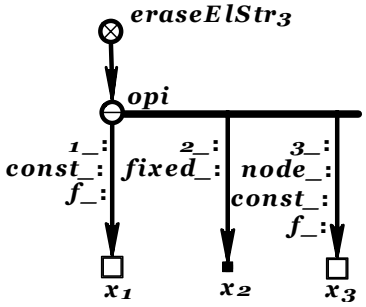
До



После

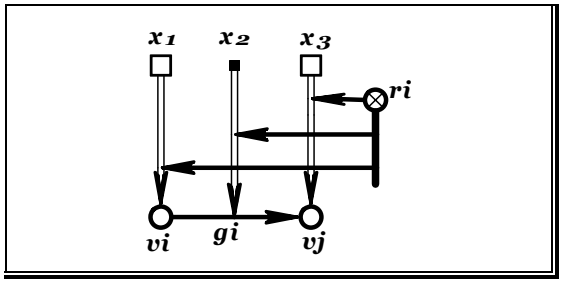


Пример 4.2.3.5

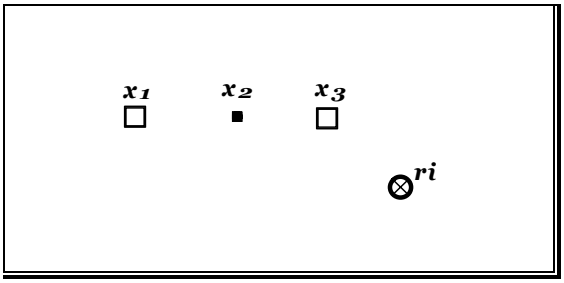


После успешного выполнения данного оператора (*opi*) будут удалены sc-элементы *vi*, *gi* и *vj*, которые являются значениями операндов *x1*, *x2*, *x3* соответственно (очевидно, что когда удаляется первый или третий элемент трехэлементной конструкции, второй элемент тоже будет удален и это можно не уточнять атрибутом *f_*). Также будут удалены дуги, связывавшие операнды *x1*, *x2*, *x3* с их значениями, и дуги принадлежности, связывавшие вышеупомянутые дуги со знаком соответствующего бинарного отношения (*ri*). Удаление этих дуг означает, что у операндов *x1*, *x2*, *x3* отсутствуют (удалены) значения.

До



После



Операторы удаления нескольких указанных элементов sc-конструкции, состоящей из трех элементов (eraseSetStr3-операторы, операторы типа *eraseSetStr3*)

Операторы данного типа осуществляют удаление нескольких указанных элементов sc-конструкции, состоящей из трех элементов, или каких-либо элементов, входящих в состав этой конструкции.

Запись scp-операторов типа ***eraseSetStr3*** на языке SCs имеет следующий вид:

< запись eraseSetStr3 –оператора > **::=**

eraseSetStr3 **—>**

< идентификатор eraseSetStr3-оператора > **—>**

(* ***fixed_*** **:** **‖**	**/*** данный атрибут уточняет, известно (вычислено) ли значение операнда данного scp-оператора ***/**
[* **(*** ***const_*** **:** **‖** ***var_*** **:** **‖** ***meta_*** **:** ***)** ***]** ***)**	**/*** данные атрибуты уточняют тип удаляемого (искомого) sc-узла по признаку "константа – переменная – метапеременная" ***/**
[* ***f_*** **:** ***]**	**/*** данный атрибут означает, что найденный sc-узел должен быть удален ***/**
*1_* **:** *< операнд x1 >* **,**	**/*** операндом является программная переменная, или если указан атрибут ***fixed_***, но не указан атрибут ***f_***, то операндом может быть и константа программы ***/**
(* ***fixed_*** **:** **‖**	**/*** данный атрибут уточняет, известно (вычислено) ли значение операнда данного scp-оператора ***/**
[* **(*** ***pos_*** **:** **‖** ***neg_*** **:** **‖** ***fuz_*** **:** ***)** ***]**	**/*** данные атрибуты уточняют тип удаляемой (искомой) sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" ***/**
[* **(*** ***const_*** **:** **‖** ***var_*** **:** **‖** ***meta_*** **:** ***)** ***]** ***)**	**/*** данные атрибуты уточняют тип удаляемой (искомой) sc-дуги по признаку "константа – переменная – метапеременная" ***/**
[* ***f_*** **:** ***]**	**/*** данный атрибут означает, что найденная sc-дуга должна быть удалена ***/**
*2_* **:** *< операнд x2 >* **,**	**/*** операндом является программная переменная, или если указан атрибут ***fixed_***, но не указан атрибут ***f_***, то операндом может быть и константа программы ***/**
(* ***fixed_*** **:** **‖**	**/*** данный атрибут уточняет известно (вычислено) ли значение операнда данного scp-оператора ***/**
(* **[*** ***node_*** **:** **‖** ***elem_*** **:** **‖** ***arc_*** **:** ***]**	**/*** данные атрибуты уточняют тип удаляемого (искомого) sc-элемента по признаку "узел – элемент неопределенного типа – дуга" ***/**
[* **(*** ***pos_*** **:** **‖** ***neg_*** **:** **‖** ***fuz_*** **:** ***)** ***]** ***)**	**/*** данные атрибуты уточняют тип удаляемой (искомой) sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" ***/**

[* (* ***const_*** : ‖ ***var_*** : ‖ ***meta_*** : *) *] *)	/* данные атрибуты уточняют тип удаляемого (искомого) sc-элемента по признаку "константа – переменная – метапеременная" */
[* ***f_*** : *]	/* данный атрибут означает, что найденный sc-элемент должен быть удален */
*3_* : ⪡ *операнд x3* ⪢ ,	/* операндом является программная переменная, или если указан атрибут ***fixed_***, но не указан атрибут ***f_***, то операндом может быть и константа программы */
[* ***set1_*** : ⪡ *операнд xs1* ⪢ , *]	/* операнд, значением которого является множество “первых” элементов sc-конструкции */
[* ***set2_*** : ⪡ *операнд xs2* ⪢ , *]	/* операнд, значением которого является множество “вторых” элементов sc-конструкции */
[* ***set3_*** : ⪡ *операнд xs3* ⪢ , *]	/* операнд, значением которого является множество “третьих” элементов sc-конструкции */
goto_ : ⪡ *идентификатор scp-оператора* ⪢ ;	/* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора */

Результаты выполнения операторов типа ***eraseSetStr3***:

- Если первый операнд не помечен атрибутом ***fixed_***, то будет искаться sc-узел. Тип искомого sc-узла уточняется следующими атрибутами:
  - ***const_*** - ищется константный sc-узел;
  - ***var_*** - ищется переменный sc-узел;
  - ***meta_*** - ищется метапеременный sc-узел.
- Если искомый sc-узел найден и указан атрибут ***f_***, то этот sc-узел удаляется.
- Если первый операнд помечен атрибутом ***fixed_***, а также атрибутом ***f_***, то значение операнда удаляется.
- Если второй операнд не помечен атрибутом ***fixed_***, то будет искаться sc-дуга. Тип искомой sc-дуги уточняется следующими атрибутами:
  - ***pos_*** - ищется позитивная sc-дуга;
  - ***neg_*** - ищется негативная sc-дуга;
  - ***fuz_*** - ищется нечеткая sc-дуга;
  - ***const_, var_, meta_*** – ищется соответственно константная, переменная или метапеременная sc-дуга.
- Если искомая sc-дуга найдена и указан атрибут ***f_***, то эта sc-дуга удаляется.
- Если второй операнд помечен атрибутом ***fixed_***, а также атрибутом ***f_***, то значение операнда удаляется.
- Если третий элемент не помечен атрибутом ***fixed_***, то будет искаться sc-элемент, тип которого уточняется следующими атрибутами:
  - ***node_*** - ищется sc-узел;
  - ***elem_*** - ищется sc-элемент неопределенного типа;
  - ***arc_*** - ищется sc-дуга. В этом случае для уточнения типа могут дополнительно использоваться атрибуты ***pos_***, ***neg_***, ***fuz_*** (семантику смотри выше);
  - ***const_, var_, meta_*** – ищется соответственно константный, переменный или метапеременный sc-элемент.
- Если искомый sc-элемент найден и указан атрибут ***f_***, то этот sc-элемент удаляется.
- Если третий операнд помечен атрибутом ***fixed_*** а также атрибутом ***f_***, то значение операнда удаляется.

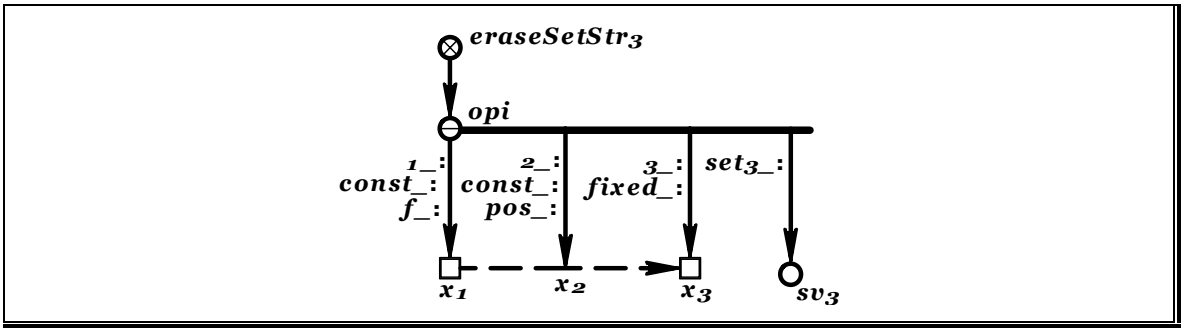
Примечание 1. Последние три операнда данного оператора (*xs1*, *xs2* и *xs3*) задают множества, соответственно, “первых”, “вторых” и “третьих” элементов sc-конструкций. Эти множества определяют область поиска sc-конструкций. В найденных sc-конструкциях удаляются элементы, у операндов которых есть атрибут *f_*.

Примечание 2. Операнды *xs1*, *xs2* и *xs3* могут быть scp-переменными и scp-константами. В описании оператора обязательно должен присутствовать хотя бы один из операндов *xs1*, *xs2* и *xs3*.

Примечание 3. Возможно, что ни один операнд не помечен атрибутом *fixed_*. Операнды *xs1*, *xs2* и *xs3* могут совпадать.

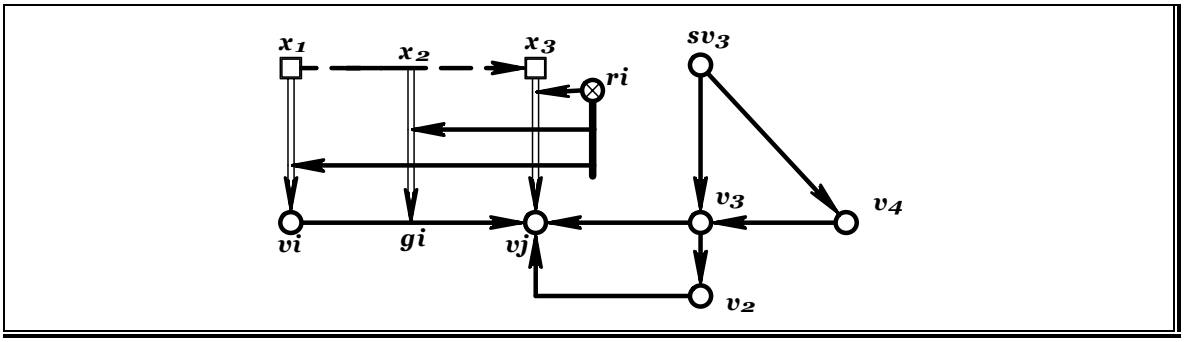
Модификации scp-операторов типа *eraseSetStr3* задаются комбинацией атрибутов, которыми помечен операнд. Приведем некоторые модификации eraseSetStr3-оператора.

Пример 4.2.3.6

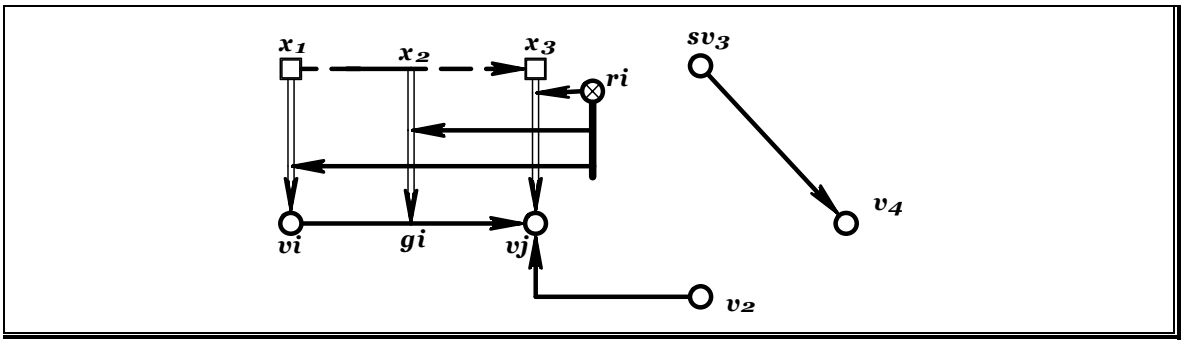


После успешного выполнения данного оператора (*opi*), будет удален sc-узел *v2*, который является первым элементом трехэлементной конструкции, соответствующей заданному шаблону и чей третий элемент *vj* входит в множество *sv3*(состоит из *vj* и *v3*). Также будут удалены дуги, входящие и выходящие из удаленного узла.

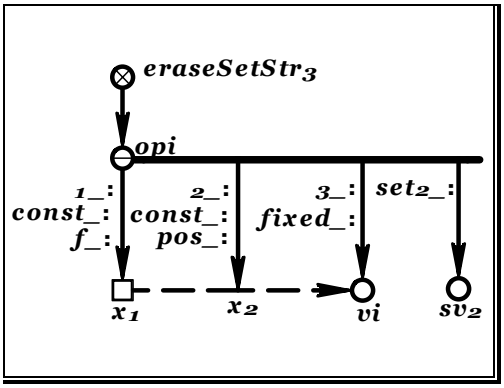
До



После

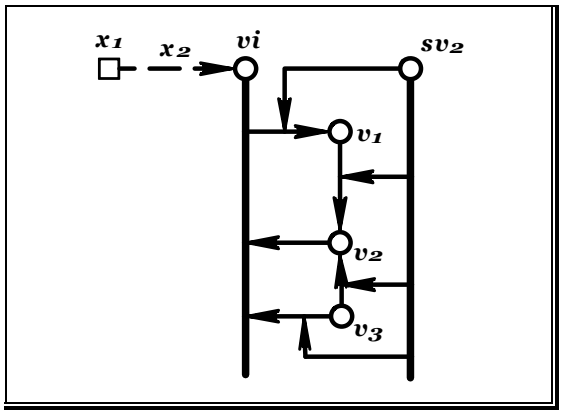


Пример 4.2.3.7

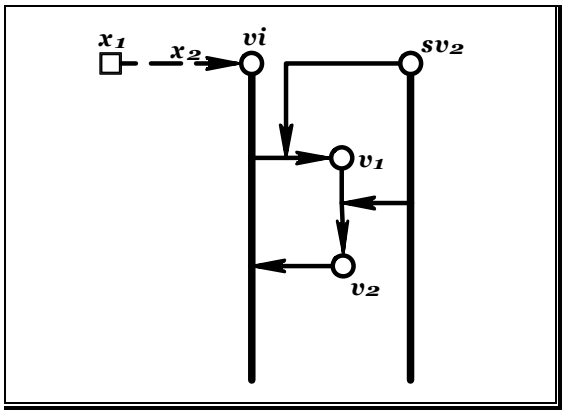


После успешного выполнения данного оператора (*opi*) будет удален sc-узел *v3*, который является первым элементом трехэлементной конструкции, соответствующей заданному шаблону и чей второй элемент входит в множество *sv2*. Также будут удалены дуги, входящие и выходящие из удаленного узла.

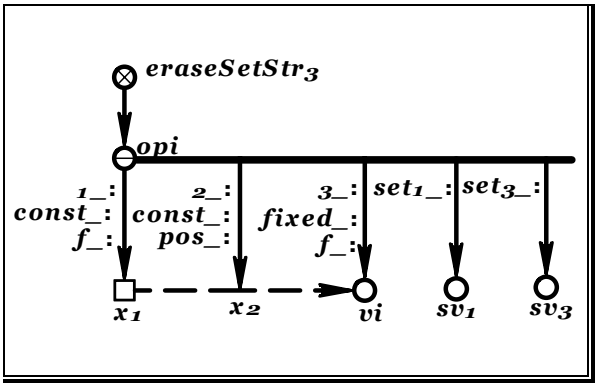
До



После

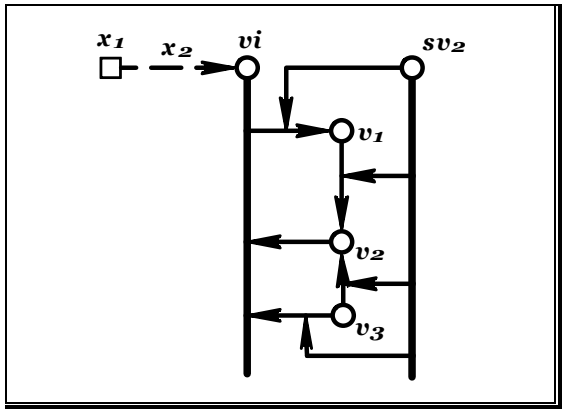


Пример 4.2.3.8

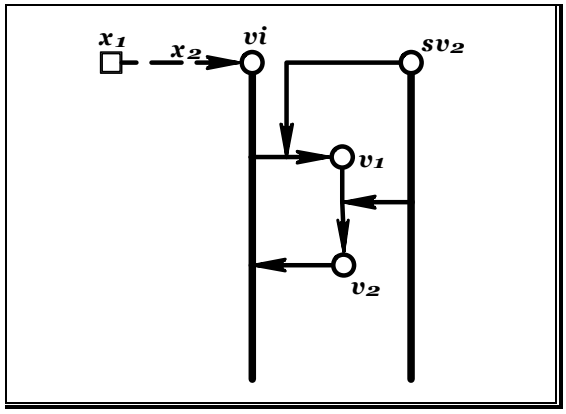


После выполнения данного оператора (*opi)* изменений в памяти не произойдет, так как во множествах *sv1* и *sv3* не содержится трехэлементной sc-конструкции, удовлетворяющей заданному шаблону (*vi* не является третьим элементом ни одной из конструкций).

До



После

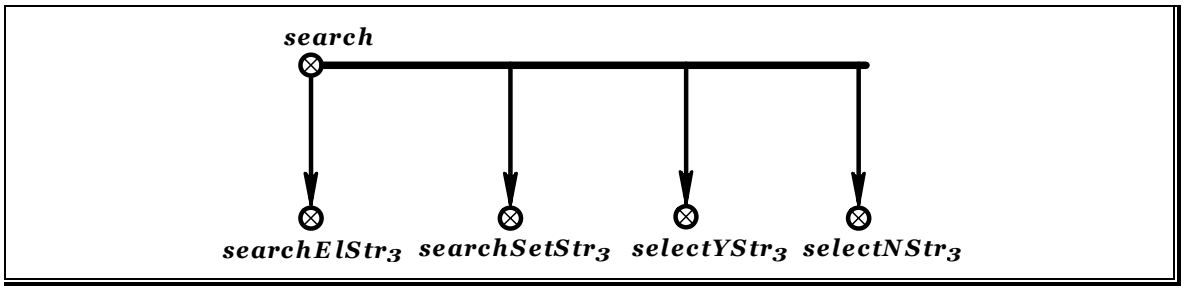


4.2.4. Операторы ассоциативного поиска конструкций языка SC

Семейство scp-операторов ассоциативного поиска sc-конструкций (такие операторы будем также называть search-операторами) разбивается на следующие типы scp-операторов:

- операторы ассоциативного поиска произвольных элементов sc-конструкции, состоящей из трёх элементов (searchElStr3-операторы);
- операторы ассоциативного поиска произвольных элементов sc-конструкций, удовлетворяющих заданным условиям и состоящих из трех элементов, и формирования из них множества (searchSetStr3-операторы);
- операторы ассоциативного поиска произвольных элементов sc-конструкций, удовлетворяющих заданным условиям и состоящих из трёх элементов, и формирования из них множества, элементами которого остаются только sc-элементы, удовлетворяющие заданным условиям (selectYStr3-операторы);
- операторы ассоциативного описка произвольных элементов sc-конструкций, удовлетворяющих заданным условиям и состоящих из трёх элементов, и формирования из них множества, элементами которого остаются только sc-элементы, не удовлетворяющие заданным условиям (selectNStr3-операторы).

Множество searchElStr3-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор ***"searchElStr3"***. Множество searchSetStr3-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор ***"searchSetStr3"***. Множество selectYStr3-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор ***"selectYStr3"***. Множество selectNStr3-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор ***"selectNStr3"***. Семейство типов search-операторов в языке SCP обозначается ключевым узлом с идентификатором ***"search"***. Соотношение между указанными ключевыми узлами задается следующей sc-конструкцией:



Перейдем к более подробному рассмотрению указанных выше типов scp-операторов.

Операторы ассоциативного поиска произвольных элементов sc-конструкции, состоящей из трёх элементов (searchElStr3-операторы, операторы типа *searchElStr3*)

Операторы данного типа осуществляют ассоциативный поиск произвольных элементов sc-конструкции, состоящей из трёх элементов.

Запись scp-операторов типа ***searchElStr3*** на языке SCs имеет следующий вид:

< *запись searchElStr3 –оператора* > ::=
searchElStr3 —>
< *идентификатор searchElStr3-оператора* > —>
(* ***fixed_*** : ‖ ***assign_*** :

/* данные атрибуты уточняют, известно (вычислено) ли значение операнда или необходимо найти sc-элемент и установить значение операнда данного оператора */

[* **(*** ***const_*** **:** ‖ ***var_*** **:** ‖ ***meta_*** **:** ***)*** ***]*** ***)***	**/*** данные атрибуты уточняют тип искомого sc-элемента по признаку "константа – переменная – метапеременная" ***/***
*1_* **:** ⪡ *операнд x1* ⪢ **,**	**/*** операндом является scp-переменная, если указан атрибут ***assign_***, значением которой становится найденный sc-элемент. Если указан атрибут ***fixed_***, то операндом может быть и scp-переменная, и scp-константа ***/***
(* ***fixed_*** **:** ‖ ***assign_*** **:**	**/*** данные атрибуты уточняют, известно (вычислено) ли значение операнда или необходимо найти sc-элемент и установить значение операнда данного оператора ***/***
[* **(*** ***pos_*** **:** ‖ ***neg_*** **:** ‖ ***fuz_*** **:** ***)*** ***]***	**/*** данные атрибуты уточняют тип искомой sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" ***/***
[* **(*** ***const_*** **:** ‖ ***var_*** **:** ‖ ***meta_*** **:** ***)*** ***]*** ***)***	**/*** данные атрибуты уточняют тип искомого sc-элемента по признаку "константа – переменная – метапеременная" ***/***
*2_* **:** ⪡ *операнд x2* ⪢ **,**	**/*** операндом является scp-переменная, если указан атрибут ***assign_***, значением которой становится найденный sc-элемент. Если указан атрибут ***fixed_***, то операндом может быть и scp-переменная и scp-константа ***/***
(* ***fixed_*** **:** ‖ ***assign_*** **:**	**/*** данные атрибуты уточняют, известно (вычислено) ли значение операнда или необходимо найти sc-элемент и установить его значением операнда данного оператора ***/***
(* **[*** ***node_*** **:** ‖ ***elem_*** **:** ‖ ***arc_*** **:** ***]***	**/*** данные атрибуты уточняют тип искомого sc-элемента по признаку "узел – элемент неопределенного типа – дуга" ***/***
[* **(*** ***pos_*** **:** ‖ ***neg_*** **:** ‖ ***fuz_*** **:** ***)*** ***]*** ***)***	**/*** данные атрибуты уточняют тип искомого sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" ***/***
[* **(*** ***const_*** **:** ‖ ***var_*** **:** ‖ ***meta_*** **:** ***)*** ***]*** ***)***	**/*** данные атрибуты уточняют тип искомого sc-элемента по признаку "константа – переменная – метапеременная" ***/***
*3_* **:** ⪡ *операнд x3* ⪢ **,**	**/*** операндом является scp-переменная, если указан атрибут ***assign_***, значением, которой становится найденный sc-элемент. Если указан атрибут ***fixed_***, то операндом может быть и scp-переменная, и scp-константа ***/***
then_ **:** ⪡ *идентификатор scp-оператора* ⪢ **;**	**/*** здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора и нахождения требуемой sc-конструкции ***/***
else_ **:** ⪡ *идентификатор scp-оператора* ⪢ **;**	**/*** здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора и ненахождения требуемой sc-конструкции ***/***

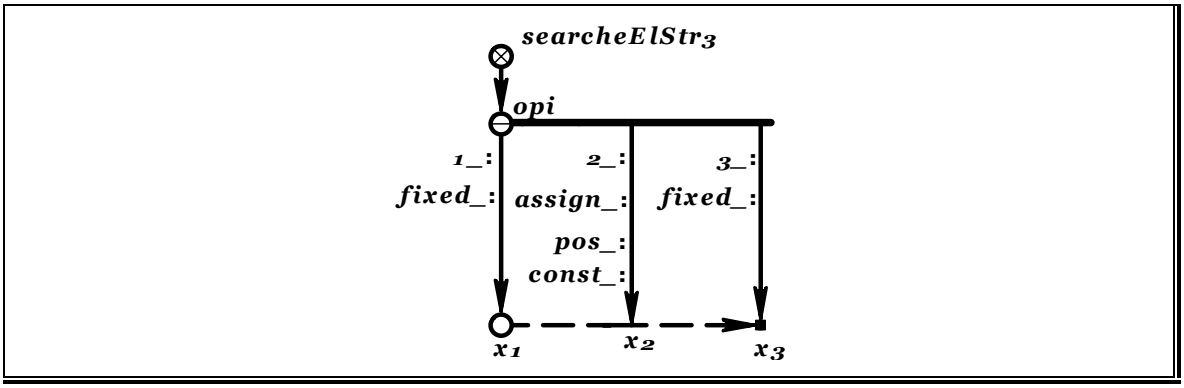
Результаты выполнения операторов типа ***searchElStr3***:

- Если первый операнд помечен атрибутом ***assign_***, то будет искаться sc-узел. Тип искомого sc-узла уточняется следующими атрибутами:
  - ***const_*** - ищется константный sc-узел;
  - ***var_*** - ищется переменный sc-узел;
  - ***meta_*** - ищется метапеременный sc-узел.
- Если второй операнд помечен атрибутом ***assign_***, то будет искаться sc-дуга. Тип искомой sc-дуги уточняется следующими атрибутами:
  - ***pos_*** - ищется позитивная sc-дуга;
  - ***neg_*** - ищется негативная sc-дуга;
  - ***fuz_*** - ищется нечеткая sc-дуга;
  - ***const_, var_, meta_*** – ищется соответственно константная, переменная или метапеременная sc-дуга.
- Если искомая sc-дуга найдена и указан атрибут ***f_***, то эта sc-дуга удаляется.
- Если третий элемент помечен атрибутом ***assign_***, то будет искаться sc-элемент, тип которого уточняется следующими атрибутами:
  - ***node_*** - ищется sc-узел;
  - ***elem_*** - ищется sc-элемент неопределенного типа;
  - ***arc_*** - ищется sc-дуга. В этом случае для уточнения типа могут дополнительно использоваться атрибуты ***pos_***, ***neg_***, ***fuz_*** (семантику смотри выше);
  - ***const_, var_, meta_*** – ищется соответственно константный, переменный или метапеременный sc-элемент.

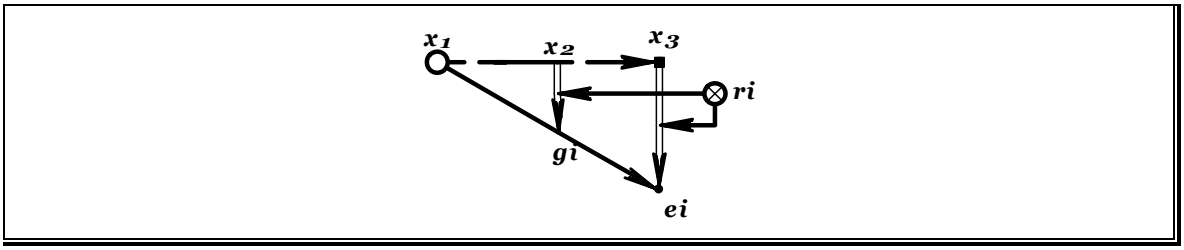
Примечание. Хотя бы один операнд оператора должен быть помечен атрибутом ***fixed_***. Если при поиске будут найдены несколько подходящих sc-конструкций, то выберется одна из них.

Модификации scp-операторов типа ***searchElStr3*** задаются комбинацией атрибутов, которыми помечен операнд. Приведем некоторые модификаций searchElStr3-оператора.

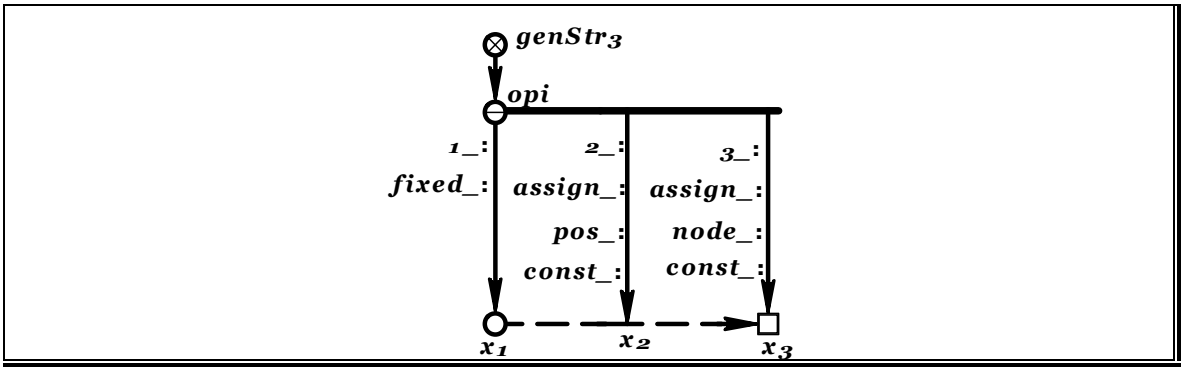
Пример 4.2.4.1



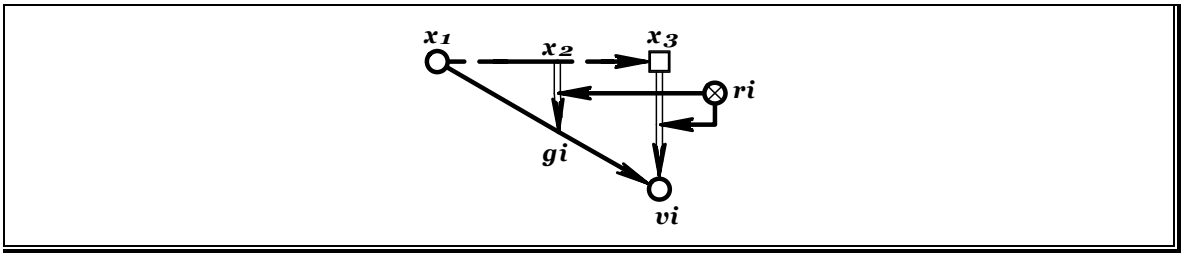
В результате успешного выполнения данного оператора (***opi***) будет найдена константная позитивная sc-дуга (***gi***), соединяющая sc-узел, являющийся значением операнда ***x1***, и sc-элемент (***ei***), являющийся значением операнда ***x3***. Также будет сгенерирована дуга, связывающая операнд ***x2*** с sc-элементом ***gi***, и дуга принадлежности, связывающая указанную выше дугу со знаком соответствующего бинарного отношения (***ri***). Генерация этой дуги означает, что у операнда ***x2*** сформировалось (вычислилось) значение:



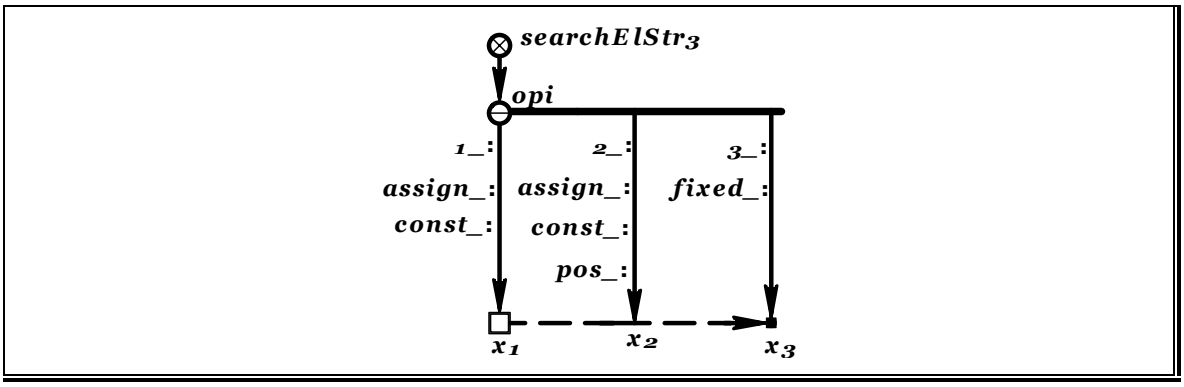
Пример 4.2.4.2



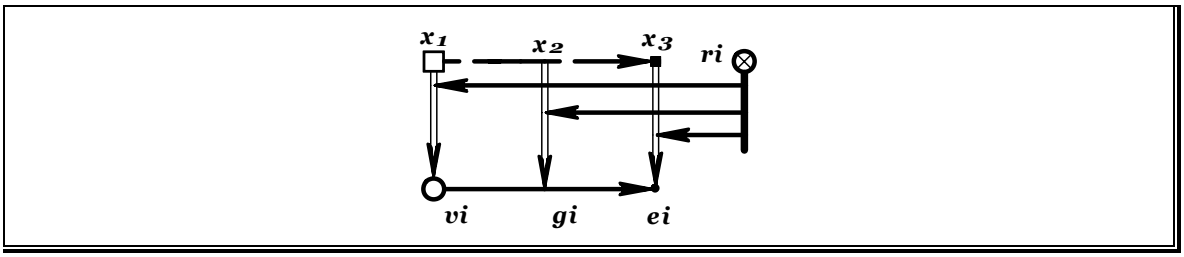
В результате успешного выполнения данного оператора (*opi*) будут найдены константный sc-узел (*vi*) и константная позитивная sc-дуга (*gi*), соединяющая его с sc-узлом, являющимся значением операнда *x1*. Также будут сгенерированы дуга, связывающая операнд *x2* с sc-элементом *gi*, и дуга, связывающая операнд *x3* с sc-элементом *vi*, а также дуги принадлежности, связывающие указанные выше дуги со знаком соответствующего бинарного отношения (*ri*). Генерация этих дуг означает, что у операндов *x2* и *x3* сформировалось (вычислилось) значение:



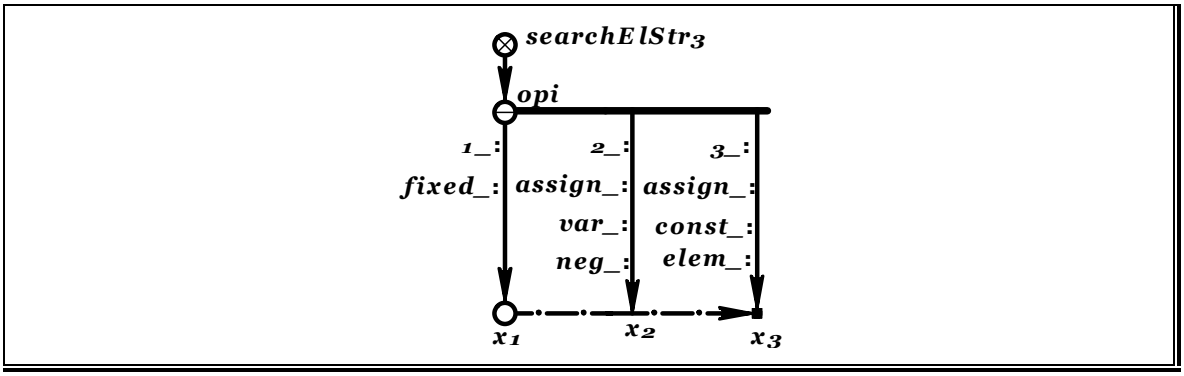
Пример 4.2.4.3



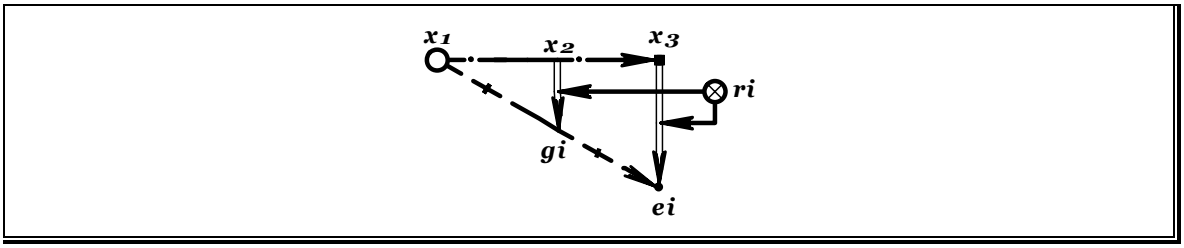
В результате успешного выполнения данного оператора (*opi*) будут найдены константный sc-узел (*vi*) и константная позитивная sc-дуга (*gi*), соединяющая его с sc-элементом (*ei*), являющимся значением операнда *x3*. Также будут сгенерированы дуга, связывающая операнд *x2* с sc-элементом *gi*, и дуга, связывающая операнд *x1* с sc-элементом *vi*, а также дуги принадлежности, связывающие указанные выше дуги со знаком соответствующего бинарного отношения (*ri*). Генерация этих дуг означает, что у операндов *x1* и *x2* сформировалось (вычислилось) значение



Пример 4.2.4.4



В результате успешного выполнения данного оператора (*opi*) будут найдены константный sc-элемент неопределенного типа (*ei*) и переменная негативная sc-дуга (*gi*), соединяющая его с sc-узлом, являющимся значением операнда *x1*. Также будут сгенерированы дуга, связывающая операнд *x2* с sc-элементом *gi*, и дуга, связывающая операнд *x3* с sc-элементом *ei*, а также дуги принадлежности, связывающие указанные выше дуги со знаком соответствующего бинарного отношения (*ri*). Генерация этих дуг означает, что у операндов *x2* и *x3* сформировалось (вычислилось) значение:



Операторы ассоциативного поиска произвольных элементов sc-конструкций, удовлетворяющих заданным условиям и состоящих из трех элементов, и формирования из них множества (searchSetStr3-операторы, операторы типа *searchSetStr3*)

Операторы данного типа осуществляют ассоциативный поиск произвольных элементов sc-конструкций, удовлетворяющих заданным условиям и состоящих из трех элементов, и формирования из них множества.

Запись scp-операторов типа *searchSetStr3* на языке SCs имеет следующий вид:

< *запись searchSetStr3 –оператора* > ::=
searchSetStr3 →
< *идентификатор searchSetStr3-оператора* > →
(* *fixed_* : ‖ /* данный атрибут уточняет, известно (вычислено) ли значение операнда */
[* (* *const_* : ‖ *var_* : ‖ *meta_* : *) *] *) /* данные атрибуты уточняют тип искомого sc-узла по признаку "константа – переменная – метапеременная" */
*1_* : < *операнд x1* > , /* операндом является программная переменная, или если указан атрибут *fixed_*, то операндом может быть и константа программы */
(* *fixed_* : ‖ /* данный атрибут уточняет, известно (вычислено) ли значение операнда */

[* (* ***pos_*** : ‖ ***neg_*** : ‖ ***fuz_*** : *) *]	/* данные атрибуты уточняют тип искомой sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" */
[* (* ***const_*** : ‖ ***var_*** : ‖ ***meta_*** : *) *] *)	/* данные атрибуты уточняют тип искомого sc-элемента по признаку "константа – переменная – метапеременная" */
*2_* : < *операнд x2* > ,	/* операндом является программная переменная, или если указан атрибут ***fixed_***, то операндом может быть и константа программы */
(* ***fixed_*** : ‖	/* данный атрибут уточняет, известно (вычислено) ли значение операнда */
(* [* ***node_*** : ‖ ***elem_*** : ‖ ***arc_*** : *]	/* данные атрибуты уточняют тип искомого sc-элемента по признаку "узел – элемент неопределенного типа – дуга" */
[* (* ***pos_*** : ‖ ***neg_*** : ‖ ***fuz_*** : *) *] *)	/* данные атрибуты уточняют тип искомой sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" */
[* (* ***const_*** : ‖ ***var_*** : ‖ ***meta_*** : *) *] *)	/* данные атрибуты уточняют тип искомого sc-элемента по признаку "константа – переменная – метапеременная" */
*3_* : < *операнд x3* > ,	/* операндом является программная переменная, или если указан атрибут ***fixed_***, то операндом может быть и константа программы */
(* ***fixed_*** : ‖ ***assign_*** : *)	/* данные атрибуты уточняют, известно (вычислено) ли значение операнда или необходимо сгенерировать sc-элемент и установить значение операнда данного оператора */
set1_ : < *операнд xs1* > ,	/* операндом является программная переменная, или если указан атрибут ***fixed_***, то операндом может быть и константа программы. Значением данного операнда становится множество “первых” элементов найденных sc-конструкций */
(* ***fixed_*** : ‖ ***assign_*** : *)	/* данные атрибуты уточняют, известно (вычислено) ли значение операнда или необходимо сгенерировать sc-элемент и установить его значением операнда данного оператора */
set2_ : < *операнд xs2* > ,	/* операндом является программная переменная, или если указан атрибут ***fixed_***, то операндом может быть и константа программы. Значением данного операнда становится множество “вторых” элементов найденных sc-конструкций */
(* ***fixed_*** : ‖ ***assign_*** : *)	/* данные атрибуты уточняют, известно (вычислено) ли значение операнда или необходимо сгенерировать sc-элемент и установить значение операнда данного оператора */

set3_ **:** ⩽ *операнд xs3* ⩾ **,**	**/*** операндом является программная переменная, или если указан атрибут ***fixed_***, то операндом может быть и константа программы. Значением данного операнда становится множество “третьих” элементов найденных sc-конструкций ***/**
then_ **:** ⩽ *идентификатор scp-оператора* ⩾ **;**	**/*** здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора и нахождения требуемой sc-конструкции ***/**
else_ **:** ⩽ *идентификатор scp-оператора* ⩾ **;**	**/*** здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора и ненахождения требуемой sc-конструкции ***/**

Результаты выполнения операторов типа ***searchSetStr3***:

- Если первый операнд не помечен атрибутом ***fixed_***, то будет искаться sc-узел. Тип искомого sc-узла уточняется следующими атрибутами:
  - ***const_*** - ищется константный sc-узел;
  - ***var_*** - ищется переменный sc-узел;
  - ***meta_*** - ищется метапеременный sc-узел.
- Если второй операнд не помечен атрибутом ***fixed_***, то будет искаться sc-дуга. Тип искомой sc-дуги уточняется следующими атрибутами:
  - ***pos_*** - ищется позитивная sc-дуга;
  - ***neg_*** - ищется негативная sc-дуга;
  - ***fuz_*** - ищется нечеткая sc-дуга;
  - ***const_, var_, meta_*** – ищется соответственно константная, переменная или метапеременная sc-дуга.
- Если третий элемент не помечен атрибутом ***fixed_***, то будет искаться sc-элемент, тип которого уточняется следующими атрибутами:
  - ***node_*** - ищется sc-узел;
  - ***elem_*** - ищется sc-элемент неопределенного типа;
  - ***arc_*** - ищется sc-дуга. В этом случае для уточнения типа могут дополнительно использоваться атрибуты ***pos_***, ***neg_***, ***fuz_*** (семантику смотри выше);
  - ***const_, var_, meta_*** – ищется соответственно константный, переменный или метапеременный sc-элемент.

Примечание 1. Хотя бы один операнд из первых трёх операндов должен быть помечен атрибутом ***fixed_***.

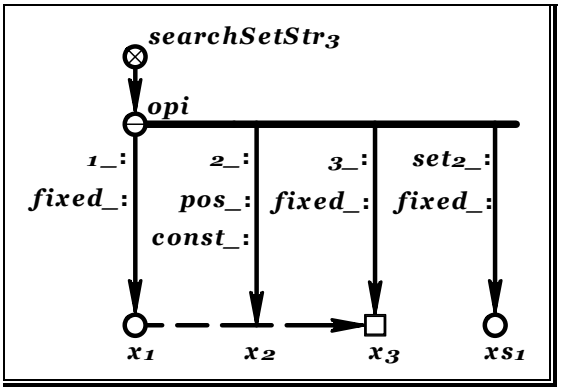
Примечание 2. Последние три операнда (***xs1***, ***xs2*** и ***xs3***) задают множества соответственно “первых”, “вторых” и “третьих” элементов sc-конструкций. Если указан атрибут ***assign_***, то генерируется новое множество, а если указан атрибут ***fixed_***, то множество дополняется найденными sc-конструкциями.

Примечание 3. Операнды ***xs1***, ***xs2*** и ***xs3*** могут быть scp-переменными и scp-константами. Должен присутствовать хотя бы один из операндов ***xs1***, ***xs2*** и ***xs3***.

Примечание 4. Операнды ***xs1***, ***xs2*** и ***xs3*** могут совпадать.

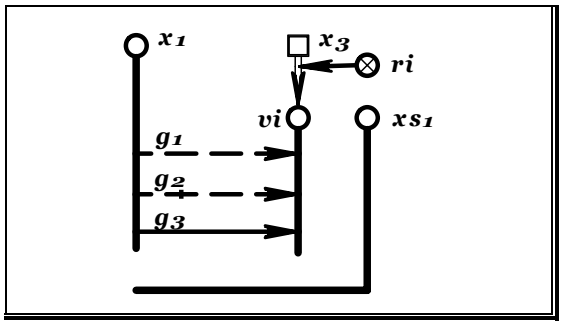
Модификации scp-операторов типа ***searchSetStr3*** задаются комбинацией атрибутов, которыми помечен операнд. Приведем некоторые модификации searchSetStr3-оператора.

Пример 4.2.4.5

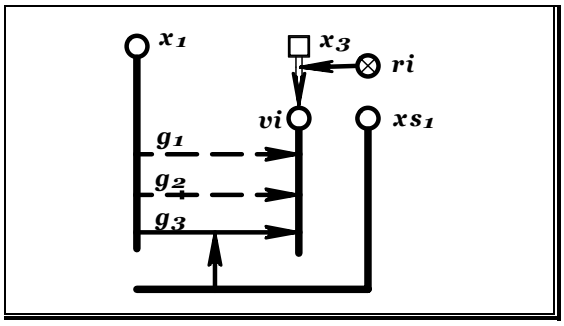


В результате успешного выполнения данного оператора (*opi*) будет найдена константная позитивная sc-дуга (*v1*) и константная позитивная sc-дуга (*gi*), соединяющая его с sc-элементом (*ei*), являющимся значением операнда *x3*. Также будут сгенерированы дуга, связывающая операнд *x2* с sc-элементом *gi*, и дуга, связывающая операнд *x1* с sc-элементом *vi*, а также дуги принадлежности, связывающие указанные выше дуги со знаком соответствующего бинарного отношения (*ri*). Генерация этих дуг означает, что у операндов *x1* и *x2* сформировалось (вычислилось) значение:

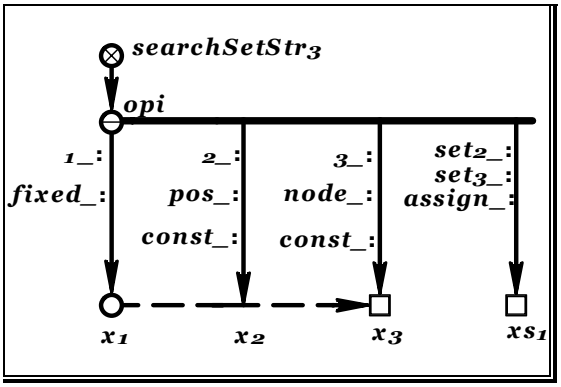
До



После

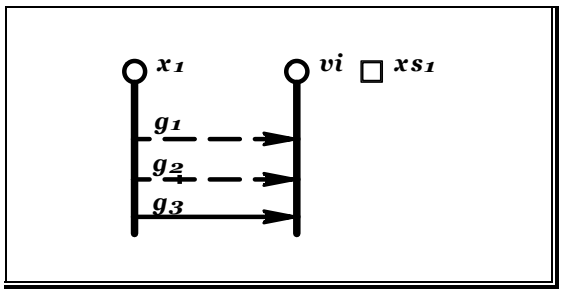


Пример 4.2.4.6

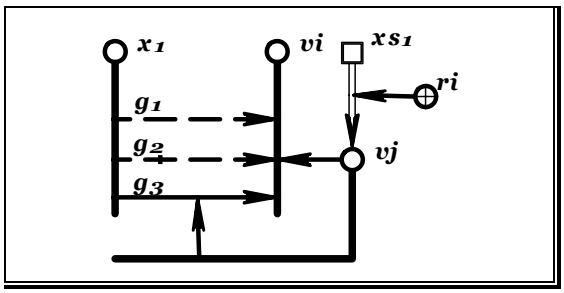


В результате успешного выполнения данного оператора (*opi*) будут найдены константный sc-узел (*vi*) и константная позитивная sc-дуга (*gi*), соединяющая его с sc-элементом (*ei*), являющимся значением операнда *x3*. Также будут сгенерированы дуга, связывающая операнд *x2* с sc-элементом *gi*, и дуга, связывающая операнд *x1* с sc-элементом *vi*, а также дуги принадлежности, связывающие указанные выше дуги со знаком соответствующего бинарного отношения (*ri*). Генерация этих дуг означает, что у операндов *x1* и *x2* сформировалось (вычислилось) значение:

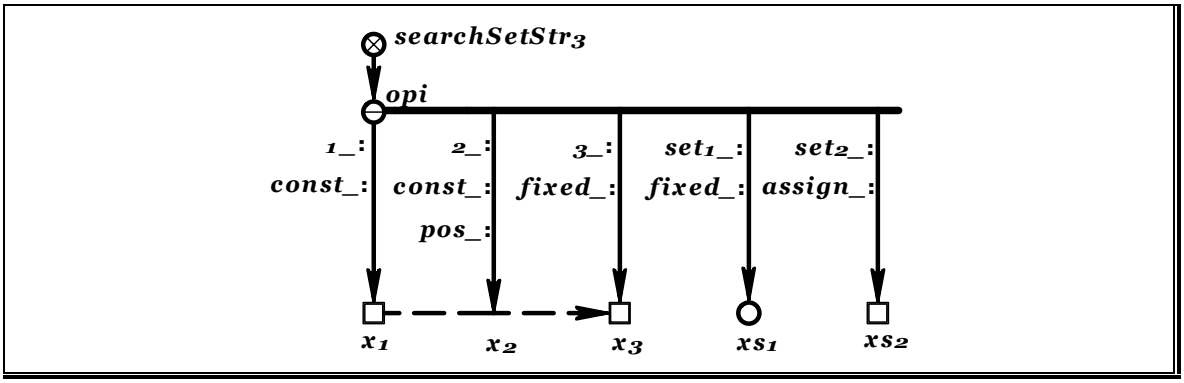
До



После

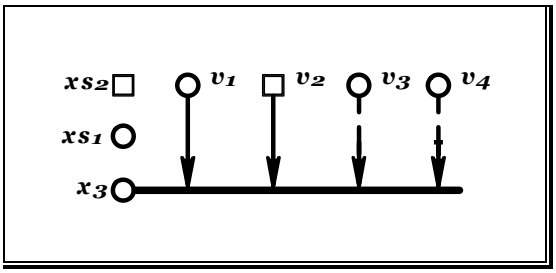


Пример 4.2.4.7

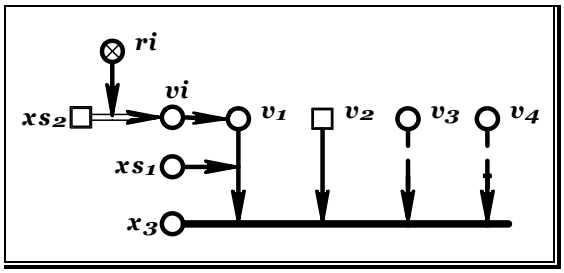


В результате успешного выполнения данного оператора (*opi*) будут найдены константный sc-узел (*vi*) и константная позитивная sc-дуга (*gi*), соединяющая его с sc-элементом (*ei*), являющимся значением операнда *x3*. Также будут сгенерированы дуга, связывающая операнд *x2* с sc-элементом *gi*, и дуга, связывающая операнд *x1* с sc-элементом *vi*, а также дуги принадлежности, связывающие указанные выше дуги со знаком соответствующего бинарного отношения (*ri*). Генерация этих дуг означает, что у операндов *x1* и *x2* сформировалось (вычислилось) значение:

До



После



Операторы ассоциативного поиска произвольных элементов, удовлетворяющих заданным условиям, и формирования из них множества (selectYStr3-операторы, операторы типа *selectYStr3*)

Операторы данного типа осуществляют ассоциативный поиск произвольных элементов sc-конструкций, удовлетворяющих заданным условиям и состоящих из трех элементов, и формирования из них множества, элементами которого остаются только sc-элементы, удовлетворяющие заданным условиям.

Запись scp-операторов типа *selectYStr3* на языке SCs имеет следующий вид:

< запись selectYStr3 –оператора > ::=
selectYStr3 —>
< идентификатор selectYStr3-оператора > —>
(* *fixed_* : ‖ /* данный атрибут уточняет, известно (вычислено) ли значение операнда */
[* (* *const_* : ‖ *var_* : ‖ *meta_* : *) *] *) /* данные атрибуты уточняют тип искомого sc-узла по признаку "константа – переменная – метапеременная" */
*1_* : *< операнд x1 >* , /* операндом является программная переменная, или если указан атрибут *fixed_*, то операндом может быть и константа программы */

(* ***fixed_*** : ‖	/* данный атрибут уточняет, известно (вычислено) ли значение операнда */
[* (* ***pos_*** : ‖ ***neg_*** : ‖ ***fuz_*** : *) *]	/* данные атрибуты уточняют тип искомой sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" */
[* (* ***const_*** : ‖ ***var_*** : ‖ ***meta_*** : *) *] *)	/* данные атрибуты уточняют тип искомого sc-элемента по признаку "константа – переменная – метапеременная" */
*2_* : ⪡ *операнд x2* ⪢ ,	/* операндом является программная переменная, или если указан атрибут ***fixed_***, то операндом может быть и константа программы */
(* ***fixed_*** : ‖	/* данный атрибут уточняет, известно (вычислено) ли значение операнда */
(* [* ***node_*** : ‖ ***elem_*** : ‖ ***arc_*** : *]	/* данные атрибуты уточняют тип искомого sc-элемента по признаку "узел – элемент неопределенного типа – дуга" */
[* (* ***pos_*** : ‖ ***neg_*** : ‖ ***fuz_*** : *) *] *)	/* данные атрибуты уточняют тип искомой sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" */
[* (* ***const_*** : ‖ ***var_*** : ‖ ***meta_*** : *) *] *)	/* данные атрибуты уточняют тип искомого sc-элемента по признаку "константа – переменная – метапеременная" */
*3_* : ⪡ *операнд x3* ⪢ ,	/* операндом является программная переменная, или если указан атрибут ***fixed_***, то операндом может быть и константа программы */
[* ***set1_*** : ⪡ *операнд xs1* ⪢ , *]	/* операнд, значением которого является множество “первых” элементов sc-конструкции */
[* ***set2_*** : ⪡ *операнд xs2* ⪢ , *]	/* операнд, значением которого является множество “вторых” элементов sc-конструкции */
[* ***set3_*** : ⪡ *операнд xs3* ⪢ , *]	/* операнд, значением которого является множество “третьих” элементов sc-конструкции */
then_ : ⪡ *идентификатор scp-оператора* ⪢ ;	/* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора и нахождения требуемой sc-конструкции */
else_ : ⪡ *идентификатор scp-оператора* ⪢ ;	/* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора и ненахождения требуемой sc-конструкции */

Результаты выполнения операторов типа ***selectYStr3***:

- Если первый операнд не помечен атрибутом ***fixed_***, то будет искаться sc-узел. Тип искомого sc-узла уточняется следующими атрибутами:
  - ***const_*** - ищется константный sc-узел;
  - ***var_*** - ищется переменный sc-узел;
  - ***meta_*** - ищется метапеременный sc-узел.
- Если второй операнд не помечен атрибутом ***fixed_***, то будет искаться sc-дуга. Тип искомой sc-дуги уточняется следующими атрибутами:

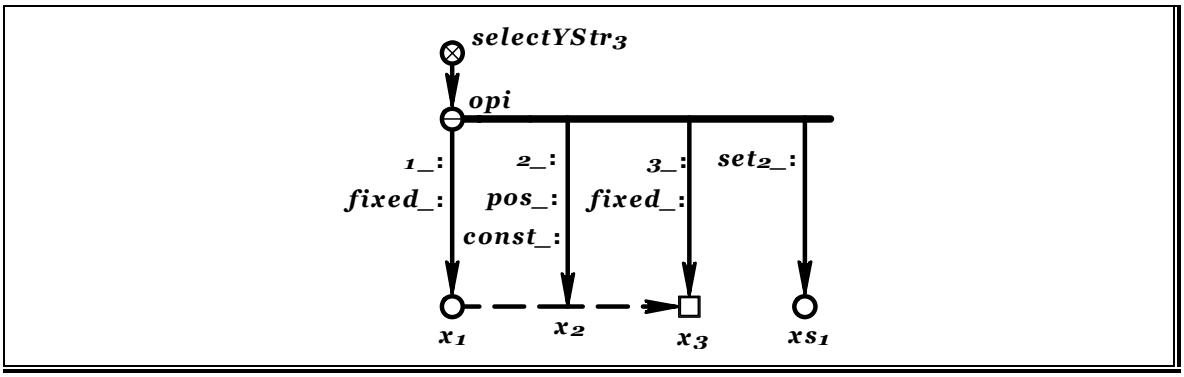
- ***pos_*** - ищется позитивная sc-дуга;
- ***neg_*** - ищется негативная sc-дуга;
- ***fuz_*** - ищется нечеткая sc-дуга;
- ***const_, var_, meta_*** – ищется соответственно константная, переменная или метапеременная sc-дуга.

- Если третий элемент не помечен атрибутом ***fixed_***, то будет искаться sc-элемент, тип которого уточняется следующими атрибутами:
  - ***node_*** - ищется sc-узел;
  - ***elem_*** - ищется sc-элемент неопределенного типа;
  - ***arc_*** - ищется sc-дуга. В этом случае для уточнения типа могут быть дополнительно использоваться атрибуты ***pos_***, ***neg_***, ***fuz_*** (семантику смотри выше);
  - ***const_, var_, meta_*** – ищется соответственно константный, переменный или метапеременный sc-элемент.

Примечание. Хотя бы один операнд из первых трёх операндов должен быть помечен атрибутом ***fixed_***. Последние три операнда (***xs1***, ***xs2*** и ***xs3***) задают множества соответственно “первых”, “вторых” и “третьих” элементов sc-конструкций. Значение этих операндов должно быть вычислено. В этих множествах остаются только те sc-конструкции, которые удовлетворяют заданному условию. Условие задается атрибутами на операндах ***x1***, ***x2*** и ***x3*** (см. выше). Операнды ***xs1***, ***xs2*** и ***xs3*** могут быть scp-переменными и scp-константами. Должен присутствовать хотя бы один из операндов ***xs1***, ***xs2*** и ***xs3***. Операнды ***xs1***, ***xs2*** и ***xs3*** могут совпадать.

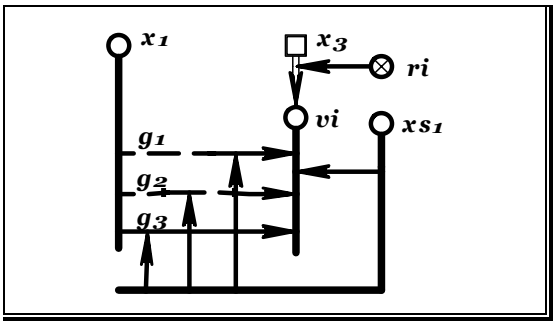
Модификации scp-операторов типа ***selectYStr3*** задаются комбинацией атрибутов, которыми помечен операнд. Приведем некоторые модификации selectYStr3-оператора.

Пример 4.2.4.8

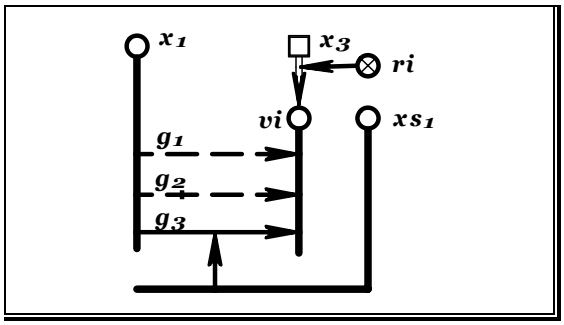


В результате успешного выполнения данного оператора (***opi***) во множестве вторых элементов ***xs1*** будет найдена константная позитивная sc-дуга (***v3***), значение операнда ***x1*** и значение операнда ***x3***- sc-узел (***vi***). Все элементы, кроме найденной sc-дуги (***v3***), будут удалены из множества ***xs1***.

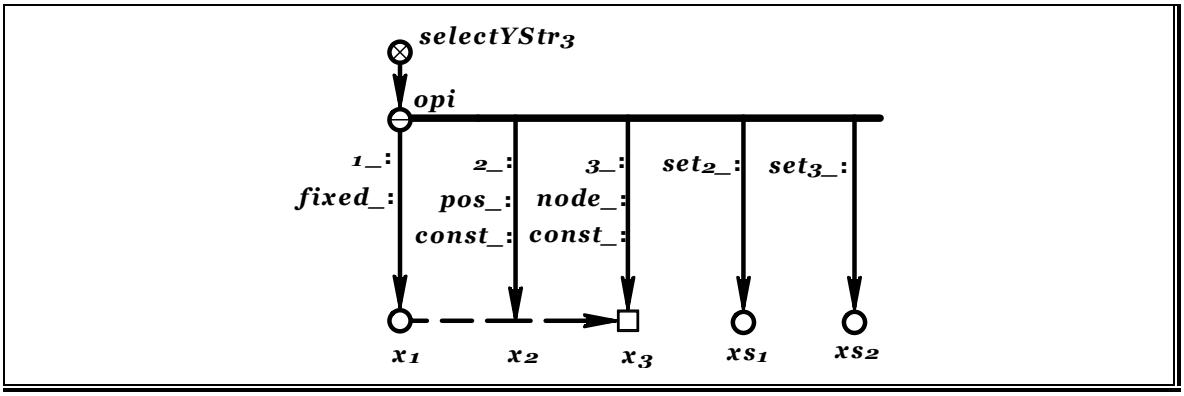
До



После

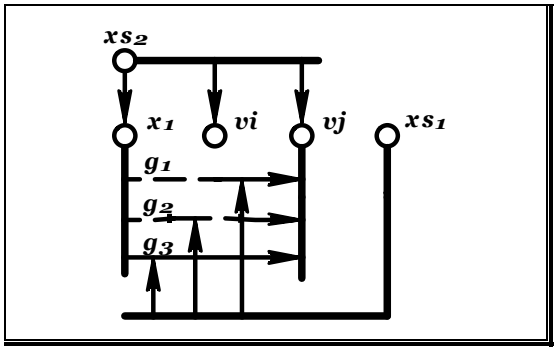


Пример 4.2.4.9

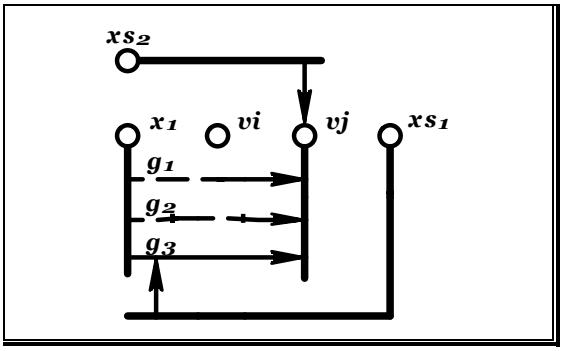


В результате успешного выполнения данного оператора (*opi*) будут найдены константный sc-узел (*vj*) и константная позитивная sc-дуга (*g3*), соединяющая его со значением операнда *x1*, причем найденный sc-узел (*vj*) принадлежит множеству третьих элементов *xs1*, а найденная sc-дуга (*g3*) – множеству вторых элементов *xs2*. Все остальные элементы удаляются из этих множеств.

До



После



Операторы ассоциативного поиска произвольных элементов, не удовлетворяющих заданным условиям, и формирования из них множества (selectNStr3-операторы, операторы типа *selectNStr3*)

Операторы данного типа осуществляют ассоциативный поиск произвольных элементов sc-конструкций, удовлетворяющих заданным условиям и состоящих из трех элементов, и формирования из них множества, элементами которого остаются только sc-элементы, не удовлетворяющие заданным условиям.

Запись scp-операторов типа *selectNStr3* на языке SCs имеет следующий вид:

< запись selectNStr3 –оператора > **:=**

selectNStr3 **→**

< идентификатор selectNStr3-оператора > **→**

(* ***fixed_*** **:** **‖**

[* **(*** ***const_*** **:** **‖** ***var_*** **:** **‖** ***meta_*** **:** ***)** ***]** ***)**

*1_* **:** *< операнд x1 >* **,**

/* данный атрибут уточняет, известно (вычислено) ли значение операнда ***/**

/* данные атрибуты уточняют тип искомого sc-узла по признаку "константа – переменная – метапеременная" ***/**

/* операндом является программная переменная, или если указан атрибут ***fixed_***, то операндом может быть и константа программы ***/**

(* ***fixed_*** : ‖	/* данный атрибут уточняет, известно (вычислено) ли значение операнда */
[* (* ***pos_*** : ‖ ***neg_*** : ‖ ***fuz_*** : *) *]	/* данные атрибуты уточняют тип искомой sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" */
[* (* ***const_*** : ‖ ***var_*** : ‖ ***meta_*** : *) *] *)	/* данные атрибуты уточняют тип искомого sc-элемента по признаку "константа – переменная – метапеременная" */
*2_* : ⪡ *операнд x2* ⪢ ,	/* операндом является программная переменная, или если указан атрибут ***fixed_***, то операндом может быть и константа программы */
(* ***fixed_*** : ‖	/* данный атрибут уточняет, известно (вычислено) ли значение операнда */
(* [* ***node_*** : ‖ ***elem_*** : ‖ ***arc_*** : *]	/* данные атрибуты уточняют тип искомого sc-элемента по признаку "узел – элемент неопределенного типа – дуга" */
[* (* ***pos_*** : ‖ ***neg_*** : ‖ ***fuz_*** : *) *] *)	/* данные атрибуты уточняют тип искомой sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" */
[* (* ***const_*** : ‖ ***var_*** : ‖ ***meta_*** : *) *] *)	/* данные атрибуты уточняют тип искомого sc-элемента по признаку "константа – переменная – метапеременная" */
*3_* : ⪡ *операнд x3* ⪢ ,	/* операндом является программная переменная, или если указан атрибут ***fixed_***, то операндом может быть и константа программы */
[* ***set1_*** : ⪡ *операнд xs1* ⪢ , *]	/* операнд, значением которого является множество “первых” элементов sc-конструкции */
[* ***set2_*** : ⪡ *операнд xs2* ⪢ , *]	/* операнд, значением которого является множество “вторых” элементов sc-конструкции */
[* ***set3_*** : ⪡ *операнд xs3* ⪢ , *]	/* операнд, значением которого является множество “третьих” элементов sc-конструкции */
then_ : ⪡ *идентификатор scp-оператора* ⪢ ;	/* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора и нахождения требуемой sc-конструкции */
else_ : ⪡ *идентификатор scp-оператора* ⪢ ;	/* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора и ненахождения требуемой sc-конструкции */

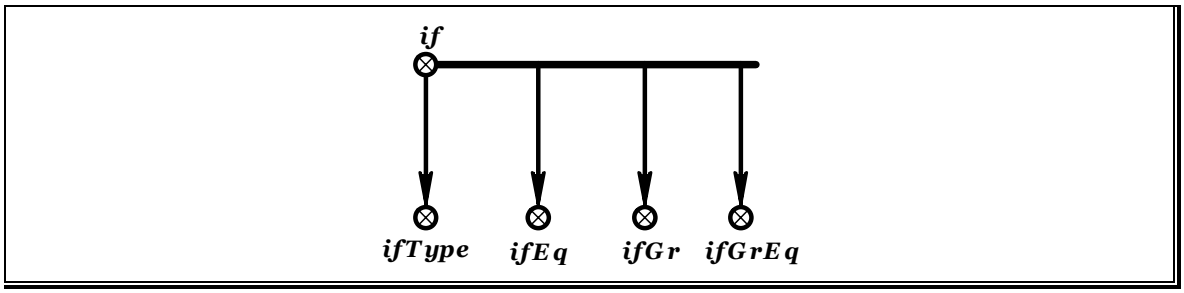
Результаты выполнения операторов типа ***selectNStr3*** полностью аналогичны результатам выполнения операторов типа ***selectYStr3***, за исключением того, что во множествах, которые являются значениями операндов ***xs1*** – ***xs3***, остаются только те sc-конструкции, которые не удовлетворяют заданному условию.

4.2.5. Операторы проверки условий

Семейство scp-операторов проверки условий (такие операторы будем также называть if-операторами) разбивается на следующие типы scp-операторов:

- операторы проверки типа sc-элемента (ifType-операторы);
- операторы проверки равенства значений двух операндов (ifEq-операторы);
- операторы проверки строгого неравенства значений двух операндов (ifGr-операторы);
- операторы проверки неравенства значений двух операндов (ifGrEq-операторы).

Множество ifType-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор ***"ifType"***. Множество ifEq-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор ***"ifEq"***. Множество ifGr-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор ***"ifGr"***. Множество ifGrEq-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор ***"ifGrEq"***. Семейство типов if-операторов в языке SCP обозначается ключевым узлом с идентификатором ***"if"***. Соотношение между указанными ключевыми узлами задается следующей sc-конструкцией:



Перейдем к более подробному рассмотрению указанных выше типов scp-операторов.

Операторы проверки типа sc-элемента (ifType-операторы, операторы типа *ifType*)

Операторы данного типа осуществляют проверку типа sc-элемента. Тип проверяется по признаку "константа – переменная – метапеременная", "узел – элемент неопределенного типа – дуга", "позитивная дуга – негативная дуга – нечеткая дуга", а также проверяется, вычислено ли значение переменной.

Запись scp-операторов типа *ifType* на языке SCs имеет следующий вид:

< запись ifType –оператора > **::=**

ifType **—>**

< идентификатор ifType-оператора > **—>**

(* **[*** ***node_*** **:** **||** ***elem_*** **:** **||** ***arc_*** **:** ***]** — **/*** данные атрибуты уточняют тип sc-элемента по признаку "узел – элемент неопределенного типа – дуга" ***/**

[* **(*** ***pos_*** **:** **||** ***neg_*** **:** **||** ***fuz_*** **:** ***)** ***]** ***)** — **/*** данные атрибуты уточняют тип sc-дуги по признаку "позитивная дуга – негативная дуга – нечеткая дуга" ***/**

[* **(*** ***const_*** **:** **||** ***var_*** **:** **||** ***meta_*** **:** ***)** ***]** ***)** — **/*** данные атрибуты уточняют тип sc-элемента по признаку "константа – переменная – метапеременная" ***/**

< операнд x1 > **,** — **/*** операндом является scp-переменная или scp-константа ***/**

then_ **:** ⪡ *идентификатор scp-оператора* ⪢ **;**

/* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора и если значение операнда удовлетворяет условию, заданному атрибутами ***/**

else_ **:** ⪡ *идентификатор scp-оператора* ⪢ **;**

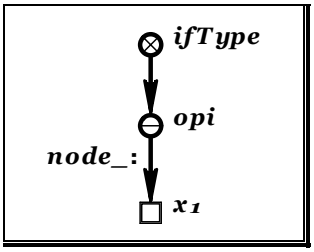
/* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора и если значение операнда не удовлетворяет условию, заданному атрибутами ***/**

Результаты выполнения операторов типа ***ifType***:

- Если значение операнда не вычислено, то управление передается scp-оператору, помеченному атрибутом ***else_*** .
- Проверяется значение операнда, тип значения уточняется следующими атрибутами:
  - ***node_*** , ***elem_*** , ***arc_*** – соответственно sc-узел, sc-элемент неопределенного типа или sc-дуга;
  - ***pos_*** , ***neg_*** , ***fuz_*** – соответственно позитивная, негативная или нечеткая sc-дуга;
  - ***const_*** , ***var_*** , ***meta_*** – соответственно константный, переменный или метапеременный sc-элемент.
- Если тип значения операнда соответствует заданному типу, то управление после реализации данного scp-оператора передается scp-оператору, помеченному атрибутом ***then_***, иначе управление передается scp-оператору, помеченному атрибутом ***else_*** .

Модификации scp-операторов типа ***ifType*** задаются комбинацией атрибутов, которыми помечен операнд. Приведем пример модификации ifType-оператора.

Пример 4.2.5.1



Если значением операнда ***x1*** является sc-узел, то проверяемое условие считается истинным, иначе условие ложно.

Операторы проверки равенства значений двух sc-элементов (ifEq-операторы, операторы типа *ifEq*)

Операторы данного типа осуществляют проверку равенства значений двух sc-элементов.

Запись scp-операторов типа ***ifEq*** на языке SCs имеет следующий вид:

⪡ *запись ifEq –оператора* ⪢ ::=

ifEq ⟶

⪡ *идентификатор ifEq-оператора* ⪢ ⟶

*1_* **:** ⪡ *операнд x1* ⪢ **,**

/* операндом является scp-переменная или scp-константа ***/**

*2_* **:** ⪡ *операнд x2* ⪢ **,**

/* операндом является scp-переменная или scp-константа ***/**

then_ **:** ⪡ *идентификатор scp-оператора* ⪢ **;** /* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-операторaи, если значения операндов равны */

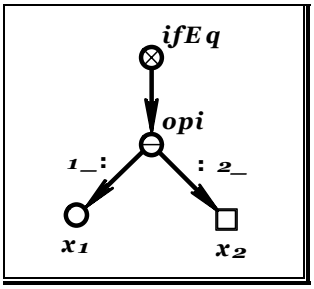
else_ **:** ⪡ *идентификатор scp-оператора* ⪢ **;** /* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора, если значения операндов не равны или не вычислены */

Результаты выполнения операторов типа ***ifEq***:

- Если значения операндов не вычислены, то управление передается scp-оператору, помеченному атрибутом ***else_*** .
- Если значение первого операнда равно значению второго операнда, то управление передается scp-оператору, помеченному атрибутом ***then_***, иначе управление передается scp-оператору, помеченному атрибутом ***else_*** .

Существует одна модификация scp-операторов типа ***ifEq*** Приведем пример этой модификации:

Пример 4.2.5.2



Если значением операнда ***x2*** равно значению операнда ***x1*** (в данном случае это сам ***x1***), то условие считается истинным, иначе условие ложно.

Операторы проверки строгого неравенства значений двух sc-элементов (ifGr-операторы, операторы типа *ifGr*)

Операторы данного типа осуществляют проверку значений двух sc-элементов по признаку “больше”.

Запись scp-операторов типа ***ifGr*** на языке SCs имеет следующий вид:

⪡ *запись ifGr–оператора* ⪢ **::=**

ifGr **⟶**

⪡ *идентификатор ifGr-оператора* ⪢ **⟶**

*1_* **:** ⪡ *операнд x1* ⪢ **,** /* операндом является scp-переменная или scp-константа */

*2_* **:** ⪡ *операнд x2* ⪢ **,** /* операндом является scp-переменная или scp-константа */

then_ **:** ⪡ *идентификатор scp-оператора* ⪢ **;** /* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора, если значение первого операнда больше значения второго операнда */

else_ **:** ⪡ *идентификатор scp-оператора* ⪢ **;** /* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора, если значение первого операнда не больше значения второго операнда или значения операндов не вычислены */

Результаты выполнения операторов типа ***ifGr***:

- Если значения операндов не вычислены, то управление передается scp-оператору, помеченному атрибутом ***else_*** .
- Если значение первого операнда больше значения второго операнда, то управление передается scp-оператору, помеченному атрибутом ***then_***, иначе управление передается scp-оператору, помеченному атрибутом ***else_*** .

Существует одна модификация scp-операторов типа ***ifGr*** (аналогично ifEq-оператору).

Операторы проверки неравенства значений двух операндов (ifGrEq-операторы, операторы типа ***ifGrEq***)

Операторы данного типа осуществляют проверку значений двух sc-элементов по признаку “больше или равно”.

Запись scp-операторов типа ***ifGrEq*** на языке SCs имеет следующий вид:

< запись ifGrEq–оператора > **::=**

ifGrEq **→**

< идентификатор ifGrEq-оператора > **→**

*1_* **:** *< операнд x1 >* **,** — **/*** операндом является scp-переменная или scp-константа ***/**

*2_* **:** *< операнд x2 >* **,** — **/*** операндом является scp-переменная или scp-константа ***/**

then_ **:** *< идентификатор scp-оператора >* **;** — **/*** здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора, если значение первого операнда больше или равно значению второго операнда ***/**

else_ **:** *< идентификатор scp-оператора >* **;** — **/*** здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора, если значение первого операнда не больше и не равно значению второго операнда или значения операндов не вычислены ***/**

Результаты выполнения операторов типа ***ifGrEq***:

- Если значения операндов не вычислены, то управление передается scp-оператору, помеченному атрибутом ***else_*** .
- Если значение первого операнда больше или равно значению второго операнда, то управление передается scp-оператору, помеченному атрибутом ***then_***, иначе управление передается scp-оператору, помеченному атрибутом ***else_*** .

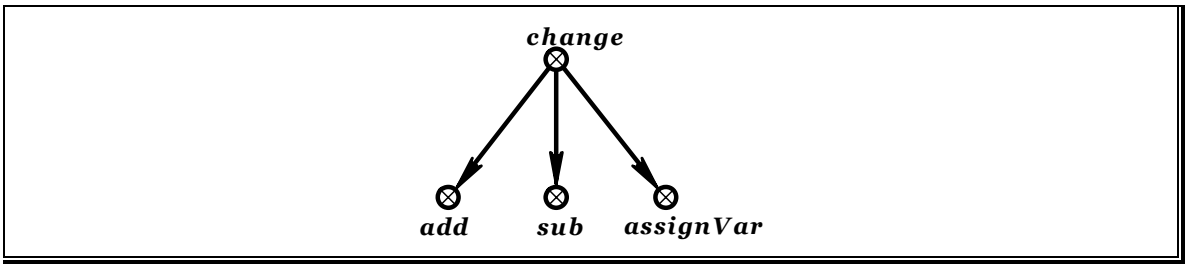
Существует одна модификация scp-операторов типа ***ifGrEq*** (аналогично ifEq-оператору).

4.2.6. Операторы изменения свойств элемента конструкций языка SC

Семейство scp-операторов изменения свойств sc-элемента (такие операторы будем также называть change-операторами) разбивается на следующие типы scp-операторов:

- операторы сложения содержимых sc-элементов (add-операторы);
- операторы вычитания содержимых sc-элементов (sub-операторы);
- операторы пересылки значения программной переменной (assign-операторы).

Множество add-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор **"*add*"**. Множество sub-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор **"*sub*"**. Множество assign-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор **"*assign*"**. Семейство типов change-операторов в языке SCP обозначается ключевым узлом с идентификатором **"*change*"**. Соотношение между указанными ключевыми узлами задается следующей sc-конструкцией:



Перейдем к более подробному рассмотрению указанных выше типов scp-операторов.

Операторы сложения содержимых sc-элементов (add-операторы, операторы типа *add*)

Операторы данного типа осуществляют сложение содержимых двух sc-узлов.

Запись scp-операторов типа ***add*** на языке SCs имеет следующий вид:

< запись add–оператора > **:=**

add **→**

< идентификатор add-оператора > **→**

*1_* **:** *< операнд x1 >* **,** **/*** операндом является scp-переменная или scp-константа ***/**

*2_* **:** *< операнд x2 >* **,** **/*** операндом является scp-переменная или scp-константа ***/**

*3_* **:** *< операнд x3 >* **,** **/*** операндом является scp-переменная или scp-константа ***/**

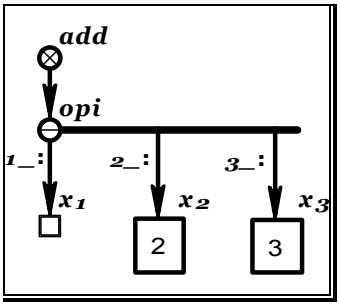
goto_ **:** *< идентификатор scp-оператора >* **;** **/*** здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора ***/**

Результаты выполнения операторов типа ***add***:

- Если значение первого операнда не вычислено (т.е. первый операнд является scp-переменной), то генерируется sc-узел, который становится значением первого операнда. В содержимое сгенерированного sc-узла записывается сумма содержимых значений двух других операндов.
- Если значение первого операнда вычислено, то содержимое формируется аналогичным образом (см. выше).

Существует одна модификация scp-операторов типа ***add***. Приведем пример этой модификации:

Пример 4.2.6.1



В результате выполнения оператора ***opi*** сгенерируется sc-узел с содержимым, равным числу 5, и этот узел станет значением операнда ***x1***.

Операторы вычитания содержимых sc-элементов (sub-операторы, операторы типа *sub*)

Операторы данного типа осуществляют вычитание числового содержимого одного sc-узла из другого.

Запись scp-операторов типа ***sub*** на языке SCs имеет следующий вид:

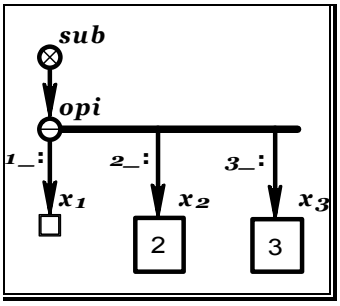
< запись sub–оператора > ::=
sub →
< идентификатор sub-оператора > →
*1_* : *< операнд x1 >* , /* операндом является scp-переменная или scp-константа */
*2_* : *< операнд x2 >* , /* операндом является scp-переменная или scp-константа */
*3_* : *< операнд x3 >* , /* операндом является scp-переменная или scp-константа */
goto_ : *< идентификатор scp-оператора >* ; /* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора */

Результаты выполнения операторов типа ***sub***:

- Если значение первого операнда не вычислено (т.е. первый операнд является scp-переменной), то генерируется sc-узел, который становится значением первого операнда. В содержимое сгенерированного sc-узла записывается разность содержимых значений второго операнда и третьего.
- Если значение первого операнда вычислено, то содержимое формируется аналогичным образом (см. выше).

Существует одна модификация scp-операторов типа ***sub***. Приведем пример модификации.

Пример 4.2.6.2



В результате выполнения оператора ***opi*** сгенерируется sc-узел с содержимым, равным числу -1, и этот узел станет значением операнда ***x1***.

Операторы пересылки значения программной переменной (assignVar-операторы, операторы типа *assignVar*)

Операторы данного типа осуществляют присваивание значения одной scp-переменной или scp-константы другой scp-переменой.

Запись scp-операторов типа ***assignVar*** на языке SCs имеет следующий вид:

< запись assignVar–оператора > **:=**
assignVar **→**
< идентификатор assignVar-оператора > **→**
*1_* **:** *< операнд x1 >* **,** /* операндом является scp-переменная */
*2_* **:** *< операнд x2 >* **,** /* операндом является scp-переменная или scp-константа */
goto_ **:** *< идентификатор scp-оператора >* **;** /* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора */

Результаты выполнения операторов типа ***assignVar***:

- Если второй операнд является scp-переменной, то значением первого операнда становится sc-элемент, являющийся значением второго операнда.
- Если второй операнд является scp-константой, то значением первого операнда становится sc-элемент, обозначающий второй операнд.

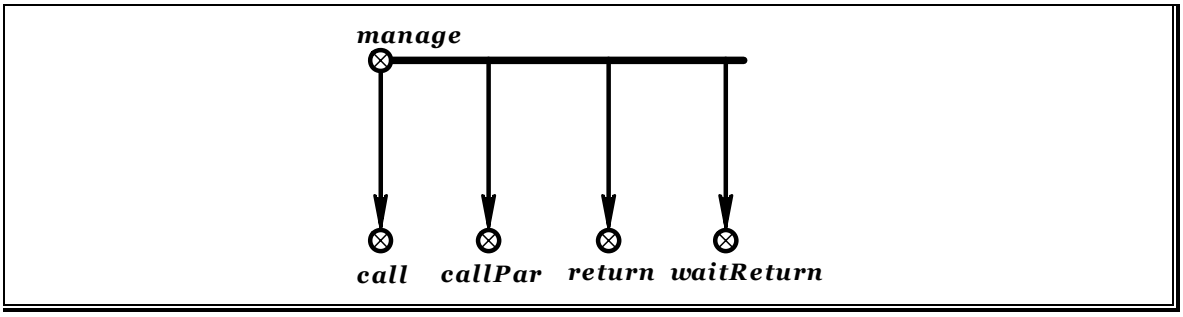
Существует одна модификация scp-операторов типа ***assignVar***.

4.2.7. Операторы управления scp-процессами

Семейство scp-операторов управления scp-процессами (такие операторы будем также называть manage-операторами) разбивается на следующие типы scp-операторов:

- операторы порождения scp-процесса (call-операторы);
- операторы порождения параллельных scp-процессов (callPar-операторы);
- операторы завершения scp-процесса (return-операторы);
- операторы ожидания завершения scp-процесса (waitReturn-операторы).

Множество call-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор **"*call*"**. Множество callPar-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор **"*callPar*"**. Множество return-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор **"*return*"**. Множество waitReturn-операторов в языке SCP будем обозначать ключевым узлом, которому поставим в соответствие идентификатор **"*waitReturn*"**. Семейство типов manage-операторов в языке SCP обозначается ключевым узлом с идентификатором **"*manage*"**. Соотношение между указанными ключевыми узлами задается следующей sc-конструкцией:



Перейдем к более подробному рассмотрению указанных выше типов scp-операторов.

Операторы порождения scp-процесса (call-операторы, операторы типа *call*)

Операторы данного типа осуществляют порождение scp-процесса для выполнения указанной scp-программы.

Запись scp-операторов типа ***call*** на языке SCs имеет следующий вид:

< запись call–оператора > **::=**

call **—>**

< идентификатор call-оператора > **—>**

*1_* **:** *< операнд x1 >* **,** — **/*** значением операнда является sc-узел вызываемой программы. Операнд может быть scp-переменной или scp-константой ***/**

*2_* **:** *< операнд x2 >* **,** — **/*** значением операнда является множество параметров, передаваемых вызываемой программе. Операнд может быть scp-переменной или scp-константой ***/**

*3_* **:** *< операнд x3 >* **,** — **/*** значением операнда после успешной реализации данного scp-оперантора становится sc-узел, обозначающий scp-процесс, выполняющий указанную scp-программу. Операнд должен быть scp-переменной ***/**

goto_ **:** *< идентификатор scp-оператора >* **;** — **/*** здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора ***/**

Результаты выполнения операторов типа ***call***:

- Создается scp-процесс, который будет выполнять scp-программу, обозначенную sc-узлом, являющимся значением первого операнда.
- Второй операнд является множеством параметров, передаваемых в scp-программу, а также множеством возвращаемых значений из этой scp-программы (см. пункт 4.2.1).
- SC-узел, обозначающий созданный scp-процесс, становится значением третьего операнда.

Примечание. Для получения значений из вызываемой программы необходимо после call-оператора поставить waitReturn-оператор.

Существует одна модификация scp-операторов типа ***call***.

Операторы порождения параллельных scp-процессов (callPar-операторы, операторы типа *callPar*)

Операторы данного типа осуществляют порождение параллельных scp-процессов.

Запись scp-операторов типа ***callPar*** на языке SCs имеет следующий вид:

< запись callPar–оператора > **::=**

callPar **—>**

< идентификатор callPar-оператора > **—>**

*1_* **:** *< операнд x1 >* **,** — **/*** значением операнда является sc-узел вызываемой программы. Операнд может быть scp-переменной или scp-константой ***/**

*2_* **:** *< операнд x2 >* **,** — **/*** значением операнда является множество, по которому происходит распараллеливание. Операнд может быть scp-переменной или scp-константой ***/**

*3_* **:** *< операнд x3 >* **,** — **/*** значением операнда является множество параметров, передаваемых вызываемой программе. Операнд может быть

scp-переменной или scp-константой */

goto_ **:** < *идентификатор scp-оператора* > **;**

/* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора */

Результаты выполнения операторов типа ***callPar***:

- Создается множество scp-процессов, которые будут выполнять scp-программу, обозначенную sc-узлом, являющимся значением первого операнда. Количество создаваемых scp-процессов зависит от мощности множества, которое является значением второго операнда.
- Третий операнд является множеством параметров, передаваемых в scp-программу, а также множеством возвращаемых значений из этой scp-программы (см. 4.2.1). В этом множестве должен отсутствовать элемент с атрибутом *1_*. Вместо этого элемента в вызываемую программу будут передаваться элементы из второго множества, которое является значением второго операнда. В каждую создаваемую scp-программу будут передаваться различные элементы.

Существует одна модификация scp-операторов типа ***callPar***.

Операторы завершения scp-программы (return-операторы, операторы типа *return*)

Операторы данного типа осуществляют завершение scp-программы.

Запись scp-операторов типа ***return*** на языке SCs имеет следующий вид:

< *запись return–оператора* > ::=
return —>
< *идентификатор return-оператора* > —>

Результат выполнения операторов типа ***return***:

- Завершается выполнение scp-программы. Вызвавшей scp-программе передаются возвращаемые значения.

Существует одна модификация scp-операторов типа ***return***.

Операторы ожидания завершения scp-процесса (waitReturn-операторы, операторы типа *waitReturn*)

Операторы данного типа осуществляют ожидание завершения scp-процесса.

Запись scp-операторов типа ***waitReturn*** на языке SCs имеет следующий вид:

< *запись waitReturn–оператора* > ::=
waitReturn —>
< *идентификатор waitReturn-оператора* > —>
[* ***process_*** **:** < *операнд* > **,** *]

/* значением операнда является sc-узел, обозначающий scp-процесс, завершение которого необходимо ожидать. Операндом должна быть scp-переменная */

goto_ **:** < *идентификатор scp-оператора* > **;**

/* здесь указывается идентификатор того scp-оператора, которому передается управление после реализации данного scp-оператора */

Результаты выполнения операторов типа ***waitReturn***:

- Если есть операнд с атрибутом ***process_***, то происходит ожидание завершения scp-процесса, обозначенного sc-узлом, являющимся значением данного операнда.
- Если операнда с атрибутом ***process_*** нет, то происходит ожидание завершения всех дочерних scp-процессов данного scp-процесса.

Существуют две модификации scp-операторов типа ***waitReturn***.

4.3. Абстрактная графодинамическая ассоциативная машина реализации языка SCP

В данном подразделе рассматривается абстрактная машина, определяющая операционную семантику языка SCP, т.е. являющаяся формальным описанием интерпретатора языка SCP. Указанную абстрактную машину будем сокращенно называть абстрактной scp-машиной. В соответствии с назначением языка SCP абстрактная scp-машина предназначена для эффективной интерпретации всевозможных графодинамических моделей.

Перечислим общие принципы организации переработки информации в абстрактной scp-машине:

- абстрактная scp-машина принадлежит к семейству абстрактных sc-машин, т.е. к семейству графодинамических абстрактных машин, в памяти которых информация представляется в виде sc-конструкций;
- абстрактная scp-машина относится к классу параллельных асинхронных sc-машин, в которых выполняемые параллельные процессы работают над общедоступной памятью;
- абстрактная scp-машина относится к классу абстрактных sc-машин, обеспечивающих реализацию (интерпретацию) хранимых в ее памяти процедурных программ – программ, представленных на графовом языке SCP.

4.3.1. Информационные конструкции, описывающие состояние абстрактной scp-машины

Основными видами информации, хранящейся в памяти абстрактной scp-машины, являются:

- реализуемые (интерпретируемые) scp-программы;
- sc-конструкции, являющиеся данными реализуемых scp-программ, т.е. такие sc-конструкции, переработка которых описывается этими scp-программами;
- sc-конструкции, являющиеся вспомогательными (специальными) данными, определяемыми и формируемыми самой scp-машиной, т.е. данными, которые используются только микропрограммами абстрактной scp-машины.

Отметим при этом, что граница между scp-программами и их данными относительна, т.к. данными для scp-программы могут быть другие scp-программы, или даже она сама. Доступ scp-программ к вспомогательным данным абстрактной scp-машины блокируется. К этим данным имеют доступ только микропрограммы абстрактной scp-машины.

Для представления информационных конструкций, описывающих состояние абстрактной scp-машины, дополнительно вводятся следующие ключевые узлы:

- ***demon*** – ключевой sc-узел, обозначающий множество демонических (самоинициируемых) программ;
- ***init*** – ключевой sc-узел, обозначающий множество инициируемых программ;
- ***processSCP*** – ключевой sc-узел, обозначающий множество scp-процессов;
- ***varValue_*** – ключевой sc-узел, обозначающий множество значений, вычисленных в рамках заданного scp-процесса;
- ***blkG_*** – ключевой sc-узел, обозначающий множество g-блокированных sc-элементов в рамках заданного scp-процесса;
- ***blkS_*** – ключевой sc-узел, обозначающий множество s-блокированных sc-элементов в рамках заданного scp-процесса;
- ***blkGE_*** – ключевой sc-узел, обозначающий множество ge-блокированных sc-элементов в рамках заданного scp-процесса;

- ***blkSE_*** – ключевой sc-узел, обозначающий множество se-блокированных sc-элементов в рамках заданного scp-процесса;
- ***program_*** – ключевой sc-узел, указывающий на scp-программу, которая выполняется в рамках заданного scp-процесса;
- ***active_*** – ключевой sc-узел, указывающий на scp-оператор, который выполняется в рамках заданного scp-процесса;
- ***readyState*** – ключевой sc-узел, обозначающий множество scp-процессов, находящихся в состоянии готовности к выполнению;
- ***runState*** – ключевой sc-узел, обозначающий множество scp-процессов, находящихся в состоянии выполнения;
- ***waitState*** – ключевой sc-узел, обозначающий множество scp-процессов, находящихся в состоянии ожидания, какого-нибудь условия;
- ***errorState*** – ключевой sc-узел, обозначающий множество scp-процессов, находящихся в состоянии ошибки;
- ***deadState*** – ключевой sc-узел, обозначающий множество scp-процессов, находящихся в состоянии завершения.

Теперь приведем примеры информационных конструкций, описывающих состояние абстрактной scp-машины.

Процесс абстрактной scp-машины (scp-процесс)

Процесс переработки информации в абстрактной scp-машине представляет собой совокупность одновременно протекающих последовательных процессов, которые будем называть **scp-процессами**.

SCP-процесс – это некая последовательность элементарных процессов, каждый из которых выполняется одной из операций scp-машины. Процесс абстрактной scp-машины порождается с помощью call-операторов или init-операций и завершается с помощью return-операторов или kill-операций. Каждому scp-процессу однозначно ставится в соответствие scp-программа, реализуемая этим scp-процессом. Обратное, вообще говоря, не верно, так как для хранимой scp-программы в текущий момент может существовать несколько реализующих ее scp-процессов.

В жизненном цикле scp-процесса условно можно выделить три стадии – порождение, непосредственное выполнение и завершение. Порождение scp-процессов инициируется хранимыми init-программами и demon-программами, а также осуществляется в результате реализации scp-операторов порождения scp-процессов, к числу которых относятся call-операторы. Завершение scp-процессов может быть естественным и внешним. Естественное завершение scp-процесса осуществляется при реализации в его рамках одного из return-операторов. Внешнее завершение (уничтожение) scp-процессов осуществляется также в результате реализации kill-операции абстрактной scp-машины. Внешнее завершение scp-процесса предполагает и внешнее завершение всех его сыновей, т.е. всех порожденных им неавтономных scp-процессов. Внешнее завершение scp-процесса приводит к внешнему завершению всех его потомков.

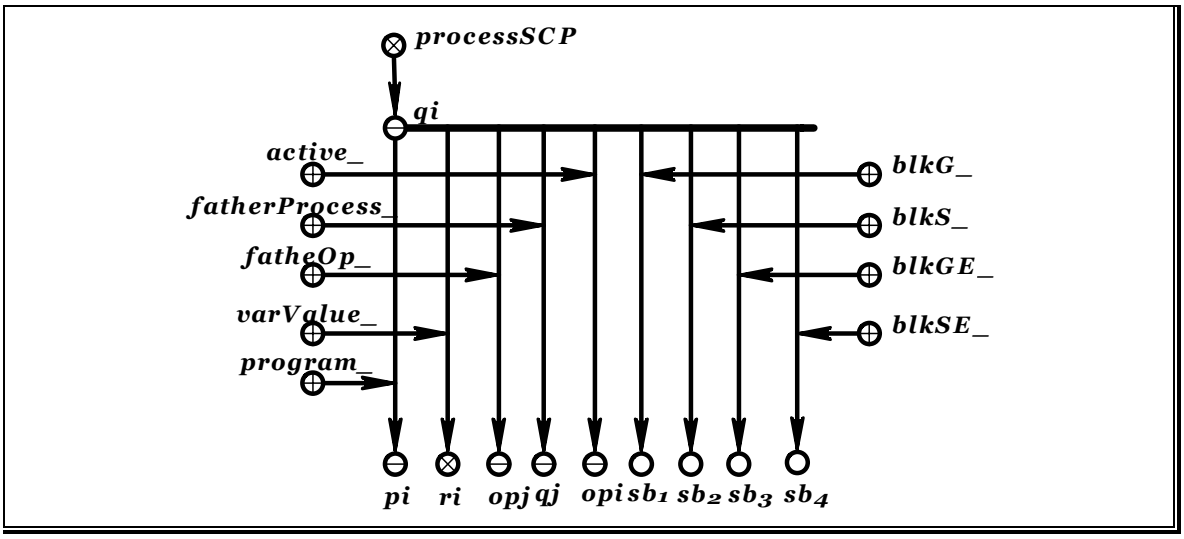
SCP-процессы бывают головными (автономными, независимыми, самостоятельными) и неголовными (неавтономными). Головные scp-процессы порождаются в результате реализации: 1) init-программ, 2) demon-программ. Все остальные scp-процессы считаются неголовными. Для каждого неголовного scp-процесса известен его отец, т.е. scp-процесс, породивший его в результате реализации соответствующего call-оператора. Для головного scp-процесса в случае, если головной процесс реализует init-программу или demon-программу, отцовский scp-процесс не существует.

Каждая scp-программа реализуется последовательно. Параллелизм scp-машины заключается в том, что разные scp-программы могут реализовываться параллельно (одновременно). Кроме того, параллельно могут выполняться scp-процессы, реализующие одну и ту же scp-программу, но работающие с разными данными. Поскольку scp-программа описывает не только преобразование перерабатываемых данных (поиск, генерацию, удаление), но и взаимодействие процесса реализации этой scp-программы с другими scp-процессами, то программист, работая на языке SCP, фактически проектирует коллектив взаимодействующих друг с другом и параллельно выполняемых scp-процессов.

Обмен перерабатываемыми данными между scp-процессами (как параллельно, так и не параллельно выполняемыми) осуществляется только через общедоступную для всех этих процессов графодинамическую память абстрактной scp-машины.

Непосредственное выполнение scp-процесса может прерываться. Прерывание наступает либо при реализации wait-оператора, либо при реализации любого другого оператора, если элементы перерабатываемых данных оказались соответствующим образом заблокированными со стороны других scp-процессов, выполняемых в текущий момент. Порождение сыновних процессов само по себе не вызывает прерывания отцовского процесса.

SCg-текст 4.3.1.1. Конструкция, описывающая состояние scp-процесса



Текущее состояние каждого scp-процесса, выполняемого в абстрактной scp-машине, описывается в памяти scp-машины с помощью соответствующей вспомогательной sc-конструкции. В описание текущего состояния scp-процесса входят (см. scg-текст 4.3.1.1):

- указание на реализуемую scp-программу (***pi***);
- указание на реализуемый в текущий момент оператор (***opi***);
- текущие значения программных переменных (***ri***);
- указание на scp-процесс, являющийся отцом (родителем) описываемого scp-процесса (***qj***);
- указание на оператор scp-программы, реализуемой отцовским процессом, в результате которого породился описываемый scp-процесс (***opj***);
- множество g-блокированных sc-элементов в рамках данного scp-процесса (***sb1***);
- множество s-блокированных sc-элементов в рамках данного scp-процесса (***sb2***);
- множество ge-блокированных sc-элементов в рамках данного scp-процесса (***sb3***);
- множество se-блокированных sc-элементов в рамках данного scp-процесса (***sb4***).

В каждый момент времени в scp-машине может существовать неограниченное количество одновременно выполняемых scp-процессов. Процесс, существующий в абстрактной scp-машине, может находиться в одном из следующих состояний:

- в активном состоянии (т.е. в состоянии реализации очередного оператора, не относящегося к классу wait-операторов);
- в состоянии ожидания , т.е. в состоянии реализации оператора из класса wait-операторов;
- в состоянии останова из-за ошибки программиста;
- в состоянии прерывания.

Реализация (интерпретация) scp-программ может осуществляться в двух режимах – в отладочном режиме и в обычном режиме. Отладочный режим проводит глубокий анализ корректности реализуемых scp-операторов и требует специальных языковых средств для записи программ отладки scp-программ. Обычный режим реализации scp-программ исходит из предположения, что реализуемая scp-программа достаточно хорошо отлажена и, следовательно, проводит только необходимый анализ корректности реализуемых scp-операторов.

Блокировки абстрактной scp-машины

Блокировка элементов перерабатываемой информации является основным способом синхронизации (согласования во времени) процессов, параллельно выполняемых над общей памятью. С помощью блокировок осуществляется разрешение конфликтов, которые могут возникнуть в случае преобразования одного и того же фрагмента перерабатываемой информации, осуществляемого одновременно несколькими процессами.

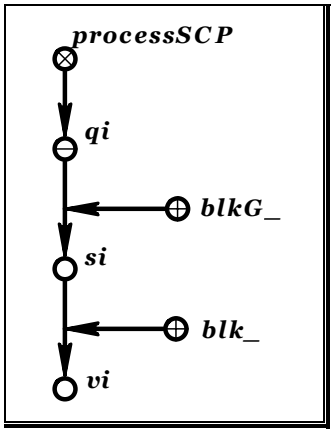
Хранимая в памяти абстрактной scp-машины sc-конструкция является общедоступной для всех параллельно выполняемых над ней scp-процессов, поэтому каждый scp-процесс должен иметь специальные возможности для защиты обрабатываемых им данных от разрушения со стороны других scp-процессов. Такая защита осуществляется с помощью блокировки sc-элементов. Блокировка sc-элемента ei, выставленная scp-процессом qi, есть запрет на некоторые виды использования sc-элемента ei другими scp-процессами, которые не известны процессу qi и, следовательно, не управляемы scp-процессом. Он может управлять только своими потомками.

Потомки scp-процесса – это продолжение того же алгоритма. Поэтому возможные конфликты между родственными scp-процессами можно регулировать. За корректное использование таких sc-элементов между родственными scp-процессами несет ответственность сам алгоритм.

Рассмотрим принципы организации блокировки sc-элементов для обеспечения взаимодействия scp-процессов через общую sc-память и для разрешения конфликтов, возникающих при обработке общих фрагментов перерабатываемых ими данных.

Каждому scp-процессу qi ставятся в соответствие четыре блокировочных sc-узла, каждый из которых обозначает множество sc-элементов, соответствующим образом заблокированных scp-процессом qi .

Эти блокировки задаются sc-конструкциями следующего вида:



В данной sc-конструкции используются следующие ключевые узлы:

- qi – узел, соответствующий некоторому scp-процессу;
- $blkG_$ – ключевой узел, обозначающий множество sc-дуг, каждая из которых выходит из узла, обозначающего конкретный scp-процесс (qi), и входит в узел, обозначающий множество sc-элементов, имеющих g-блокировку в рамках указанного процесса;
- $blk_$ – ключевой узел, обозначающий множество дуг, каждая из которых выходит из узла, обозначающего множество некоторым образом заблокированных элементов, и входит в заблокированный sc-элемент;
- vi – один из sc-элементов, имеющих g-блокировку в рамках процесса qi .

Аналогичным образом задаются и другие виды блокировок.

В памяти абстрактной scp-машины sc-элемент может иметь только одну блокировку из четырёх типов блокировок. Рассмотрим принципы работы с блокировками.

SC-элемент, имеющий **g-блокировку** в рамках scp-процесса qi , – это sc-элемент, сгенерированный в ходе выполнения scp-процесса qi в результате реализации одного из операторов генерации sc-элементов. SC-элемент, имеющий g-блокировку в рамках scp-процесса qi, абсолютно недоступен (невидим) для всех scp-процессов, не являющихся потомками scp-процесса qi. G-блокируемые sc-элементы – это элементы, которые мы предполагаем ввести в состав общедоступного sc-графа, но окончательное решение об этом мы откладываем до благополучного завершения головного scp-процесса.

SC-элемент, имеющий **s-блокировку** в рамках scp-процесса qi , – это sc-элемент, существовавший до начала выполнения scp-процесса qi и активно используемый в ходе выполнения scp-процесса qi . Чаще всего s-блокировка появляется в результате реализации оператора ассоциативного поиска.

S-блокировка sc-элемента в рамках scp-процесса qi запрещает другим процессам применять к этому sc-элементу erase-операторы. Исключение составляют только потомки scp-процесса qi.

Рассмотрим варианты работы с блокированными sc-элементами.

В процессе реализации операции генерации sc-элемента или sc-конструкции:

- если генерируется sc-узел или sc-элемент неопределенного типа, то на сгенерированный sc-элемент устанавливается g-блокировка текущего scp-процесса;
- если генерируется sc-дуга, и она входит или выходит из sc-элемента, имеющего s-блокировку (se-блокировку) неродительского scp-процесса, то scp-процесс, реализующий генерацию, переходит в состояние ожидания снятия s-блокировки (se-блокировки).

В процессе реализации операции ассоциативного поиска sc-элемента или sc-конструкции:

- если sc-элемент имеет g-блокировку (ge-блокировку) неродительского scp-процесса, то он исключается из области поиска (т.е. scp-процесс, реализующий операцию ассоциативного поиска, этот sc-элемент не видит);
- если sc-элемент имеет ge-блокировку (se-блокировку) текущего или родительского scp-процесса, то он исключается из области поиска;
- во всех остальных случаях sc-элемент видит scp-процесс, реализующий операцию ассоциативного поиска;
- при нахождении sc-элемента ему устанавливается s-блокировка текущего scp-процесса.

В процессе реализации операции удаления sc-элемента:

- если sc-элемент имеет g-блокировку текущего scp-процесса, то этот sc-элемент удаляется;
- если sc-элемент имеет g-блокировку родительского scp-процесса, то на этот sc-элемент устанавливается ge-блокировка родительского scp-процесса;
- если sc-элемент имеет s-блокировку родительского scp-процесса, то на этот sc-элемент устанавливается se-блокировка текущего scp-процесса;
- если sc-элемент имеет s-блокировку (se-блокировку) неродительского scp-процесса, то scp-процесс, реализующий операцию удаления, переходит в состояние ожидания снятия s-блокировки;
- во всех остальных случаях sc-элементу устанавливается se-блокировка текущего scp-процесса.

Естественное завершение головного scp-процесса состоит из следующих этапов:

- все sc-элементы, имеющие g-блокировку текущего scp-процесса, разблокируются, т.е. становятся общедоступными;
- все s-блокированные элементы разблокируются;
- все sc-элементы, имеющие ge-блокировку (se-блокировку) текущего scp-процесса, удаляются.

Естественное завершение неголовного scp-процесса:

- на все sc-элементы, имеющие g-блокировку текущего scp-процесса, устанавливается g-блокировка родительского scp-процесса;
- все sc-элементы, имеющие ge-блокировку (se-блокировку) текущего scp-процесса, удаляются;
- все s-блокированные элементы становятся s-блокированными в рамках отцовского процесса.

Внешнее завершение scp-процесса:

- все sc-элементы, имеющие g-блокировку, удаляются;
- если sc-элемент имеет другие блокировки текущего scp-процесса, то блокировки снимаются.

4.3.2. Элементарные преобразования над графодинамической ассоциативной памятью

Элементарные преобразования над графодинамической ассоциативной памятью по виду преобразуемых элементов делятся на два типа:

- преобразование заданного sc-элемента;
- преобразование окрестности sc-элемента.

Преобразования заданного sc-элемента включают в себя следующие операции:

- проверка типа sc-элемента (является ли sc-элемент sc-дугой, является ли sc-узел константным и т.д.);
- изменение типа sc-элемента (преобразование константного sc-элемента в переменный sc-элемент, преобразование позитивной sc-дуги в негативную sc-дугу и т.д.);
- проверка содержимого sc-элемента (является ли содержимое sc-элемента сформированным, сравнение двух содержимых sc-элементов и т.д.);
- изменение содержимого sc-элемента (арифметические операции над содержимым sc-элемена, преобразование из одного формата представления в другой и т.д.);
- поиск sc-элемента по идентификатору (нахождение sc-элемента с заданным идентификатором);
- изменение идентификатора sc-элемента (создание идентификатора sc-элемента).

Преобразование окрестности sc-элемента состоит из следующих операций:

- генерация выходящей или входящей дуги sc-элемента;
- удаление выходящей или входящей дуги sc-элемента;
- нахождение множества входящих или выходящих дуг sc-элемента;
- объединение sc-множеств;
- нахождение пересечения sc-множеств

4.3.3. Операции генерации конструкций языка SC

В рассматриваемое семейство операций абстрактной scp-машины входят следующие операции:

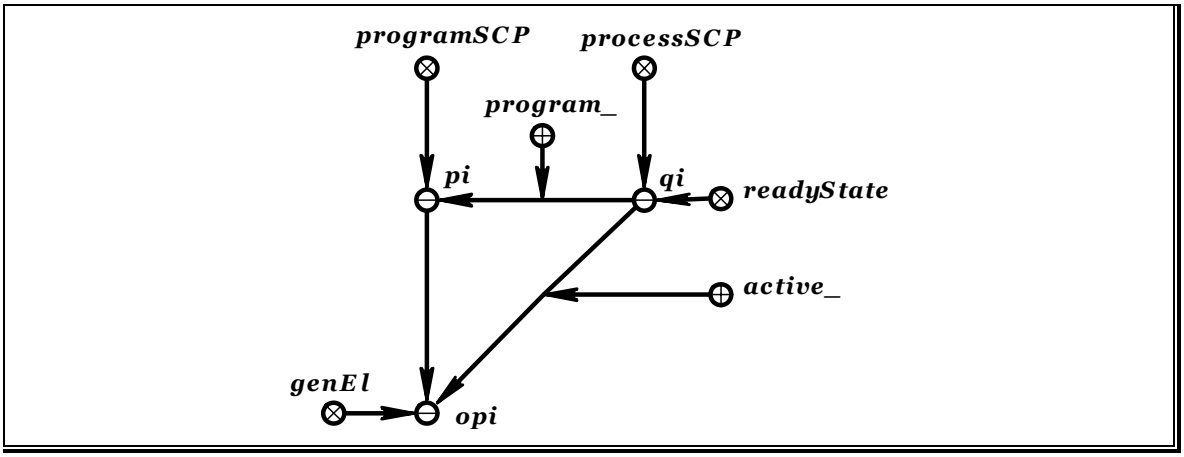
- операция генерации sc-элемента;
- операция генерации sc-конструкции, состоящей из трёх элементов.

Каждая из вышеперечисленных операций реализует конкретный тип scp-операторов (соответственно genEl-операторы и genStr3-операторы).

Операция генерации sc-элемента

Условием выполнения scp-операции генерации sc-элемента является наличие в памяти абстрактной scp-машины sc-конструкции, представленной на scg-тексте 4.3.3.1.

SCg-текст 4.3.3.1. Условие выполнения scp-операции генерации sc-элемента



Результатом выполнения scp-операции генерации sc-элемента является создание в памяти абстрактной scp-машины sc-элемента (см. описание операторов типа ***genEl***).

Микропрограмма scp-операции генерации sc-элемента имеет следующий вид.

Шаг 1. Ищется scp-оператор (обозначим его ***opi***) типа ***genEl***, являющийся активным в рамках некоторого scp-процесса (обозначим его ***qi***). Для этого надо найти sc-конструкцию, определяющую

условие выполнения рассматриваемой операции (см. scg-текст 4.3.3.1). Если sc-конструкция найдена, то перейти к шагу 2, иначе к шагу 1.

Шаг 2. Ищется sc-дуга (обозначим её gi), выходящая из узла, обозначающего scp-оператор opi (см. шаг 1) и не отмеченная атрибутом $goto_$. Узел, в который найденная дуга входит, обозначим xi , а саму дугу – gi .

Шаг 3. Ищется sc-конструкция вида:

$$node_ \rightarrow gi;$$

Если указанная конструкция имеется (это означает, что необходимо сгенерировать sc-узел), то переход к шагу 4, иначе к шагу 19.

Шаг 4. Проверяется, существует ли в памяти sc-конструкция вида:

$$const_ \rightarrow gi;$$

Если существует (это означает, что необходимо сгенерировать константный sc-узел), то переход к шагу 5, иначе к шагу 9.

Шаг 5. Ищется sc-дуга (обозначим её gj), выходящая из sc-узла операнда (xi) и входящая в sc-элемент (обозначим его vj), в которую входит дуга принадлежности, связывающая указанную выше sc-дугу со знаком бинарного отношения (обозначим это отношение ri). Наличие этой sc-дуги означает, что у операнда xi значение вычислено. Отношение ri – это множество всех вычисленных значений в рамках конкретного scp-процесса (см. 4.3.1). Если эта sc-дуга не существует, то переходим к шагу 6, иначе к шагу 7.

Шаг 6. Проверяем, является ли sc-элемент vj константным узлом, и если является, то переход к шагу 37, иначе к шагу 7.

Шаг 7. Удаляем sc-дугу gj, это означает, что у операнда xi значение становится не вычисленным.

Шаг 8. Генерируем константный sc-узел (обозначим его vi) и переходим к шагу 35.

Шаг 9. Проверяется, существует ли в памяти sc-конструкция вида:

$$var_ \rightarrow gi;$$

Если существует (это означает, что необходимо сгенерировать переменный sc-узел), то переход к шагу 10, иначе к шагу 14.

Шаг 10. Ищется sc-дуга (обозначим её gj), выходящая из sc-узла операнда (xi) и входящая в sc-элемент (обозначим его vj), в которую входит дуга принадлежности, связывающая указанную выше sc-дугу со знаком бинарного отношения (обозначим это отношение ri). Наличие этой sc-дуги означает, что у операнда xi значение вычислено. Отношение ri – это множество всех вычисленных значений в рамках конкретного scp-процесса (см. 4.3.1). Если эта sc-дуга не существует, то переходим к шагу 13, иначе к шагу 11.

Шаг 11. Проверяем, является ли sc-элемент vj переменным узлом, если является, то переход к шагу 37, иначе к шагу 12.

Шаг 12. Удаляем sc-дугу gj, это означает, что у операнда xi значение становится не вычисленным.

Шаг 13. Генерируем переменным sc-узел (обозначим его vi) и переходим к шагу 35.

Шаг 14. Проверяется, существует ли в памяти sc-конструкция вида:

meta_ —> gi ;

Если существует (это означает, что необходимо сгенерировать метапеременный sc-узел), то перейти к шагу 15, иначе к шагу 43.

Шаг 15. Ищется sc-дуга (обозначим её ***gj***), выходящая из sc-узла операнда (***xi***) и входящая в sc-элемент (обозначим его ***vj***), в которую входит дуга принадлежности, связывающая указанную выше sc-дугу со знаком бинарного отношения (обозначим это отношение ***ri***). Наличие этой sc-дуги означает, что у операнда ***xi*** значение вычислено. Отношение ***ri*** – это множество всех вычисленных значений в рамках конкретного scp-процесса (см. 4.3.1). Если эта sc-дуга не существует, то переходим к шагу 18, иначе к шагу 16.

Шаг 16. Проверяем, является ли sc-элемент ***vj*** константным узлом, и если является, то переходим к шагу 37, иначе к шагу 17.

Шаг 17. Удаляем sc-дугу ***gj***, это означает, что у операнда ***xi*** становится не вычислено значение.

Шаг 18. Генерируем метапеременный sc-узел (обозначим его ***vi***) и переходим к шагу 35.

Шаг 19. Проверяется, существует ли в памяти sc-конструкция вида:

elem_ —> gi ;

Если существует (это означает, что необходимо сгенерировать sc-элемент неопределенного типа), то переходим к шагу 20, иначе к шагу 38.

Шаг 20. Проверяется, существует ли в памяти sc-конструкция вида:

const_ —> gi ;

Если существует (это означает, что необходимо сгенерировать константный sc-элемент неопределенного типа), то переход к шагу 21, иначе к шагу 25.

Шаг 21. Ищется sc-дуга (обозначим её ***gj***), выходящая из sc-узла операнда (***xi***) и входящая в sc-элемент (обозначим его ***vj***), в которую входит дуга принадлежности, связывающая указанную выше sc-дугу со знаком бинарного отношения (обозначим это отношение ***ri***). Наличие этой sc-дуги означает, что у операнда ***xi*** значение вычислено. Отношение ***ri*** – это множество всех вычисленных значений в рамках конкретного scp-процесса (см. 4.3.1). Если эта sc-дуга не существует, то переходим к шагу 24, иначе к шагу 22.

Шаг 22. Проверяем, является ли sc-элемент ***vj*** константным элементом неопределенного типа, если является, то переход к шагу 37, иначе к шагу 23.

Шаг 23. Удаляем sc-дугу ***gj***, это означает, что у операнда ***xi*** значение становится невычисленным.

Шаг 24. Генерируем константный sc-элемент неопределенного типа (обозначим его ***vi***) и переходим к шагу 35.

Шаг 25. Проверяется, существует ли в памяти sc-конструкция вида:

var_ —> gi ;

Если существует (это означает, что необходимо сгенерировать переменный sc-элемент неопределенного типа), то переход к шагу 26, иначе к шагу 30.

Шаг 26. Ищется sc-дуга (обозначим её gj), выходящая из sc-узла операнда (xi) и входящая в sc-элемент (обозначим его vj), в которую входит дуга принадлежности, связывающая указанную выше sc-дугу со знаком бинарного отношения (обозначим это отношение ri). Наличие этой sc-дуги означает, что у операнда xi значение вычислено. Отношение ri – это множество всех вычисленных значений в рамках конкретного scp-процесса (см. 4.3.1). Если эта sc-дуга не существует, то переходим к шагу 27, иначе к шагу 29.

Шаг 27. Проверяем, является ли sc-элемент vj переменным элементом неопределенного типа, и если является то переход к шагу 37, иначе к шагу 28.

Шаг 28. Удаляем sc-дугу gj, это означает, что у операнда xi значение становится невычисленным.

Шаг 29. Генерируем переменный sc-элемент неопределенного типа (обозначим его vi) и переходим к шагу 35.

Шаг 30. Проверяется, существует ли в памяти sc-конструкция вида:

$$meta_ \rightarrow gi\ ;$$

Если существует (это означает, что необходимо сгенерировать метапеременный sc-элемент неопределенного типа), то переход к шагу 31, иначе к шагу 43.

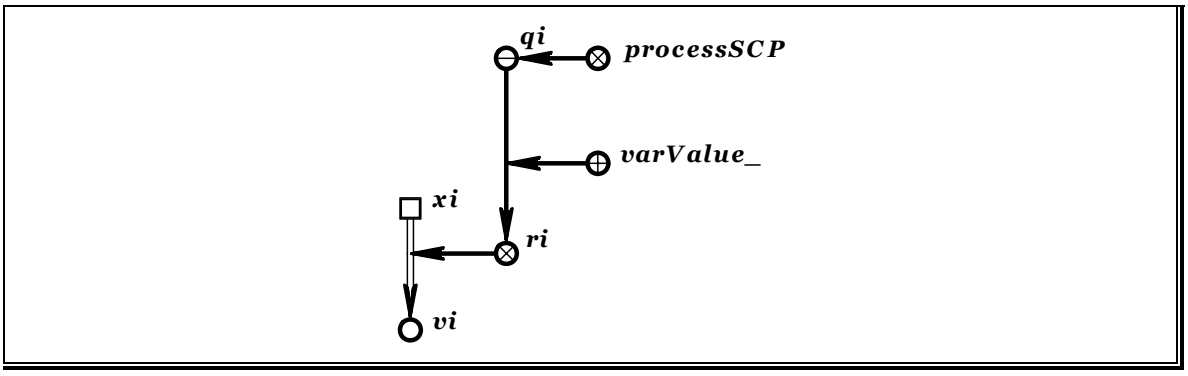
Шаг 31. Ищется sc-дуга (обозначим её gj), выходящая из sc-узла операнда (xi) и входящая в sc-элемент (обозначим его vj), в которую входит дуга принадлежности, связывающая указанную выше sc-дугу со знаком бинарного отношения (обозначим это отношение ri). Наличие этой sc-дуги означает, что у операнда xi значение вычислено. Отношение ri – это множество всех вычисленных значений в рамках конкретного scp-процесса (см. 4.3.1). Если эта sc-дуга не существует, то переходим к шагу 34, иначе к шагу 32.

Шаг 32. Проверяем, является ли sc-элемент vj метапеременным элементом неопределенного типа, если является, то переход к шагу 37, иначе к шагу 33.

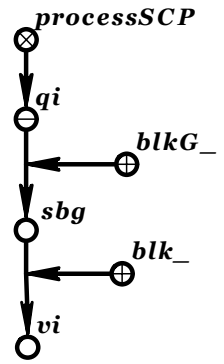
Шаг 33. Удаляем sc-дугу gj, это означает, что у операнда xi значение становится не вычисленным.

Шаг 34. Генерируем метапеременный sc-элемент неопределенного типа (обозначим его vi).

Шаг 35. Генерируется sc-дуга, выходящая из sc-узла xi операнда и входящая в сгенерированный sc-элемент (vi), а также в эту sc-дугу генерируется sc-дуга из sc-узла ri, обозначающего множество вычисленных значений конкретного scp-процесса (qi).

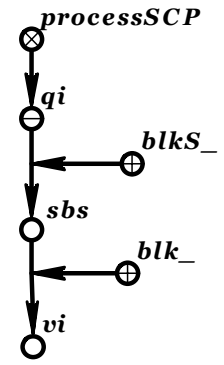


Шаг 36. Генерируется sc-дуга, выходящая из множества (sbg), обозначающего g-блокированные элементы конкретного scp-процесса, в сгенерированный sc-элемент vi. В сгенерированную sc-дугу проводится sc-дуга из атрибута $blk_$.



Переход к шагу 38.

Шаг 37. Генерируется sc-дуга, выходящая из множества (***sbs***), обозначающего s-блокированные элементы конкретного scp-процесса, в сгенерированный sc-элемент ***vi***. В сгенерированную sc-дугу проводится sc-дуга из атрибута ***blk_***.



Шаг 38. Ищется следующий scp-оператор (обозначим его ***opj***). Для этого надо найти sc-конструкцию вида:

opi —≻ ***goto_*** : ***opj*** ;

Шаг 39. Удаляется sc-дуга, выходящая из sc-узла ***qi*** процесса и входящая в sc-узел оператора ***opi***:

qi —≻ ***opi*** ;

Шаг 40. Генерируется sc-дуга между sc-узлом процесса (***qi***) и sc-узлом оператора (***opj***), и в эту sc-дугу генерируется sc-дуга из ключевого sc-узла ***active_*** :

qi —≻ ***active_*** : ***opj*** ;

Шаг 41. Переход к шагу 1.

Шаг 42. Удаляется sc-дуга, выходящая из ключевого узла ***readyState*** и входящая в sc-узел scp-процесса (***qi***):

readyState —≻ ***qi*** ;

Шаг 43. Генерируется sc-дуга между ключевым узлом ***errorState*** и sc-узлом scp-процесса (***qi***):

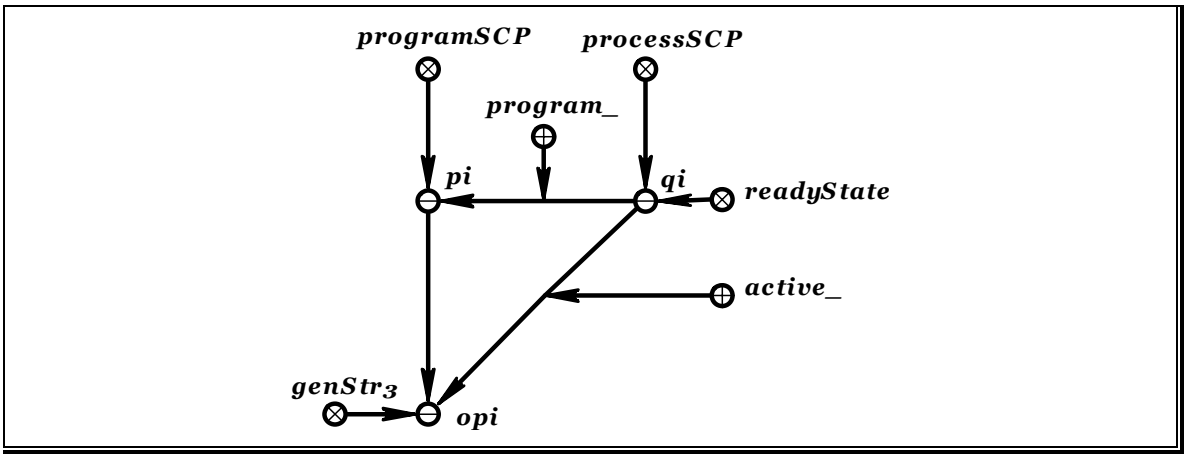
errorState —> ***qi*** **;**

Шаг 44. Переход к шагу 1.

Операция генерации sc-конструкций, состоящих из трёх элементов.

Условием выполнения scp-операции генерации sc-конструкции, состоящей из трёх элементов, является наличие в памяти абстрактной scp-машины sc-конструкции, представленной на scg-тексте 4.3.3.2.

SCg-текст 4.3.3.2. Условие выполнения scp-операции генерации sc-конструкции, состоящей из трёх элементов



Результатом выполнения scp-операции генерации sc-конструкции, состоящей из трёх элементов, является создание в памяти абстрактной scp-машины sc-конструкции. Конфигурация генерируемой sc-конструкции описана при рассмотрении scp-оператора типа ***genStr3***. На все сгенерированные sc-элементы устанавливается g-блокировка текущего scp-процесса.

4.3.4. Операции удаления конструкций языка SC

В рассматриваемое семейство операций абстрактной scp-машины входят следующие операции:

- операция удаления sc-элемента;
- операция удаления sc-конструкции, состоящей из трёх элементов;
- операция удаления нескольких sc-конструкций, состоящих из трёх элементов.

Каждая из вышеперечисленных операций реализует конкретный тип scp-операторов (соответственно eraseEl-операторы, eraseElStr3-операторы и eraseSetStr3-операторы).

Условием выполнения каждой рассматриваемой операции является наличие в рамках выполняемой scp-программы активного scp-оператора соответствующего типа.

Результат выполнения рассматриваемых операций аналогичен описанному результату выполнения соответствующих scp-операторов с учётом работы с блокировками, а именно:

- если удаляемый sc-элемент имеет g-блокировку текущего scp-процесса, то этот sc-элемент удаляется;
- если удаляемый sc-элемент имеет g-блокировку родительского scp-процесса, то на sc-элемент устанавливается ge-блокировка родительского scp-процесса;
- если удаляемый sc-элемент имеет s-блокировку родительского scp-процесса, то на sc-элемент устанавливается se-блокировка текущего scp-процесса;

- если удаляемый sc-элемент имеет s-блокировку (se-блокировку) не родительского scp-процесса, то scp-процесс реализующий операцию удаления переходит в состояние ожидания снятия s-блокировки;
- во всех остальных случаях удаляемому sc-элементу устанавливается se-блокировка текущего scp-процесса.

4.3.5. Операции ассоциативного поиска конструкций языка SC

В рассматриваемое семейство операций абстрактной scp-машины входят следующие операции:
- операция ассоциативного поиска произвольных элементов sc-конструкции, состоящей из трёх элементов;
- операция ассоциативного поиска произвольных элементов sc-конструкций, удовлетворяющих заданным условиям и состоящих из трех элементов, и формирования из них множества;
- операция ассоциативного поиска произвольных элементов sc-конструкций, удовлетворяющих заданным условиям и состоящих из трёх элементов, и формирования из них множества, элементами которого остаются только sc-элементы, удовлетворяющие заданным условиям;
- операция ассоциативного поиска произвольных элементов sc-конструкций, удовлетворяющих заданным условиям и состоящих из трёх элементов, и формирования из них множества, элементами которого остаются только sc-элементы, не удовлетворяющие заданным условиям.

Каждая из выше перечисленных операций реализует конкретный тип scp-операторов (соответственно searchElStr3-операторы, searchSetStr3-операторы, selectYStr3-операторы и selectNStr3-операторы).

Условием выполнения каждой рассматриваемой операции является наличие в рамках выполняемой scp-программы активного scp-оператора соответствующего типа.

Результат выполнения рассматриваемых операций аналогичен описанному результату выполнения соответствующих scp-операторов с учётом работы с блокировками, а именно:
- если найденный sc-элемент имеет g-блокировку (ge-блокировку) неродительского scp-процесса, то он исключается из области поиска (т.е. scp-процесс реализующий операцию ассоциативного описка это sc-элемент не видит);
- если найденный sc-элемент имеет ge-блокировку (se-блокировку) текущего или родительского scp-процесса, то он исключается из области поиска;
- во всех остальных случаях найденный sc-элемент доступен scp-процессу, реализующему операцию ассоциативного поиска;
- при нахождении sc-элемента ему устанавливается s-блокировка текущего scp-процесса.

4.3.6. Операции проверки условий

В рассматриваемое семейство операций абстрактной scp-машины входят следующие операции:
- операция проверки типа sc-элемента;
- операция проверки равенства значений двух операндов;
- операция проверки строгого неравенства значений двух операндов;
- операция проверки неравенства значений двух операндов.

Каждая из вышеперечисленных операций реализует конкретный тип scp-операторов (соответственно ifType-операторы, ifEq-операторы, ifGr-операторы и ifGrEq-операторы).

Условием выполнения каждой рассматриваемой операции является наличие в рамках выполняемой scp-программы активного scp-оператора соответствующего типа.

Результат выполнения рассматриваемых операций аналогичен описанному результату выполнения соответствующих scp-операторов.

4.3.7. Операции изменения свойств элементов конструкций языка SC

В рассматриваемое семейство операций абстрактной scp-машины входят следующие операции:

- операции сложения содержимых sc-элементов (add-операторы);
- операции вычитания содержимых sc-элементов (sub-операторы)

Каждая из вышеперечисленных операций реализует конкретный тип scp-операторов (соответственно add-операторы и sub-операторы).

Условием выполнения каждой рассматриваемой операции является наличие в рамках выполняемой scp-программы активного scp-оператора соответствующего типа.

Результат выполнения рассматриваемых операций аналогичен описанному результату выполнения соответствующих scp-операторов.

4.3.8. Операции управления scp-процессами

В рассматриваемое семейство операций абстрактной scp-машины входят следующие операции:

- операция порождения scp-процесса (call-операторы);
- операция порождения параллельных scp-процесов (callPar-операторы);
- операция завершения scp-процесса (return-операторы);
- операция ожидания завершения scp-процесса (waitReturn-операторы);
- операция инициализации scp-программы;
- операция уничтожения scp-процесса.

Операция порождения scp-процесса реализует call-операторы. Условием выполнения операции порождения scp-процесса является наличие в рамках выполняемой scp-программы активного call-оператора. Результатом выполнения рассматриваемой операции является последовательность следующих действий:

- создание структуры scp-процесса, который будет выполнять вызываемую scp-программу. Узел, соответствующий вызываемой scp-программе, является значением первого операнда call-оператора;
- устанавливается связь созданного scp-процесса с текущим scp-процессом и call-оператором (см. структуру scp-процесса);
- активным scp-оператором становится scp-оператор, помеченный атрибутом ***init_*** ;
- в вызванную scp-программу передаются значения входных параметров;
- значением третьего операнда call-оператора становится sc-узел, обозначающий scp-процесс;
- созданный scp-процесс устанавливается в состояние ***readyState***.

Операция порождения параллельных scp-процессов реализует callPar-операторы. Условием выполнения данной операции является наличие в рамках выполняемой scp-программы активного callPar-оператора. Результатом выполнения рассматриваемой операции является создание множества scp-процессов (о создании отдельного scp-процесса см. выше).

Операция завершения scp-процесса реализует return-операторы. Условием выполнения данной операции является наличие в рамках выполняемой scp-программы активного return-оператора. Результатом выполнения рассматриваемой операции является последовательность следующих описанных ниже действий.

Если завершаемый scp-процесс является головным, то:

- все sc-элементы, имеющие g-блокировку текущего scp-процесса, разблокируются, т.е. становятся общедоступными;
- все s-блокированные элементы разблокируются;
- все sc-элементы, имеющие ge-блокировку (se-блокировку) текущего scp-процесса, удаляются;
- все дочерние scp-процессы включаются во множество ***kill***, т.е. уничтожаются;
- удаляется структура scp-процесса.

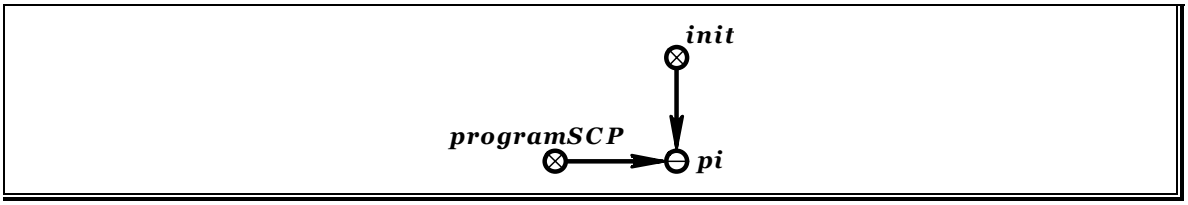
Если заверашаемый scp-процесс является неголовным, то:

- на все sc-элементы, имеющие g-блокировку текущего scp-процесса, устанавливается g-блокировка родительского scp-процесса;
- все sc-элементы имеющие ge-блокировку (se-блокировку) текущего scp-процесса, удаляются;
- все s-блокированные элементы становятся s-блокированными в рамках отцовского процесса;
- передаются значения возвращаемых параметров;
- все дочерние scp-процессы включаются во множество ***kill***, т.е. уничтожаются;
- удаляется структура scp-процесса.

Операция ожидания завершения scp-процесса реализует waitReturn-операторы. Условием выполнения данной операции является наличие в рамках выполняемой scp-программы активного waitReturn-оператора. Результатом выполнения рассматриваемой операции является последовательность следующих действий:

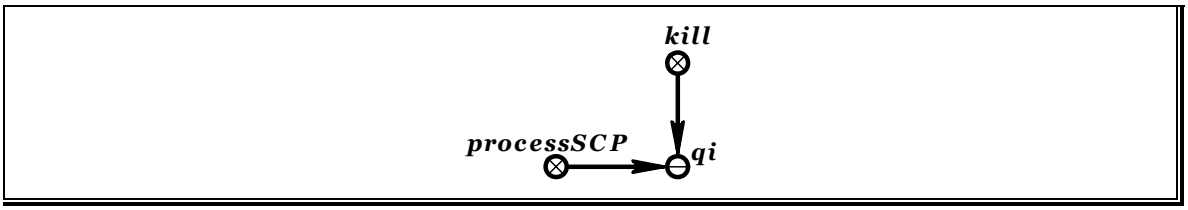
- текущий scp-процесс переводится в состояние ожидания (это означает, что проводится дуга из ключевого sc-узла ***waitState*** в sc-узел, обозначающий scp-процесс);
- когда ожидаемый scp-процесс завершается или уничтожается, текущий процесс переходит в состояние готовности (это означает, что проводится дуга из ключевого sc-узла ***readyState*** в sc-узел, обозначающий scp-процесс);
- управление передается следующему scp-оператору.

Операция инициализации scp-программы осуществляет запуск init-программ. Условием выполнения данной операции является наличие в памяти следующей sc-конструкции:



Результатом выполнения операции инициализации scp-программы является создание структуры scp-процесса, который будет реализовывать init-программу (см. операцию порождения scp-процесса).

Операция уничтожения scp-процесса осуществляет внешнее завершения scp-процесса. Условием выполнения данной операции является наличие в памяти следующей sc-конструкции:



Результатом выполнения операции уничтожения scp-процесса является последовательность следующих действий:

- все sc-элементы, имеющие g-блокировку, удаляются;
- если sc-элемент имеет другие блокировки текущего scp-процесса, то блокировки снимаются;
- все дочерние scp-процессы включаются во множество ***kill***, т.е. уничтожаются.

4.4. Ключевые узлы языка SCP

В данном подразделе описываются ключевые узлы языка SCP с комментариями. Также приводится описание системных конструкций scp-машины (структура процесса, блокировок, коммуникационного узла и т.д.). Данное описание произведено на языке SC, который описан в работе [342] (***ПредстИОЗвГАМ-2001кн***).

active_; **/*** Описание ключевого узла ***active_*** ***/***

пояснение **—>** **‹*** ***active_, текст пояснения_*** **: /"** Ключевой узел ***active_*** является знаком атрибута, указывающего, что элемент в отношении является активным. В частности в отношении описывающем scp-процесс указывает активный оператор **"/ ›;**
главный синоним **—>** ***active_;***

arc_; **/*** Описание ключевого узла ***arc_*** ***/***

пояснение **—>** **‹*** ***arc_, текст пояснения_*** **: /"**Ключевой узел ***arc_*** указывает, что в составе некоторого оператора значением операнда, помеченного атрибутом ***arc_,*** является сгенерированная (искомая) sc-дуга. **"/ ›;**
главный синоним **—>** ***arc_;***

assign_; **/*** Описание ключевого узла ***assign_*** ***/***

пояснение **—>** **‹*** ***assign_, текст пояснения_*** **: /"**Ключевой узел ***assign_*** указывает, что значениями соответствующих операндов некоторого оператора должны стать новые сгенерированные sc-элементы. **"/ ›;**
главный синоним **—>** ***assign_;***

blkG_; **/*** Описание ключевого узла ***blkG_*** ***/***

пояснение **—>** **‹** ***blkG_, текст пояснения_*** **: /"**Ключевой узел ***blkG_*** является атрибутом узла, обозначающего множество элементов, имеющих g-блокировку в рамках заданного scp-процесса. **"/›;**
главный синоним **—>** ***blkG_;***

blkGE_; **/*** Описание ключевого узла ***blkGE_*** ***/***

пояснение **—>** **‹** ***blkGE_, текст пояснения_*** **: /"**Ключевой узел ***blkGE_*** является атрибутом узла, обозначающего множество элементов, имеющих ge-блокировку в рамках заданного scp-процесса. **"/›;**
главный синоним **—>** ***blkGE_;***

blkS_; **/*** Описание ключевого узла ***blkS_*** ***/***

пояснение **—>** **‹** ***blkS_, текст пояснения_*** **: /"**Ключевой узел ***blkS_*** является атрибутом узла, обозначающего множество элементов, имеющих s-блокировку в рамках заданного scp-процесса. **"/›;**
главный синоним **—>** ***blkS_;***

blkSE_; **/*** Описание ключевого узла ***blkSE_*** ***/***

пояснение **—>** **‹** ***blkSE_, текст пояснения_*** **: /"**Ключевой узел ***blkSE_*** является атрибутом узла, обозначающего множество элементов, имеющих se-блокировку в рамках заданного scp-процесса. **"/›;**
главный синоним **—>** ***blkSE_;***

connBin_; **/*** Описание ключевого узла ***connBin_*** ***/***

пояснение —> **<** ***connBin_, текст пояснения_*** **: /"**Ключевой узел ***connBin_*** указывает что значениями соответствующих операндов некоторого опера тора являются сгенерированные (искомые) sc-дуги, представляющие бинарный кортеж, не принадлежащий отношению принадлежности. **"/>;**
главный синоним —> ***connBin_*;**

const_*;** **/ Описание ключевого узла ***const_*** ***/**

пояснение —> **<** ***const_, текст пояснения_*** **: /"**Ключевой узел ***const_*** указывает, что в составе некоторого оператора значением операнда, помеченного атрибутом ***const_,*** является сгенерированный (искомый) константный sc-элемент. **"/>;**
главный синоним —> ***const_*;**

nearSem —> **(*const_, var_*);**

demon*;** **/ Описание ключевого узла ***demon*** ***/**

пояснение —> **<** ***demon, текст пояснения_*** **: /"**Ключевой узел ***demon*** является знаком множества демонических (самоинициируемых) программ. **"/>;**
главный синоним —> ***demon*;**

elem_*;** **/ Описание ключевого узла ***elem_*** ***/**

пояснение —> **<** ***elem_, текст пояснения_*** **: /"**Ключевой узел ***elem_*** указывает, что в составе некоторого оператора значением операнда, помеченного атрибутом ***elem_,*** является сгенерированный (искомый) sc-элемент неопределенного типа. **"/>;**
главный синоним —> ***elem_*;**

else_*;** **/ Описание ключевого узла ***else_*** ***/**

пояснение —> **<** ***else_, текст пояснения_*** **: /"**Ключевой узел ***else_*** является атрибутом, связывающего два соседних оператора. Этот атрибут задается для операторов класса ***search*** и класса ***if.*** Семантика атрибута ***else_*** определяется в зависимости от конкретного выполняемого в текущий момент оператора. **"/>;**
главный синоним —> ***else_*;**

fixed_*;** **/ Описание ключевого узла ***fixed_*** ***/**

пояснение —> **<** ***fixed_, текст пояснения_*** **: /"**Ключевой узел ***fixed_*** указывает, что текущими значениями соответствующих операндов в некотором операторе являются уже существующие (известные) sc-элементы. В случае, если значение операнда не вычисленно, то будет сгенерирован новый sc-элемент. **"/>;**
главный синоним —> ***fixed_*;**

fuz_*;** **/ Описание ключевого узла ***fuz_*** ***/**

пояснение —> **<** ***fuz_, текст пояснения_*** **: /"**Ключевой узел ***fuz_*** указывает, что в составе некоторого оператора значением операнда, помеченного атрибутом ***fuz_,*** является сгенерированная (искомая) не четкая sc-дуга. **"/>;**
главный синоним —> ***fuz_*;**

goto_*;** **/ Описание ключевого узла ***goto_*** ***/**

пояснение —> **<** ***goto_, текст пояснения_*** **: /"**Ключевой узел ***goto_*** является знаком атрибута, связывающего два соседних оператора (предыдущий и следующий) **"/>;**
главный синоним —> ***goto_*;**

init*;** **/ Описание ключевого узла ***init*** ***/***

пояснение **—>** **‹** ***init, текст пояснения_*** **: /"**Ключевой узел ***init*** является знаком множества инициируемых программ. **"/›;**
главный синоним **—>** ***init*;**

init_*;** **/ Описание ключевого узла ***init_*** ***/***

пояснение **—>** **‹** ***init_, текст пояснения_*** **: /"**Ключевой узел ***init_*** указывает на первый выполняемый оператор конкретной программы. **"/›;**
главный синоним **—>** ***init_*;**

neg_*;** **/ Описание ключевого узла ***neg_*** ***/***

пояснение **—>** **‹** ***neg_, текст пояснения_*** **: /"**Ключевой узел ***neg_*** указывает, что в составе некоторого оператора значением операнда, помеченного атрибутом ***neg_,*** является сгенерированная (искомая) негативная sc-дуга. **"/›;**
главный синоним **—>** ***neg_*;**

node_*;** **/ Описание ключевого узла ***node_*** ***/***

пояснение **—>** **‹** ***node_, текст пояснения_*** **: /"**Ключевой узел ***node_*** указывает, что в составе некоторого оператора значением операнда, помеченного атрибутом ***node_,*** является сгенерированный (искомый) sc-узел. **"/›;**
главный синоним **—>** ***node_*;**

pos_*;** **/ Описание ключевого узла ***pos_*** ***/***

пояснение **—>** **‹** ***pos_, текст пояснения_*** **: /"**Ключевой узел ***pos_*** указывает, что в составе некоторого оператора значением операнда, помеченного атрибутом ***pos_,*** является сгенерированная (искомая) позитивная SC-дуга. **"/›;**
главный синоним **—>** ***pos_*;**

prm_*;** **/ Описание ключевого узла ***prm_*** ***/***

пояснение **—>** **‹** ***prm_, текст пояснения_*** **: /"**Ключевой узел ***prm_*** указывает на множество параметров конкретной программы. **"/›;**
главный синоним **—>** ***prm_*;**

programSCP*;** **/ Описание ключевого узла ***programSCP*** ***/***

пояснение **—>** **‹** ***programSCP, текст пояснения_*** **: /"**Ключевой узел ***programSCP*** является знаком множества всех программ, написанных на языке SCP. **"/›;**
главный синоним **—>** ***programSCP*;**

set_1**_;** set_2**_;** set_3**_;** set_4**_;** set_5**_;** set_6**_;** set_7**_;**

/* Описание ключевых узлов set_1**_,** set_2**_,** set_3**_** set_4**_,** set_5**_,** set_6**_,** set_7**_** ***/***

пояснение **—>** **‹** set_1**_, *текст пояснения_*** **: /"**Ключевые узлы set_1**_,** set_2**_,** set_3**_** set_4**_,** set_5**_,** set_6**_,** set_7**_** являются атрибутами операндов операторов подкласса ***searchSet.*** **"/›;**

пояснение **—>** **‹** set_2**_, *текст пояснения_*** **: /"**Ключевые узлы set_1**_,** set_2**_,** set_3**_** set_4**_,** set_5**_,** set_6**_,** set_7**_** являются атрибутами операндов операторов подкласса ***searchSet.*** **"/›;**

пояснение **—>** **‹** set_3**_, *текст пояснения_*** **: /"**Ключевые узлы set_1**_,** set_2**_,** set_3**_** set_4**_,** set_5**_,** set_6**_,** set_7**_** являются атрибутами операндов операторов подкласса ***searchSet.*** **"/›;**

пояснение —> **<** ***set4_, текст пояснения_*** **: /"**Ключевые узлы ***set1_, set2_, set3_ set4_, set5_, set6_, set7_*** являются атрибутами операндов операторов подкласса ***searchSet.*** **"/>;**

пояснение —> **<** ***set5_, текст пояснения_*** **: /"**Ключевые узлы ***set1_, set2_, set3_ set4_, set5_, set6_, set7_*** являются атрибутами операндов операторов подкласса ***searchSet.*** **"/>;**

пояснение —> **<** ***set6_, текст пояснения_*** **: /"**Ключевые узлы ***set1_, set2_, set3_ set4_, set5_, set6_, set7_*** являются атрибутами операндов операторов подкласса ***searchSet.*** **"/>;**

пояснение —> **<** ***set7_, текст пояснения_*** **: /"**Ключевые узлы ***set1_, set2_, set3_ set4_, set5_, set6_, set7_*** являются атрибутами операндов операторов подкласса ***searchSet.*** **"/>;**
главный синоним —> ***set1_, set2_, set3_, set4_, set5_, set6_, set7_;***

then_; **/*** Описание ключевого узла ***then_*** ***/**

пояснение —> **<** ***then_, текст пояснения_*** **: /"**Ключевой узел ***then_*** является знаком атрибута, связывающего два соседних оператора. Это отношение задается для операторов класса ***search*** и класса ***if.*** Семантика атрибута ***then_*** определяется в зависимости от конкретного выполняемого в текущий момент оператора. **"/>;**
главный синоним —> ***then_;***

nearSem —> **(*****then_, goto_, else_*****);**

var_; **/*** Описание ключевого узла ***var_*** ***/**

пояснение —> **<** ***var_, текст пояснения_*** **: /"**Ключевой узел ***var_*** указывает, что в составе некоторого оператора значением операнда, помеченного атрибутом ***var_,*** является сгенерированный (искомый) переменный SC-элемент. **"/>;**
главный синоним —> ***var_;***

nearSem —> **(*****var_, const_*****);**

1_; 2_; **/*** Описание ключевого узла ***1_, 2_*** ***/**

пояснение —> **<** ***1_, текст пояснения_*** **: /"**Ключевые узлы ***1_, 2_*** и т.д. являются атрибутами, используемыми для указания порядковых номеров операндов операторов и параметров scp-программы. **"/>;**

пояснение —> **<** ***2_, текст пояснения_*** **: /"**Ключевые узлы ***1_, 2_*** и т.д. являются атрибутами, используемыми для указания порядковых номеров операндов операторов и параметров scp-программы. **"/>;**
главный синоним —> ***1_, 2_;***

4.5. Расширение ядра языка SCP

В подразделе 4.2 было рассмотрено ядро языка SCP, а в подразделе 4.3 – базовая абстрактная scp-машина. В данном подразделе рассматриваются направления и принципы эволюции языка SCP и scp-машины.

Пути эволюции операторов языка можно разделить на следующие два направления:

- новые операторы создаются путём композиции существующих операторов (genStr7, searchElStr7 и т.д.);
- новые операторы создаются как операции над операторами (различные call-операторы, wait-операторы).

При рассмотрении абстрактной scp-машины (см. подраздел 4.3) не затрагивались детально принципы управления параллельными процессами. При расширении абстрактной scp-машины для решения

задач управления параллельными процессами и также вытекающими из этого задачами обеспечения многопользовательской работы будем использовать следующие понятия:

- событие;
- сообщение;
- сегмент.

Рассмотрим подробнее каждое из понятий.

Событие

Пользовательская (с использованием явных языковых средств) синхронизация scp-процессов базируется на концепции событий. **Событием** является завершение формирования (генерации) некоторой (в общем случае произвольной) sc-конструкции в определенном контексте. Событие доступно (произошло) для scp-процесса, если и только если данный scp-процесс имеет соответствующие права доступа (например, на чтение) к sc-узлу, обозначающему контекст формирования события.

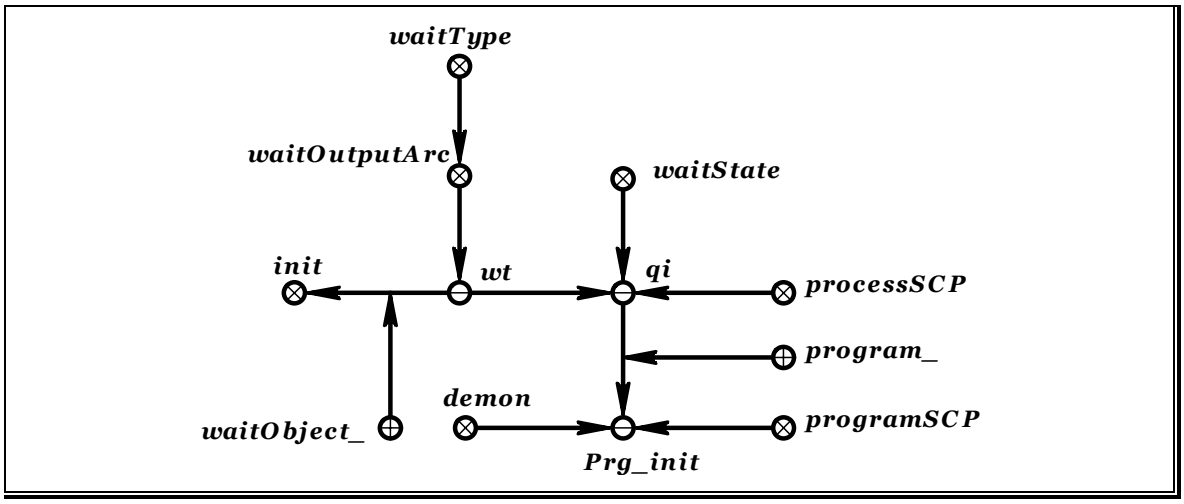
Программист может поручить системе автоматический запуск демонических программ, обрабатывающих интересующие его события по мере их возникновения или ожидать их возникновения путем применения операторов ***wait***. Операторами ***wait*** могут объявляться сложные события, состоящие в наступлении одного или всех заданных событий. Сложное событие в свою очередь может участвовать в образовании нового сложного события.

В одном и том же событии могут быть "заинтересованы" сразу несколько scp-процессов. В зависимости от приоритета, времени обращения и способа использования сведений о событии (монопольного или совместного) заказчики устанавливаются в очередь.

Определим как базовые события, возможность возникновения которых проверяется интерпретатором SCP при выполнении операций над любыми элементами sc-памяти, имеющими непустой список ассоциированных с ними событий (что отражается соответствующим образом в структуре системного графа, связанного с данным sc-элементом). Базовыми являются следующие классы событий:

- изменение количества входящих дуг данного sc-элемента (добавление дуги; удаление дуги; отсутствие входящих дуг);
- изменение количества выходящих дуг данного sc-элемента (добавление дуги; удаление дуги; отсутствие выходящих дуг);
- изменение значения содержимого указанного типа у данного sc-элемента (формирование содержимого; удаление содержимого; равенство числового содержимого нулю);
- появление связи (sc-дуги) заданного типа между двумя указанными sc-элементами;
- удаление данного sc-элемента в указанном сегменте;
- создание некоторого sc-элемента в указанном сегменте.

Приведем пример описания события и ожидание его каким-то scp-процессом:



Здесь sc-узел ***wt*** обозначает ожидание генерации выходящей дуги из sc-узла ***init*** scp-процессом ***qi***, который выполняет scp-программу ***Prg_init***.

Предлагаемый набор базовых событий вкупе с возможностью образования сложных событий накрывает возможности всех типовых механизмов синхронизации, используемых в современных операционных системах. Реализация событий встроенными механизмами интерпретатора позволяет использовать события для организации мониторинга (например, в отладочных целях) пользовательских процессов без изменения логики и эффективности их работы (особенно на мультипроцессорной аппаратной платформе).

Сообщение

Подчеркнем, что весь обмен информацией между прикладной системой и ее окружением (операционной системой, инструментальной средой разработки и т.д.) осуществляется только через сообщения. Под **сообщением** понимается sc-конструкция (событие), операционная (и частично денотационная) семантика которой известна (определена) в сегменте, общедоступном для компонентов системы, между которыми происходит обмен информацией.

Сегмент

Понятие сегмента в графодинамической ассоциативной памяти, схоже с понятие в традиционных архитектурах [324] (***Олифер В.Г.2001кн-СетевОС***), за исключением следующих особенностей:

- сегмент – это множество sc-элементов. Каждому sc-элементу соответствует один и только один сегмент;
- сегмент определяет права доступа для конкретного scp-процесса;
- для какого-то scp-процесса может быть доступно одновременно ограниченное количество сегментов. SCP-процесс может открывать (получать доступ) к сегментам динамически;
- при осуществления каких-либо преобразований в графодинамической памяти можно указать конкретный сегмент действия, например, сузить область поиска.

4.6. Использование scp-машины для интерпретации графодинамических моделей

Описать операционную семантику некоторого языка (языка-объекта) – это значит построить интерпретатор этого языка на некотором другом языке (метаязыке). Наш подход к описанию операционной семантики самых различных языков (языков знаний и языков программирования) в общем случае состоит из двух этапов.

На первом этапе осуществляется переход от различных (например, символьных) способов представления знаний, программ и данных для программ к их представлению в виде семантически эквивалентных sc-конструкций. Переход от конструктивных объектов произвольного вида, представляющих знания, программы и данные, к sc-конструкциям осуществляется путем уточнения денотационной семантики исходных конструктивных объектов. Следовательно, трансляция произвольным образом оформленных знаний и программ на язык SC не меняет смысла и набора операций, осуществляющих интерпретацию этих знаний и программ, а только меняет формальную (конструктивную) сторону реализации этих операций, превращая, в частности, некоторые нелокальные операции в локальные. В этом смысле, рассматривая трансляцию знаний, программ и данных на язык SC [342] (***ПредстИОЗвГАМ-2001кн***) следует говорить о частном виде трансляции – перекодировке, которая существенно проще, чем, например, трансляция с языка программирования высокого уровня на язык программирования более низкого уровня, осуществляющая переход на принципиально иную систему более "мелких" операций. Примером перекодировки программ является трансляция с языка ассемблера некоторой машины на ее внутренний язык. Переход от произвольных языков знаний, программ и данных к эквивалентным подъязыкам языка SC, сохраняющим суть (парадигму) организации операционной семантики транслируемых языков и меняющим только форму этой организации, не является сложной проблемой, особенно, если имеется опыт и соответствующие технологические приемы. На первом этапе описания операционной семантики различных языков осуществляется сведение всевозможных языков ко множеству графовых языков, являющихся подъязыками языка SC. Каждому такому графовому языку ставится в соответствие свой тип

абстрактной sc-машины. При этом, конечно, надо стремиться к тому, чтобы эти частные виды абстрактных sc-машин имели как можно больше общих операций.

Язык SC является удобной основой для представления (кодирования) не только структур данных самого различного вида, не только баз знаний различного вида, не только непроцедурных программ различного вида, но и всевозможных процедурных программ (процедур). Это дает возможность в абстрактных sc-машинах интерпретировать самые различные парадигмы программирования, а также интегрировать различные парадигмы программирования в рамках одной абстрактной sc-машины.

Таким образом, от проблемы описания операционной семантики произвольного языка осуществляется переход к проблеме описания (интерпретации) абстрактной sc-машины произвольного частного вида, т.е. абстрактной sc-машины с произвольным набором операций над sc-памятью. Абстрактные sc-машины произвольного вида удобнее всего интерпретировать на базе sc-машины определенного вида, которая обеспечивает реализацию программ, описывающих алгоритмы любых классов преобразований sc-конструкций (в том числе и классов преобразований, соответствующих операциям интерпретируемых абстрактных sc-машин). В качестве такой базовой sc-машины предлагается scp-машина.

Существенно подчеркнуть, что созданные на базе scp-машины интерпретаторы различных языков программирования могут эффективно взаимодействовать в ходе совместного решения общей задачи.

Переход от интерпретируемой абстрактной sc-машины определенного вида (т.е. с заданным набором операций) к интерпретирующей абстрактной scp-машине осуществляется следующим образом:

- путем дополнительного введения в sc-память scp-программ, обеспечивающих описание алгоритмов реализации операций интерпретируемой sc-машины. При этом множеству операций интерпретируемой sc-машины взаимно однозначно соответствует множество демонических scp-программ. Кроме демонических scp-программ в описание операций интерпретируемой sc-машины входит большое количество используемых ими пассивных подпрограмм. Перерабатываемыми данными для указанных scp-программ являются sc-конструкции, хранимые в памяти интерпретируемой абстрактной машины, например, программы или знания, представленные на интерпретируемом графовом языке. Подчеркнем при этом, что переход от интерпретируемой sc-машины к интерпретирующей scp-машине не требует никакого преобразования конструктивного объекта, хранимого в памяти интерпретируемой sc-машины;
- путем дополнительного использования в sc-памяти специальных sc-конструкций, описывающих текущее состояние выполняемых scp-процессов;
- путем замены набора операций интерпретируемой абстрактной sc-машины на набор операций абстрактной scp-машины.

Осуществляя такой переход, необходимо учитывать следующее:

- при параллельном выполнении scp-программ над общей sc-памятью неизбежно возникновение конфликтов. Например, пока одна scp-программа, найдя некоторое условие выполнения описываемой ею операции интерпретируемой sc-машины, продолжает работу, другая scp-программа может это условие разрушить. Такие конфликтующие scp-программы необходимо либо "разводить" во времени (переводя одну из них в состояние ожидания вплоть до завершения другой), либо разрешать возникающие конфликты путем уничтожения процесса выполнения одной из конфликтующих scp-программ. Во втором случае необходимо перевести перерабатываемую sc-память в состояние, которое возникло бы в случае, если бы уничтоженный процесс так и не был в свое время инициирован;
- необходимо предотвратить повторное выполнение одних и тех же операций интерпретируемой sc-машины над одними и теми же sc-конструкциями.

При решении перечисленных выше проблем важно учесть принципиальные особенности sc-машин, отличающие их от большинства других абстрактных машин переработки информации. Главной из этих особенностей является произвольная последовательность срабатывания операций. Учет этого обстоятельства позволяет решать вопросы, связанные со взаимодействием параллельных процессов в абстрактной sc-машине, гораздо более простыми способами, чем это делается в традиционных параллельных вычислительных системах.

При переходе от интерпретируемой абстрактной sc-машины к абстрактной scp-машине scp-процессы не всегда соответствуют одному элементарному процессу интерпретируемой sc-машины. Это связано с тем, что любую scp-программу, реализующую некоторую операцию интерпретируемой sc-машины, целесообразно (прежде всего по соображениям эффективности) программировать таким образом,

чтобы при однократном вызове такой программы (т.е. в результате однократного инициирования процесса ее реализации) соответствующая интерпретируемая операция выполнялась всюду, где для ее выполнения будут найдены условия. При этом процесс выполнения интерпретируемой операции через некоторое время распадается на несколько автономных процессов, каждый из которых выполняет эту операцию применительно к отдельному условию. Такие автономные scp-процессы уже однозначно соответствуют элементарным процессам интерпретируемой sc-машины. Различные процессы выполнения одной и той же операции конфликтуют между собой точно также, как и процессы выполнения различных операций. При разрешении возникающих конфликтов отдельный процесс может быть уничтожен совершенно независимо от того, что в это время будет происходить с другими процессами выполнения этой же интерпретируемой операции.

Проблема параллельного выполнения различных процессов может быть решена, если каждый из них будет формировать несколько множеств sc-элементов, доступ к которым со стороны других процессов оказывается ограниченным. Число типов ограничения доступа подлежит уточнению, но заранее можно выделить, по крайней мере, три типа:

- sc-элементы, входящие в состав sc-графа, являющегося условием выполнения процесса (s-блокированные элементы);
- sc-элементы, предварительно сгенерированные процессом (g-блокировка);
- sc-элементы, подлежащие удалению в случае успешного завершения процесса (e-блокировка).

Наличие sc-элементов двух последних типов определяется тем, что ни один процесс выполнения операции не вносит необратимых изменений в sc-память вплоть до своего завершения.

Конфликт между двумя процессами проявляется в том, что один и тот же sc-элемент (реже – два инцидентных sc-элемента) попадают в разные из вышеперечисленных множеств для двух разных процессов. В результате разрешения конфликта один из процессов может оказаться снятым с выполнения. В этом случае он обязан "собрать мусор", т.е. ликвидировать промежуточные результаты, возникшие в ходе его выполнения, удалить предварительно сгенерированные им элементы и освободить для свободного доступа со стороны других процессов все элементы, которые он "собирался удалить" в случае своего успешного завершения. Эта процедура алгоритмически почти не отличается от процедуры успешного завершения процесса, включающей "сборку мусора", удаление e-блокированных элементов и освобождение доступа к g-блокированным элементам.

Выше было указано, что программирование интерпретируемых операций на языке SCP требует, чтобы все промежуточные данные, возникающие при выполнении операций, представлялись в виде sc-конструкций.

Специфика интерпретации различных языков представления знаний на базе scp-машины заключается в том, что важное место в операциях переработки знаний занимают преобразования, связанные с изоморфизмом sc-конструкций, такие, как 1) поиск по образцу – поиск sc-конструкции, которая изоморфна заданной и для которой известно несколько sc-элементов, соответствующих указываемым элементам заданной sc-конструкции (образца), 2) копирование – генерация sc-конструкции, которая изоморфна заданной и для которой известно несколько sc-элементов, соответствующих указываемым элементам заданного образца. Такие преобразования занимают важное место не только для языков представления знаний, но и для любых языков, построенных на базе семантических сетей.

Поиск по образцу и копирование имеют большое количество модификаций. Так, например, при поиске по образцу а) можно ограничивать область поиска, т.е. искать sc-конструкции, являющиеся подмножествами (фрагментами) заданной sc-конструкции – примером такого ассоциативного поиска является поиск sc-конструкций в области констант некоторого знания, т.е. поиск в области фактографических знаний, б) можно говорить о поиске изоморфных высказываний, удовлетворяющих заданному высказыванию в рамках некоторого (заданного) знания, – такой поиск, например, осуществляется при поиске ответов на информационно-поисковые запросы, в) можно говорить о поиске изоморфных высказываний, удовлетворяющих правым частям (следствиям) импликативных высказываний, входящих в состав заданного знания, – такой поиск осуществляется при реализации некоторых операций сведения заданий к подзаданиям.

Язык SCP является достаточно хорошо приспособленным к реализации преобразований, связанных с изоморфизмом sc-конструкций, благодаря ориентации языка SCP на переработку графовых конструкций, наличию в языке SCP операторов ассоциативного поиска, параллелизму.

Выводы к разделу 4

Язык SCP в графодинамических ассоциативных машинах занимает особое место, поскольку его удобство и эффективность во многом определяют эффективность реализуемых с его помощью графодинамических моделей.

Принципиальным отличием языка SCP от известных языков программирования является ориентация на графодинамическую (структурно перестраиваемую) ассоциативную память.