# **Введение**

Основной целью данной курсовой работы является разработка транслятора для языка программирования DVV-2018. Главной задачей для транслятора будет сделать язык DVV-2018 понятным для вычислительной машины. В данном курсовом проекте будет происходить трансляция кода на языке DVV-2018 в код на языке ассемблер.

Для данного курсового проекта были определены следующие задачи:

* разработка спецификации языка программирования;
* разработка структуры транслятора;
* разработка лексического анализатора;
* разработка синтаксического анализатора;
* разработка семантического анализатора;
* обработка выражений;
* генерация кода в Assembler;
* тестирование транслятора.

Решение для каждой из поставленных задач будут отображены в следующих главах курсового проекта:

* спецификация языка программирования;
* структура транслятора;
* лексический анализатор;
* синтаксический анализатор;
* семантический анализатор;
* преобразование выражений;
* генерация кода;
* тестирование транслятора.

Язык программирования DVV-2018 предназначается для выполнения простейших арифметический и логических действий , а также для операций над строками.

**Глава 1 Спецификация языка программирования**

**1.1 Характеристика языка программирования**

Язык DVV-2018 – это процедурный, универсальный, строготипизированный, компилируемый язык. Так же язык не является объектно-ориентированным.

* универсальный — язык, система терминов, определенных строго и однозначно, а потому допускающих над собой чисто формальные операции.
* процедурный – язык при программировании на котором последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода, с помощью механизмов самого языка.
* сильной(строгой) типизацией в языке является отсуствия приведения типов
* компилируемый – язык который конвертируется в машинный код.

**1.2 Алфавит языка**

Алфавит языка DVV-2018 основан на кодировке Windows-1251, представленной на рисунке 1.1. Исходный код DVV-2018 может содержать символы латинского алфавита, цифры десятичной системы счисления от 0 до 9, символы русского алфавита можно использовать только в строковых литералах. Код, написанный на языке DVV-2018, может содержать символы : +, -, \*, /, {, }, (, ), ', >, <, =, : .

**1.3 Символы сепараторы**

Символы, которые являются сепараторами в языке DVV-2018 представлены в таблице 1.1.

Таблица 1.1 – Сепараторы

|  |  |
| --- | --- |
| Сепаратор | Назначение |
| (space)  (\n)  .  , | разделение инструкций |
| {  } | программный блок |
| (  ) | параметры/приоритетность операций (в выражениях) |

**1.4 Применяемы кодировки**

Для написания исходного кода на языке программирования DVV-2018 используется кодировка Windows-1251. Кодировка Windows-1251 представлена на рисунке 1.1



Рисунок 1.1 – Алфавит входных символов

**1.5 Типы данных**

В языке DVV-2018 есть 3 типа данных: целочисленный integer, строковый string и логический bool. Описание типов данных, предусмотренных данным языком представлены в таблице 1.2.

Таблица 1.2 – Типы данных языка DVV-2018

|  |  |
| --- | --- |
| Тип данных | Описание типа данных |
| integer | Фундаментальный тип данных. Предусмотрен для объявления целочисленных положительных данных.  Автоматически инициализируется нулевым значением.  Возможные операции:  + – суммирование;  - – вычитание;  \* – умножение;  / – деление;  = – присваивание значения; |
| String | Фундаментальный тип данных. Предусмотрен для объявления строк. Автоматическая инициализируется строкой нулевой длины. Максимальное количество символов в строке – 255. |
| Bool | Фундаментальный тип данных. Предусмотрен для использования в конструкции if() else. Может быть инициализирован 2 значениями truth или false. Автоматическая инициализация не предусмотрена.  Возможные операции:  == , < , > – логические операторы сравнения. |

**1.6 Преобразование типов данных**

Так как язык DVV-2018 является строготипизированным , то по определению

сильной(строгой) типизации преобразование(приведение) типов в языке отсутствует.

**1.7 Идентификаторы**

В языке DVV-2018 для идентификатора можно использовать строчные буквы латинского алфавита. Максимальная длина идентификатора 4096 символов.

**1.8Литералы**

Для языка DVV-2018 характерно присутствие литералов 3 типов: литералы целого типа, строковые литералы, а также литералы типа bool. Описание данных литералов можно прочитать ниже в таблице 1.3.

Таблица 1.3 – Описание литералов

|  |  |
| --- | --- |
| Тип литерала | Описание |
| Литералы целого типа | Целочисленные неотрицательные литералы, инициализируются 0. Литералы только могут быть только rvalue. Диапазон изменения от 0 до 4294967296. |
| Строковые литералы | Символы, заключённые в " " (двойные кавычки), инициализируются пустой строкой, строковые переменные. Литералы могут быть только rvalue. |
| Логические литералы | Логические литералы могут быть инициализированы только двумя значениями truth или false. |

**1.9 Область видимости идентификаторов**

Область видимости «сверху вниз». В языке DVV-2018 требуется обязательное объявление переменной перед её использованием. Все переменные должны располагаться после точки входа в приложение. Возможность объявления переменных c одинаковым идентификаторами отсутствует.

**1.10 Инициализация данных**

Таблица 1.4 – Инициализация данных

|  |  |
| --- | --- |
| Вид инициализации | Примечание |
| <тип данных><идентификатор>; | Автоматическая инициализация: переменные типа integer инициализируются нулём, переменные типа string – пустой строкой, переменные типа bool не имеют автоматической инициализации. |
| <идентификатор> : <значение>; | Присваивание переменной значения. |

**1.11 Инструкции языка**

Все возможные инструкции языка программирования DVV-2018 представлены в общем виде в таблице 1.5.

Таблица 1.5 – Инструкции языка программирования DVV-2018

|  |  |
| --- | --- |
| Инструкция | Запись на языке DVV-2018 |
| Объявление переменной | <тип данных> <идентификатор>; |
| Присваивание | <идентификатор> : <значение>/<идентификатор>; |
| Блок инструкций | [… ] |
| Вывод данных | print (<идентификатор> / <литерал>); |

**1.12 Операции языка**

Язык программирования DVV-2018 может выполнять арифметические и логические операции, представленные в таблице 1.5.

Таблица 1.6 – Приоритетности операций языка программирования DVV-2018

|  |  |
| --- | --- |
| Операция | Приоритетность операции |
| (  ) | 0 |
| , | 1 |
| \*  / | 3 |
| +  - | 2 |
| = , > , < | Данные операции не имеют приоритета |

Максимальным значением приоритетности является “3”, минимальным “0” соответственно.

**1.13 Выражения и их вычисления**

Выражения могут содержать вызов библиотечных функций. Допускаются выражения которые вычисляют значение из операндов. Вычисление в выражения происходит по приоритетности операций языка.

**1.14 Программные конструкции языка**

Ключевые программные конструкции языка программирования DVV-2018 представлены в таблице 1.7.

Таблица 1.7 – Программные конструкции языка DVV-2018

|  |  |
| --- | --- |
| Точка входа в приложение | dvv  [  …  ] |
| Блок if() else | if(проверяемое условие) […]  else […] |

**1.15 Область видимости**

В языке DVV-2018 переменные находятся внутри одной области видимости расположенной внутри блока dvv. Объявление пользовательских областей видимости не предусмотрено.

**1.16 Семантические проверки**

Таблица с перечнем семантических проверок, предусмотренных языком, приведена в таблице 1.8.

Таблица 1.8 – Семантические проверки

|  |  |
| --- | --- |
| Номер | Правило |
| 1 | Тип данных передаваемых значений в функцию стандартной библиотеки должен соответствовать заявленному. |
| 2 | Объявление идентификатора должно происходить до его использования. |
| 3 | В арифметических выражениях операнды должны иметь один тип данных. |

**1.17 Распределение оперативной памяти на этапе выполнения**

Размещение всех переменных происходит в куче. Распределение оперативной памяти происходит на этапе генерации. Таблица лексем и таблица идентификаторов сохраняются в структуры с выделенной динамической памятью, которая очищается по окончанию работы транслятора.

**1.18 Стандартная библиотека и её состав**

Функции стандартной библиотеки с описанием представлены в таблице 1.9. Стандартная библиотека написана на языке программирования C++.

Таблица 1.9 – Состав стандартной библиотеки

|  |  |  |  |
| --- | --- | --- | --- |
| Имя функции | Возвращаемое значение | Принимаемые параметры | Описание |
| Strlen | Integer | str s - строка | Возвращает длину строки s |
| Substr | String | string x – строка,  string y – строка | Возвращает значение либо throw либо false в зависимости от того является ли строка 2 подстрокой строки 1. |

**1.19 Ввод и вывод данных**

Ввод данных не предусмотрен в языке DVV-2018.А за вывод данных отвечает функция: print (<идентификатор или литерал>);

**1.20 Точка входа**

Точкой входа в языке DVV-2018 является – ключевое слово dvv.

**1.21 Препроцессор**

Препроцессор для языка DVV-2018 не предусмотрен.

**1.22 Соглашение о вызовах**

В языке вызов функций происходит по соглашению о вызовах stdcall. Выбор пал на это соглашение о вызовах так как это stdcall является одним из самых популярных соглашениях о вызовах.

Особенности stdcall:

* параметры передаются через стек;
* запись в стек идет справа налево
* очистку стека производит вызываемый код

**1.23 Объектный код**

Код на языке DVV-2018 транслируется в код на языке ассемблера.

**1.24 Классификация сообщений транслятора**

При возникновении ошибки в коде программы на языке DVV-2018 и выявления её транслятором в файл протокола выводится сообщение. Классификация сообщений приведена в таблице 1.10.

Таблица 1.10 – Классификация сообщений транслятора

|  |  |
| --- | --- |
| Интервал | Описание ошибок |
| 100,104 | Ошибки параметров |
| 107-112 | Ошибки открытия и чтения файлов |
| 113-121 | Ошибки лексического анализа |
| 600-607 | Ошибки синтаксического анализа |
| 700-799 | Ошибки семантического анализа |

## **1.25 Контрольный пример**

Пример программы на языке DVV-2018 представлен ниже:

dvv

[

integer x : 20.

integer y : 10.

integer l,z,q,w,m.

string str : 'mystring'.

string mystr : 'mynewstring'.

bool mb.

mb : false.

if (x > y)

[

integer p.

p : x + y.

print (p).

]

else

[

l : strlen(str).

]

print (mb).

z : (x + y) \* (x - y).

bool nb : substr(mystr,str).

print (mystr).

return z.

]

# **Глава 2 Структура транслятора**

## **2.1 Компоненты транслятора их назначение и принципы взаимодействия**

Транслятор – обслуживающая программа, преобразующая исходную программу, предоставленную на входном языке программирования, в рабочую программу, представленную на объектном языке. Транслятор включает в себя такие компоненты как лексический анализатор, синтаксический анализатор, генератор объектного кода.

Лексический анализатор:

* выполняет аналитический разбор входной последовательности символов на распознанные группы — лексемы.
* входным параметром является исходный код.
* выходными параметрами являются таблица лексем и таблица идентификаторов.

Синтаксический анализатор:

* выполняет сопоставление линейной последовательности лексем естественного или формального языка с его формальной грамматикой.
* входным параметром является таблица лексем.
* выходным параметром является дерево разбора или иначе говоря синтаксическое дерево

Семантический анализатор:

* выполняет последовательности действий алгоритма автоматического понимания текстов, заключающегося в выделении семантических отношений, формировании семантического представления текстов.
* входными параметрами являются таблица идентификаторов и дерево разбора.
* выходными параметрами является семантическая структура

Генератор объектного кода:

* занимается построением для программы на входном языке эквивалентной машинной программы.
* входным параметром является промежуточное представление программы.
* выходным параметром является объектный код.

Подробное описание компонентов транслятора можно найти в главах курсового проекта под названием:

* лексический анализатор
* синтаксический анализатор
* семантический анализатор
* генерация кода.

## **2.2 Перечень входных параметров транслятора**

Перечень входных параметров для языка DVV-2018 представлен в таблице 2.1

Таблица 2.1

|  |  |
| --- | --- |
| Имя входного параметра | Выполняемая функция |
| -in | Указывает на файл из которого транслятор будет брать код |
| -log | Указывает на файл в который будет выводиться протокол работы. |
| -id | Указывает на файл в который будет выводиться таблицу идентификаторов |
| -lx | Указывает на файл в который будет выводиться таблица лексем |
| -rl | Указывает на файл в который будет выводиться дерево разбора или иначе говоря синтаксическое дерево |

## **2.3 Протоколы, формируемые транслятором**

Для языка DVV-2018 предусмотрено формирование трех протоколов представленных в таблице 2.2

Таблица 2.2- Протоколы формируемые транслятором

|  |  |
| --- | --- |
| Имя входного протокола | Информация которая содержится в протоколе |
| <Имяфайла>.txt.log | Ошибки, если они есть, иначе будет выводится, что тест выполнен без ошибок. Дата создания протокола.  Перечень входных параметров языка DVV-2018.  Исходные данные: Количество символов, Количество строк. |
| <Имяфайла>.txt.id | Таблица идентификаторов. Дата создания протокола. |
| <Имяфайла>.txt.lx | Таблица лексем. Дата создания протокола. |
| <Имяфайла>.txt.rl | Дерево разбора(Синтаксическое дерево)ю Дата создания протокола. |

# **Глава 3 Лексический анализатор**

## **3.1 Структура лексического анализатора**

Лексический анализатор представляет собой программу, которая транслирует исходную программу в набор символ, которая формирует таблицу лексем и таблицу идентификаторов, записывая в таблицы полученную информацию об идентификаторах и литералах.

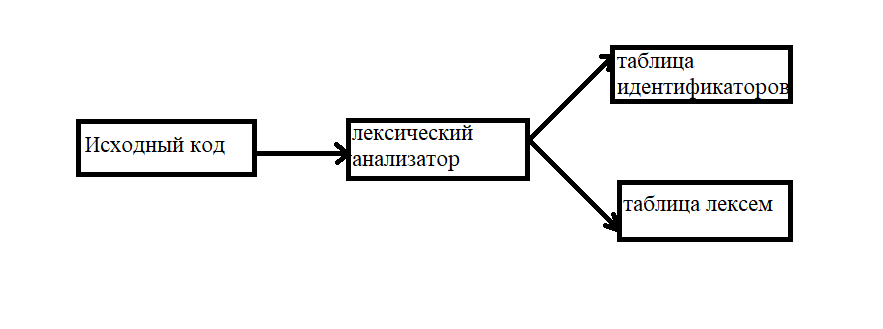


Рис. 3.1 Структура лексического анализатора

## **3.2 Контроль входных символов**

## Таблица предназначеная для контроля входных символов представлена на рисунке 3.2.

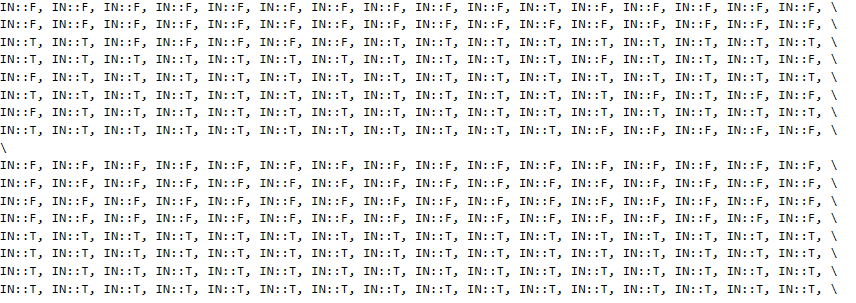


Рис 3.2 Входная таблица символов языка DVV-2018

Принцип работы таблицы состоит в следующем: каждому элементу в таблице соответствует значение в 16 системе счисления такой же, как и в кодовой таблице ASCII. Допустимые символы на рисунке обозначены буквой - T, запрещенные – F.

**3.3 Удаление избыточных символов**

Алгоритм удаления избыточных символов(пробел и переход на новую строку) происходит путем их игнорирования. Если при считывании из файла в буфер попадают избыточные символы ,то сначала проверяется не входят ли данные символы в строковый литерал, а иначе процесс сравнивания содержимого буфера с конечными автоматами пропускается.

## **3.4 Перечень ключевых слов**

Перечень ключевых слов, сепараторов, конечных автоматов языка DVV-2018 представлен в приложении А.

**3.5 Основные структуры данных**

Основными структурами данных лексического анализатора являются таблица лексем и таблица идентификаторов.

Структуры таблицы лексем:

struct Entry//строка таблицы лексем

{

char lexema; // лексема

int sn;//номер строки в тексте

int indxTI;//индекс в таблице идентификаторов или LT\_TI\_NULLIDX

};

struct LexTable//экземпяр таблицы лексем

{

int maxsize;//емкость таблицы лексем( < LT\_MAXSIZE)

int size;//текущий размер таблицы лексем ( < maxsize)

Entry\* table;//массив строк таблицы лексем

};

Структуры таблицы идентификаторов:

struct Entry

{

short idxfirstLE;//индекс первой строки в таблице лексем

char id[ID\_MAXSIZE \* 2];//идентификатор

IDDATATYPE iddatatype;//тип данных

IDTYPE idtype;//тип идентификатора

union

{

char vbool[6];//значение bool

int vint;//значение инт

struct

{

int len;//длина строки стр

char str[TI\_STR\_MAXSIZE - 1];//строка

}vstr[TI\_STR\_MAXSIZE];//значение стр

}value;//значение идентификатора

};

struct IdTable//экземпляр таблицы идентификаторов

{

int maxsize;//макс емкость таблицы идентификаторов( < TI\_MAXSIZE)

int size;//текущий размер таблицы идентификаторов ( < maxsize)

Entry\* table;//массив строк таблицы идентификаторов

};

## **3.6. Структура и перечень сообщений лексического анализатора**

Для лексического анализатора языка DVV-2018 в таблице ошибок зарезервированы ошибки с 113 по 121. Структура ошибок одинакова для всех частей транслятора при возникновении ошибки в консоль выводится:

* номер ошибки;
* сообщение об ошибке;
* строка где возникла ошибка;
* позиция в строке, где возникла ошибка.

В таблице 3.1 представлены номера и текст ошибок лексического анализатора.

Таблица 3.1 – Перечень ошибок лексического анализатора

|  |  |
| --- | --- |
| Номер ошибки | Выводимое сообщение |
| 113 | Превышен максимальный размер таблицы лексем |
| 114 | Превышен максимальный размер таблицы идентификаторов |
| 115 | Недопустимы номер возвращаемой строки таблицы лексем |
| 117 | Отсутствует таблица лексем |
| 118 | Отсутствует таблица идентификаторов |
| 119 | Недопустимый номер возвращаемой строки таблицы идентификаторов |
| 120 | Размер длины строки превышен(256) |
| 121 | Превышено максимальное значение целочисленного литерала |

## **3.7. Принцип обработки ошибок**

При обнаружении в коде ошибки лексический анализатор прекращает свою работу и выводит в консоль одну из ошибок представленных в таблице 3.1. В языке задан лимит на количество ошибок для лексического анализа их количество равно 8.

## **3.8 Параметры лексического анализатора**

Язык DVV-2018 допускает использование параметров для управления работой лексического анализатора. Принцип их использования описан в табл. 2.1.

**3.9. Алгоритм лексического анализа**

Алгоритм лексического анализа представлен ниже:

* из исходно файла подается текст пока в тексте не будет встречен один из символов сепараторов.
* при достижении символа сепаратора буфер усекается на нем.
* если это был сепаратор который не является пробелом или переходом на другую строку то он добавляется во второй буфер.
* дальше идет проверка текста записанного в первый буфер на то совпадает он с конечным автоматом.
* при совпадении происходит запись лексемы соответствующей совпавшему автомату в таблицу лексем
* если совпавший автомат был идентификатором или лексемой он также заносится в таблицу идентификаторов
* после этого производится проверка на совпадение записанного во второй буфер.
* при совпадении происходит запись лексемы соответствующей совпавшему автомату в таблицу лексем.

**3.10 Контрольный пример**

Результат работы Лексического Анализа на основе контрольного примера представленного в пункте 1.25 расположены ниже.

Таблица Лексем:

0. d

1. [

2. ti:l.

3. ti:l.

4. ti,i,i,i,i.

5. ti:l.

6. ti:l.

7. ti.

8. i:l.

9. w(i+i)

10. [

11. ti.

12. i:i+i.

13. p(i).

14. ]

15. e

16. [

17. i:f(i).

18. ]

19. p(i).

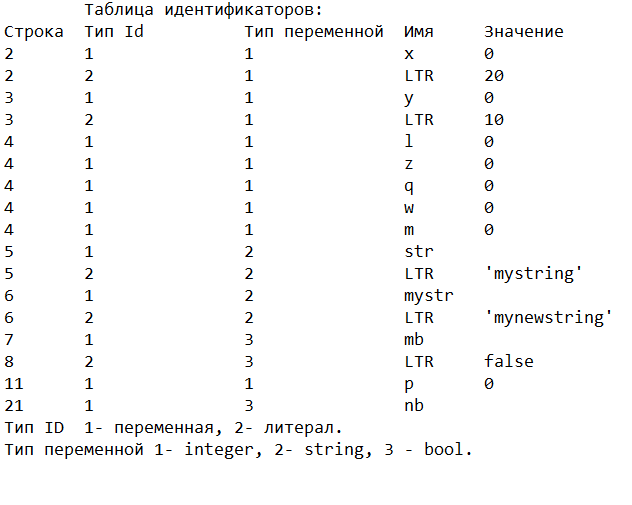
20. i:(i+i)\*(i-i).

21. ti:f(i,i).

22. p(i).

23. ri.

Рис 3.2 Таблица идентификаторов



# **Глава 4 Синтаксический анализатор**

## **4.1 Структура синтаксического анализатора**

Синтаксический анализатор представляет собой программу, проверяющую таблицу лексем по определенным правилам грамматики Грейбах. При удачной проверке таблицы лексем – формирует протокол работы, представляющий из дерево разбора или синтаксическое дерево.

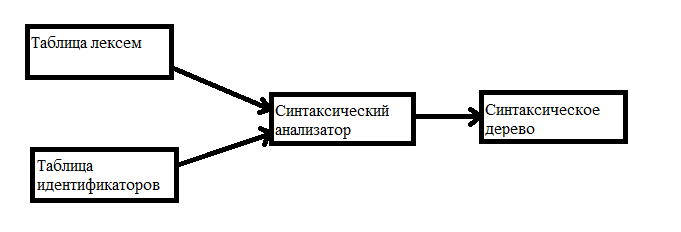


Рис. 4.1 Структура синтаксического анализатора

## **4.2. Контекстно-свободная грамматика, описывающая синтаксис языка**

Формальная грамматика или просто грамматика в теории формальных языков — способ описания формального языка, то есть выделения некоторого подмножества из множества всех слов некоторого конечного алфавита. Различают порождающие и распознающие (или аналитические) грамматики — первые задают правила, с помощью которых можно построить любое слово языка, а вторые позволяют по данному слову определить, входит ли оно в язык или нет.

Стартовым символом для грамматики используемой в языке DVV-2018 является нетерминал S. А правила перехода нетерминальных автоматов указаны в таблице 4.1.

Таблица 4.1 – Таблица правил перехода нетерминальных автоматов

|  |  |  |
| --- | --- | --- |
| Нетерминалы | Правила | Назначение правил |
| S→ | d[N] | Определение правильности написания общей структуры программы |
| N→ | ti:E.N  ti.N  tE.N  ti:P.N  i:P.N  i:E.N  w(K)[C]e[C]N  w(K)[C]N  p(E).N  p(E).  rE.  ri:P. | Определение правильности написания инструкций и конструкций языка. |
| E→ | i  l  i,E  f(E)  f(E)E | Определение правильности написания операции инициализации |
| P→ | i  l  iMP  lMP  (P)  (P)MP  (P)P | Определение правильности написания выражения |
| C→ | i:E.  i:E.C  i:P.  i:P.C  ti:E.C  ti:E.  ti:P.C  ti:P.  ti.C  ti.  p(E).C  p(E). | Определение правильности написания инструкций в теле конструкции языка if() else |
| M→ | +  -  \*  / | Определение правильности написания арифметических операторов в выражении |
| B→ | >  <  = | Определение правильности написания логических операторов в сравнении |
| K→ | iBi  lBl  f(E)Bf(E)  f(E)  l | Определение правильности написания в условии конструкции языка if() else |

## **4.3 Построение конечного магазинного автомата**

В теории автоматов, автомат с магазинной памятью — это конечный автомат, который использует стек для хранения состояний. Память работает как стек, то есть для чтения доступен последний записанный в неё элемент. Таким образом, функция перехода является отображением. То есть, по комбинации текущего состояния, входного символа и символа на вершине магазина автомат выбирает следующее состояние и, возможно, символ для записи в магазин. В случае, когда в правой части автоматного правила присутствует, в магазин ничего не добавляется, а элемент с вершины стирается. Если магазин пуст, то срабатывают правила в левой части.

Формальное описание конечного магазинного автомата для грамматики представлено на Рис 4.2.

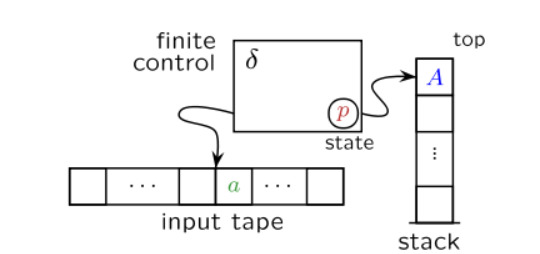


Рис 4.2. Формальное описание конечного магазинного автомата для грамматики

**4.4Основные структуры данных**

Основные структуры данных для синтаксического анализа в языке DVV-2018 будут представлены ниже.

Структура Грейбах:

struct Greibach

{

short size;

GRBALPHABET startN;

GRBALPHABET stbottomT;

Rule\* rules;

Greibach() { short size = 0; startN = 0; stbottomT = 0; rules = 0; }

Greibach(GRBALPHABET pstartN, GRBALPHABET pstbottomT, short psize, Rule r, ...);

short getRule(GRBALPHABET pnn, Rule& prule);

Rule getRule(short n);

};

Структура правила:

struct Rule

{

GRBALPHABET nn;

int iderror;

short size;

struct Chain

{

short size;

GRBALPHABET\* nt;

Chain() { size = 0; nt = 0; }

Chain(short psize, GRBALPHABET s, ...);

char\* getCChain(char\* b);

static GRBALPHABET T(char t) { return GRBALPHABET(t); }

static GRBALPHABET N(char n) { return -GRBALPHABET(n); }

static bool isT(GRBALPHABET s) { return s > 0; }

static bool isN(GRBALPHABET s) { return !isT(s); }

static char alphabet\_to\_char(GRBALPHABET s) { return isT(s) ? char(s) : char(-s);}

}\*chains;

Rule() { nn = 0x00; size = 0; }

Rule(GRBALPHABET pnn, int iderror, short psize, Chain c, ...);

char\* getCRule(char\* b, short nchain);

short getNextChain(GRBALPHABET t, Rule::Chain& pchain, short j);

};

**4.5 Описание алгоритма синтаксического разбора.**

Принцип работы синтаксического разбора для языка DVV-2018 приведен ниже:

* В магазин записывается стартовый символ.
* На основе полученной таблицы лексем формируется входная лента.
* Запускается автомат и выбирается цепочка, соответствующая нетерминальному символу, записывается в магазин в обратном порядке.
* Если терминалы в стеке и в ленте совпадают, то данный терминал удаляется с ленты и стека. Иначе возвращаемся в предыдущее сохраненное состояние и выбираем другую цепочку нетерминала.
* Если в магазине встретился нетерминал, переходим к пункту 3.
* Если символ достиг символа дна стека, и лента в этот момент имеет символ дна стека, то синтаксический анализ выполнен успешно. Иначе генерируется ошибка.

**4.6 Структура и перечень сообщений синтаксического анализатора**

Для синтаксического анализатора языка DVV-2018 в таблице ошибок зарезервированы ошибки с 600 по 607. Структура ошибок одинакова для всех частей транслятора при возникновении ошибки в консоль выводится:

* Номер ошибки
* Сообщение об ошибке
* Строка где возникла ошибка

В таблице 4.2 представлены номера и текст ошибок лексического анализатора.

Таблица 4.2

|  |  |
| --- | --- |
| Номер ошибки | Выводимое сообщение |
| 600 | Неверная структура программы |
| 601 | Ошибка в конструкции языка |
| 602 | Ошибка инициализации |
| 603 | Ошибка выражения |
| 604 | Ошибка в теле if |
| 605 | Ошибка знака в выражении |
| 606 | Ошибка знака в сравнении |
| 607 | Ошибка в условии if |

**4.7 Параметры синтаксического анализатора и режимы его работы**

Язык DVV-2018 допускает использование параметров для управления работой синтаксического анализатора. Принцип их использования описан в табл. 2.1.

**4.8. Принцип обработки ошибок**

При обнаружении в коде ошибки синтаксический анализатор прекращает свою работу и выводит в консоль одну из ошибок представленных в таблице 4.2. В языке

DVV-2018 для синтаксического анализа допустимо использование только 8 ошибок.

**4.9. Контрольный пример**

Распечатка дерева разбора предоставлено в приложении B.

**Приложение А**

**Ключевые слова и соответствующие им конечные автоматы**

|  |  |
| --- | --- |
| Ключевое слово | Конечный автомат |
| integer | #define FST\_INTEGER 8, \  FST::NODE(1, FST::RELATION('i', 1)),\  FST::NODE(1, FST::RELATION('n', 2)),\  FST::NODE(1, FST::RELATION('t', 3)),\  FST::NODE(1, FST::RELATION('e', 4)),\  FST::NODE(1, FST::RELATION('g', 5)),\  FST::NODE(1, FST::RELATION('e', 6)),\  FST::NODE(1, FST::RELATION('r', 7)),\  FST::NODE() |
| string | #define FST\_STRING 7, \  FST::NODE(1, FST::RELATION('s', 1)),\  FST::NODE(1, FST::RELATION('t', 2)),\  FST::NODE(1, FST::RELATION('r', 3)),\  FST::NODE(1, FST::RELATION('i', 4)),\  FST::NODE(1, FST::RELATION('n', 5)),\  FST::NODE(1, FST::RELATION('g', 6)),\  FST::NODE() |
| bool | #define FST\_BOOL 5, \  FST::NODE(1, FST::RELATION('b', 1)),\  FST::NODE(1, FST::RELATION('o', 2)),\  FST::NODE(1, FST::RELATION('o', 3)),\  FST::NODE(1, FST::RELATION('l', 4)),\  FST::NODE() |
| return | #define FST\_RETURN 7, \  FST::NODE(1, FST::RELATION('r', 1)),\  FST::NODE(1, FST::RELATION('e', 2)),\  FST::NODE(1, FST::RELATION('t', 3)),\  FST::NODE(1, FST::RELATION('u', 4)),\  FST::NODE(1, FST::RELATION('r', 5)),\  FST::NODE(1, FST::RELATION('n', 6)),\  FST::NODE() |
| print | #define FST\_PRINT 6, \  FST::NODE(1, FST::RELATION('p', 1)),\  FST::NODE(1, FST::RELATION('r', 2)),\  FST::NODE(1, FST::RELATION('i', 3)),\  FST::NODE(1, FST::RELATION('n', 4)),\  FST::NODE(1, FST::RELATION('t', 5)),\  FST::NODE() |
| define | #define FST\_DVV 4, \  FST::NODE(1, FST::RELATION('d', 1)),\  FST::NODE(1, FST::RELATION('v', 2)),\  FST::NODE(1, FST::RELATION('v', 3)),\  FST::NODE() |
| if | #define FST\_IF 3, \  FST::NODE(1, FST::RELATION('i',1)),\  FST::NODE(1, FST::RELATION('f',2)),\  FST::NODE() |
| else | #define FST\_ELSE 5, \  FST::NODE(1, FST::RELATION('e', 1)),\  FST::NODE(1, FST::RELATION('l', 2)),\  FST::NODE(1, FST::RELATION('s', 3)),\  FST::NODE(1, FST::RELATION('e', 4)),\  FST::NODE() |
| strlen | #define FST\_STRLEN 7,\  FST::NODE(1, FST::RELATION('s',1)),\  FST::NODE(1, FST::RELATION('t',2)),\  FST::NODE(1, FST::RELATION('r',3)),\  FST::NODE(1, FST::RELATION('l',4)),\  FST::NODE(1, FST::RELATION('e',5)),\  FST::NODE(1, FST::RELATION('n',6)),\  FST::NODE() |
| substr | #define FST\_SUBSTR 7,\  FST::NODE(1, FST::RELATION('s',1)),\  FST::NODE(1, FST::RELATION('u',2)),\  FST::NODE(1, FST::RELATION('b',3)),\  FST::NODE(1, FST::RELATION('s',4)),\  FST::NODE(1, FST::RELATION('t',5)),\  FST::NODE(1, FST::RELATION('r',6)),\  FST::NODE() |
| <id> | #define FST\_ID 2, \  FST::NODE(52, \  FST::RELATION('a', 0), FST::RELATION('b', 0), FST::RELATION('c', 0), FST::RELATION('d', 0), FST::RELATION('e', 0), FST::RELATION('f', 0),\  FST::RELATION('g', 0), FST::RELATION('h', 0), FST::RELATION('i', 0), FST::RELATION('j', 0), FST::RELATION('k', 0), FST::RELATION('l', 0),\  FST::RELATION('m', 0), FST::RELATION('n', 0), FST::RELATION('o', 0), FST::RELATION('p', 0), FST::RELATION('q', 0), FST::RELATION('r', 0),\  FST::RELATION('s', 0), FST::RELATION('t', 0), FST::RELATION('u', 0), FST::RELATION('v', 0), FST::RELATION('w', 0), FST::RELATION('x', 0),\  FST::RELATION('y', 0), FST::RELATION('z', 0),\  \  FST::RELATION('a', 1), FST::RELATION('b', 1), FST::RELATION('c', 1), FST::RELATION('d', 1), FST::RELATION('e', 1), FST::RELATION('f', 1),\  FST::RELATION('g', 1), FST::RELATION('h', 1), FST::RELATION('i', 1), FST::RELATION('j', 1), FST::RELATION('k', 1), FST::RELATION('l', 1),\  FST::RELATION('m', 1), FST::RELATION('n', 1), FST::RELATION('o', 1), FST::RELATION('p', 1), FST::RELATION('q', 1), FST::RELATION('r', 1),\  FST::RELATION('s', 1), FST::RELATION('t', 1), FST::RELATION('u', 1), FST::RELATION('v', 1), FST::RELATION('w', 1), FST::RELATION('x', 1),\  FST::RELATION('y', 1), FST::RELATION('z', 1)),\  FST::NODE() |
| <intliteral> | #define FST\_INTLIT 2, \  FST::NODE(20, \  FST::RELATION('1', 0), FST::RELATION('2', 0), FST::RELATION('3', 0), FST::RELATION('4', 0), FST::RELATION('5', 0), FST::RELATION('6', 0),\  FST::RELATION('7', 0), FST::RELATION('8', 0), FST::RELATION('9', 0), FST::RELATION('0', 0),\  \  FST::RELATION('1', 1), FST::RELATION('2', 1), FST::RELATION('3', 1), FST::RELATION('4', 1), FST::RELATION('5', 1), FST::RELATION('6', 1),\  FST::RELATION('7', 1), FST::RELATION('8', 1), FST::RELATION('9', 1), FST::RELATION('0', 1)),\  FST::NODE() |
| <stringliteral> | #define FST\_STRLIT 4, \  FST::NODE(2, FST::RELATION('\'', 1), FST::RELATION('\'', 2)),\  FST::NODE(158, \  FST::RELATION('a', 1), FST::RELATION('b', 1), FST::RELATION('c', 1), FST::RELATION('d', 1), FST::RELATION('e', 1), FST::RELATION('f', 1),\  FST::RELATION('g', 1), FST::RELATION('h', 1), FST::RELATION('i', 1), FST::RELATION('j', 1), FST::RELATION('k', 1), FST::RELATION('l', 1),\  FST::RELATION('m', 1), FST::RELATION('n', 1), FST::RELATION('o', 1), FST::RELATION('p', 1), FST::RELATION('q', 1), FST::RELATION('r', 1),\  FST::RELATION('s', 1), FST::RELATION('t', 1), FST::RELATION('u', 1), FST::RELATION('v', 1), FST::RELATION('w', 1), FST::RELATION('x', 1),\  FST::RELATION('y', 1), FST::RELATION('z', 1), FST::RELATION('1', 1), FST::RELATION('2', 1), FST::RELATION('3', 1), FST::RELATION('4', 1),\  FST::RELATION('5', 1), FST::RELATION('6', 1), FST::RELATION('7', 1), FST::RELATION('8', 1), FST::RELATION('9', 1), FST::RELATION('0', 1),\  \  FST::RELATION('а', 1), FST::RELATION('б', 1), FST::RELATION('в', 1), FST::RELATION('г', 1), FST::RELATION('д', 1), FST::RELATION('е', 1),\  FST::RELATION('ё', 1), FST::RELATION('ж', 1), FST::RELATION('з', 1), FST::RELATION('и', 1), FST::RELATION('й', 1), FST::RELATION('к', 1),\  FST::RELATION('л', 1), FST::RELATION('м', 1), FST::RELATION('н', 1), FST::RELATION('о', 1), FST::RELATION('п', 1), FST::RELATION('р', 1),\  FST::RELATION('с', 1), FST::RELATION('т', 1), FST::RELATION('у', 1), FST::RELATION('ф', 1), FST::RELATION('х', 1), FST::RELATION('ц', 1),\  FST::RELATION('ч', 1), FST::RELATION('ш', 1), FST::RELATION('щ', 1), FST::RELATION('ъ', 1), FST::RELATION('ы', 1), FST::RELATION('ь', 1),\  FST::RELATION('э', 1), FST::RELATION('ю', 1), FST::RELATION('я', 1), FST::RELATION(' ', 1), FST::RELATION('.', 1), FST::RELATION(',', 1),\  FST::RELATION('?', 1), FST::RELATION('!', 1), FST::RELATION(';', 1), FST::RELATION(':', 1), FST::RELATION('-', 1), FST::RELATION(')', 1),\  FST::RELATION('(', 1), \  \  FST::RELATION('a', 2), FST::RELATION('b', 2), FST::RELATION('c', 2), FST::RELATION('d', 2), FST::RELATION('e', 2), FST::RELATION('f', 2),\  FST::RELATION('g', 2), FST::RELATION('h', 2), FST::RELATION('i', 2), FST::RELATION('j', 2), FST::RELATION('k', 2), FST::RELATION('l', 2),\  FST::RELATION('m', 2), FST::RELATION('n', 2), FST::RELATION('o', 2), FST::RELATION('p', 2), FST::RELATION('q', 2), FST::RELATION('r', 2),\  FST::RELATION('s', 2), FST::RELATION('t', 2), FST::RELATION('u', 2), FST::RELATION('v', 2), FST::RELATION('w', 2), FST::RELATION('x', 2),\  FST::RELATION('y', 2), FST::RELATION('z', 2), FST::RELATION('1', 2), FST::RELATION('2', 2), FST::RELATION('3', 2), FST::RELATION('4', 2),\  FST::RELATION('5', 2), FST::RELATION('6', 2), FST::RELATION('7', 2), FST::RELATION('8', 2), FST::RELATION('9', 2), FST::RELATION('0', 2),\  \  FST::RELATION('а', 2), FST::RELATION('б', 2), FST::RELATION('в', 2), FST::RELATION('г', 2), FST::RELATION('д', 2), FST::RELATION('е', 2),\  FST::RELATION('ё', 2), FST::RELATION('ж', 2), FST::RELATION('з', 2), FST::RELATION('и', 2), FST::RELATION('й', 2), FST::RELATION('к', 2),\  FST::RELATION('л', 2), FST::RELATION('м', 2), FST::RELATION('н', 2), FST::RELATION('о', 2), FST::RELATION('п', 2), FST::RELATION('р', 2),\  FST::RELATION('с', 2), FST::RELATION('т', 2), FST::RELATION('у', 2), FST::RELATION('ф', 2), FST::RELATION('х', 2), FST::RELATION('ц', 2),\  FST::RELATION('ч', 2), FST::RELATION('ш', 2), FST::RELATION('щ', 2), FST::RELATION('ъ', 2), FST::RELATION('ы', 2), FST::RELATION('ь', 2),\  FST::RELATION('э', 2), FST::RELATION('ю', 2), FST::RELATION('я', 2), FST::RELATION(' ', 2), FST::RELATION('.', 2), FST::RELATION(',', 2),\  FST::RELATION('?', 2), FST::RELATION('!', 2), FST::RELATION(';', 2), FST::RELATION(':', 2), FST::RELATION('-', 2), FST::RELATION(')', 2),\  FST::RELATION('(', 2)),\  \  FST::NODE(1, FST::RELATION('\'', 3)),\  FST::NODE() |
| <boolliteral> | #define FST\_BOOLLIT 6,\  FST::NODE(2, FST::RELATION('t',1), FST::RELATION('f',1)),\  FST::NODE(2, FST::RELATION('r',2), FST::RELATION('a',2)),\  FST::NODE(2, FST::RELATION('u',3), FST::RELATION('l',3)),\  FST::NODE(2, FST::RELATION('t',4), FST::RELATION('s',4)),\  FST::NODE(2, FST::RELATION('h',5), FST::RELATION('e',5)),\  FST::NODE() |
| . | #define FST\_POINT 2,\  FST::NODE(1, FST::RELATION('.',1)),\  FST::NODE() |
| , | #define FST\_COMMA 2,\  FST::NODE(1, FST::RELATION(',',1)),\  FST::NODE() |
| [ | #define FST\_LEFTBRACE 2,\  FST::NODE(1, FST::RELATION('[',1)),\  FST::NODE() |
| ] | #define FST\_RIGHTBRACE 2,\  FST::NODE(1, FST::RELATION(']',1)),\  FST::NODE() |
| ( | #define FST\_LEFTHESIS 2, \  FST::NODE(1, FST::RELATION('(', 1)),\  FST::NODE() |
| ) | #define FST\_RIGHTHESIS 2, \  FST::NODE(1, FST::RELATION(')', 1)),\  FST::NODE() |
| > | #define FST\_MORE 2, \  FST::NODE(1, FST::RELATION('>', 1)),\  FST::NODE() |
| < | #define FST\_LESS 2, \  FST::NODE(1, FST::RELATION('<', 1)),\  FST::NODE() |
| = | #define FST\_EQUALLY 2, \  FST::NODE(1, FST::RELATION('=', 1)),\  FST::NODE() |
| : | #define FST\_COMPARE 2, \  FST::NODE(1, FST::RELATION(':', 1)),\  FST::NODE() |
| + | #define FST\_PLUS 2, \  FST::NODE(1, FST::RELATION('+', 1)),\  FST::NODE() |
| - | #define FST\_MINUS 2, \  FST::NODE(1, FST::RELATION('-', 1)),\  FST::NODE() |
| \* | #define FST\_STAR 2, \  FST::NODE(1, FST::RELATION('\*', 1)),\  FST::NODE() |
| / | #define FST\_DIRSLASH 2, \  FST::NODE(1, FST::RELATION('/', 1)),\  FST::NODE() |

**Приложение B**

**Распечатка дерева разбора**

Шаг: Правило Входная лента Стек

0 : S->d[N] d[ti:l.ti:l.ti,i,i,i,i.ti S$

0 : SAVESTATE: 1

0 : d[ti:l.ti:l.ti,i,i,i,i.ti d[N]$

1 : [ti:l.ti:l.ti,i,i,i,i.ti: [N]$

2 : ti:l.ti:l.ti,i,i,i,i.ti:l N]$

3 : N->ti:E.N ti:l.ti:l.ti,i,i,i,i.ti:l N]$

3 : SAVESTATE: 2

3 : ti:l.ti:l.ti,i,i,i,i.ti:l ti:E.N]$

4 : i:l.ti:l.ti,i,i,i,i.ti:l. i:E.N]$

5 : :l.ti:l.ti,i,i,i,i.ti:l.t :E.N]$

6 : l.ti:l.ti,i,i,i,i.ti:l.ti E.N]$

7 : E->l l.ti:l.ti,i,i,i,i.ti:l.ti E.N]$

7 : SAVESTATE: 3

7 : l.ti:l.ti,i,i,i,i.ti:l.ti l.N]$

8 : .ti:l.ti,i,i,i,i.ti:l.ti: .N]$

9 : ti:l.ti,i,i,i,i.ti:l.ti:l N]$

10 : N->ti:E.N ti:l.ti,i,i,i,i.ti:l.ti:l N]$

10 : SAVESTATE: 4

10 : ti:l.ti,i,i,i,i.ti:l.ti:l ti:E.N]$

11 : i:l.ti,i,i,i,i.ti:l.ti:l. i:E.N]$

12 : :l.ti,i,i,i,i.ti:l.ti:l.t :E.N]$

13 : l.ti,i,i,i,i.ti:l.ti:l.ti E.N]$

14 : E->l l.ti,i,i,i,i.ti:l.ti:l.ti E.N]$

14 : SAVESTATE: 5

14 : l.ti,i,i,i,i.ti:l.ti:l.ti l.N]$

15 : .ti,i,i,i,i.ti:l.ti:l.ti. .N]$

16 : ti,i,i,i,i.ti:l.ti:l.ti.i N]$

17 : N->ti:E.N ti,i,i,i,i.ti:l.ti:l.ti.i N]$

17 : SAVESTATE: 6

17 : ti,i,i,i,i.ti:l.ti:l.ti.i ti:E.N]$

18 : i,i,i,i,i.ti:l.ti:l.ti.i: i:E.N]$

19 : ,i,i,i,i.ti:l.ti:l.ti.i:l :E.N]$

20 : TS\_NOK/NS\_NORULECHAIN

20 : RESSTATE

20 : ti,i,i,i,i.ti:l.ti:l.ti.i N]$

21 : N->ti.N ti,i,i,i,i.ti:l.ti:l.ti.i N]$

21 : SAVESTATE: 6

21 : ti,i,i,i,i.ti:l.ti:l.ti.i ti.N]$

22 : i,i,i,i,i.ti:l.ti:l.ti.i: i.N]$

23 : ,i,i,i,i.ti:l.ti:l.ti.i:l .N]$

24 : TS\_NOK/NS\_NORULECHAIN

24 : RESSTATE

24 : ti,i,i,i,i.ti:l.ti:l.ti.i N]$

25 : N->tE.N ti,i,i,i,i.ti:l.ti:l.ti.i N]$

25 : SAVESTATE: 6

25 : ti,i,i,i,i.ti:l.ti:l.ti.i tE.N]$

26 : i,i,i,i,i.ti:l.ti:l.ti.i: E.N]$

27 : E->i i,i,i,i,i.ti:l.ti:l.ti.i: E.N]$

27 : SAVESTATE: 7

27 : i,i,i,i,i.ti:l.ti:l.ti.i: i.N]$

28 : ,i,i,i,i.ti:l.ti:l.ti.i:l .N]$

29 : TS\_NOK/NS\_NORULECHAIN

29 : RESSTATE

29 : i,i,i,i,i.ti:l.ti:l.ti.i: E.N]$

30 : E->i,E i,i,i,i,i.ti:l.ti:l.ti.i: E.N]$

30 : SAVESTATE: 7

30 : i,i,i,i,i.ti:l.ti:l.ti.i: i,E.N]$

31 : ,i,i,i,i.ti:l.ti:l.ti.i:l ,E.N]$

32 : i,i,i,i.ti:l.ti:l.ti.i:l. E.N]$

33 : E->i i,i,i,i.ti:l.ti:l.ti.i:l. E.N]$

33 : SAVESTATE: 8

33 : i,i,i,i.ti:l.ti:l.ti.i:l. i.N]$

34 : ,i,i,i.ti:l.ti:l.ti.i:l.w .N]$

35 : TS\_NOK/NS\_NORULECHAIN

35 : RESSTATE

35 : i,i,i,i.ti:l.ti:l.ti.i:l. E.N]$

36 : E->i,E i,i,i,i.ti:l.ti:l.ti.i:l. E.N]$

36 : SAVESTATE: 8

36 : i,i,i,i.ti:l.ti:l.ti.i:l. i,E.N]$

37 : ,i,i,i.ti:l.ti:l.ti.i:l.w ,E.N]$

38 : i,i,i.ti:l.ti:l.ti.i:l.w( E.N]$

39 : E->i i,i,i.ti:l.ti:l.ti.i:l.w( E.N]$

39 : SAVESTATE: 9

39 : i,i,i.ti:l.ti:l.ti.i:l.w( i.N]$

40 : ,i,i.ti:l.ti:l.ti.i:l.w(i .N]$

41 : TS\_NOK/NS\_NORULECHAIN

41 : RESSTATE

41 : i,i,i.ti:l.ti:l.ti.i:l.w( E.N]$

42 : E->i,E i,i,i.ti:l.ti:l.ti.i:l.w( E.N]$

42 : SAVESTATE: 9

42 : i,i,i.ti:l.ti:l.ti.i:l.w( i,E.N]$

43 : ,i,i.ti:l.ti:l.ti.i:l.w(i ,E.N]$

44 : i,i.ti:l.ti:l.ti.i:l.w(i> E.N]$

45 : E->i i,i.ti:l.ti:l.ti.i:l.w(i> E.N]$

45 : SAVESTATE: 10

45 : i,i.ti:l.ti:l.ti.i:l.w(i> i.N]$

46 : ,i.ti:l.ti:l.ti.i:l.w(i>i .N]$

47 : TS\_NOK/NS\_NORULECHAIN

47 : RESSTATE

47 : i,i.ti:l.ti:l.ti.i:l.w(i> E.N]$

48 : E->i,E i,i.ti:l.ti:l.ti.i:l.w(i> E.N]$

48 : SAVESTATE: 10

48 : i,i.ti:l.ti:l.ti.i:l.w(i> i,E.N]$

49 : ,i.ti:l.ti:l.ti.i:l.w(i>i ,E.N]$

50 : i.ti:l.ti:l.ti.i:l.w(i>i) E.N]$

51 : E->i i.ti:l.ti:l.ti.i:l.w(i>i) E.N]$

51 : SAVESTATE: 11

51 : i.ti:l.ti:l.ti.i:l.w(i>i) i.N]$

52 : .ti:l.ti:l.ti.i:l.w(i>i)[ .N]$

53 : ti:l.ti:l.ti.i:l.w(i>i)[t N]$

54 : N->ti:E.N ti:l.ti:l.ti.i:l.w(i>i)[t N]$

54 : SAVESTATE: 12

54 : ti:l.ti:l.ti.i:l.w(i>i)[t ti:E.N]$

55 : i:l.ti:l.ti.i:l.w(i>i)[ti i:E.N]$

56 : :l.ti:l.ti.i:l.w(i>i)[ti. :E.N]$

57 : l.ti:l.ti.i:l.w(i>i)[ti.i E.N]$

58 : E->l l.ti:l.ti.i:l.w(i>i)[ti.i E.N]$

58 : SAVESTATE: 13

58 : l.ti:l.ti.i:l.w(i>i)[ti.i l.N]$

59 : .ti:l.ti.i:l.w(i>i)[ti.i: .N]$

60 : ti:l.ti.i:l.w(i>i)[ti.i:i N]$

61 : N->ti:E.N ti:l.ti.i:l.w(i>i)[ti.i:i N]$

61 : SAVESTATE: 14

61 : ti:l.ti.i:l.w(i>i)[ti.i:i ti:E.N]$

62 : i:l.ti.i:l.w(i>i)[ti.i:i+ i:E.N]$

63 : :l.ti.i:l.w(i>i)[ti.i:i+i :E.N]$

64 : l.ti.i:l.w(i>i)[ti.i:i+i. E.N]$

65 : E->l l.ti.i:l.w(i>i)[ti.i:i+i. E.N]$

65 : SAVESTATE: 15

65 : l.ti.i:l.w(i>i)[ti.i:i+i. l.N]$

66 : .ti.i:l.w(i>i)[ti.i:i+i.p .N]$

67 : ti.i:l.w(i>i)[ti.i:i+i.p( N]$

68 : N->ti:E.N ti.i:l.w(i>i)[ti.i:i+i.p( N]$

68 : SAVESTATE: 16

68 : ti.i:l.w(i>i)[ti.i:i+i.p( ti:E.N]$

69 : i.i:l.w(i>i)[ti.i:i+i.p(i i:E.N]$

70 : .i:l.w(i>i)[ti.i:i+i.p(i) :E.N]$

71 : TS\_NOK/NS\_NORULECHAIN

71 : RESSTATE

71 : ti.i:l.w(i>i)[ti.i:i+i.p( N]$

72 : N->ti.N ti.i:l.w(i>i)[ti.i:i+i.p( N]$

72 : SAVESTATE: 16

72 : ti.i:l.w(i>i)[ti.i:i+i.p( ti.N]$

73 : i.i:l.w(i>i)[ti.i:i+i.p(i i.N]$

74 : .i:l.w(i>i)[ti.i:i+i.p(i) .N]$

75 : i:l.w(i>i)[ti.i:i+i.p(i). N]$

76 : N->i:P.N i:l.w(i>i)[ti.i:i+i.p(i). N]$

76 : SAVESTATE: 17

76 : i:l.w(i>i)[ti.i:i+i.p(i). i:P.N]$

77 : :l.w(i>i)[ti.i:i+i.p(i).] :P.N]$

78 : l.w(i>i)[ti.i:i+i.p(i).]e P.N]$

79 : P->l l.w(i>i)[ti.i:i+i.p(i).]e P.N]$

79 : SAVESTATE: 18

79 : l.w(i>i)[ti.i:i+i.p(i).]e l.N]$

80 : .w(i>i)[ti.i:i+i.p(i).]e[ .N]$

81 : TS\_NOK/NS\_NORULECHAIN

81 : RESSTATE

81 : l.w(i>i)[ti.i:i+i.p(i).]e P.N]$

82 : P->lMP l.w(i>i)[ti.i:i+i.p(i).]e P.N]$

82 : SAVESTATE: 18

82 : l.w(i>i)[ti.i:i+i.p(i).]e lMP.N]$

83 : .w(i>i)[ti.i:i+i.p(i).]e[ MP.N]$

84 : TNS\_NORULECHAIN/NS\_NORULE

84 : RESSTATE

84 : l.w(i>i)[ti.i:i+i.p(i).]e P.N]$

85 : TNS\_NORULECHAIN/NS\_NORULE

85 : RESSTATE

85 : i:l.w(i>i)[ti.i:i+i.p(i). N]$

86 : N->i:E.N i:l.w(i>i)[ti.i:i+i.p(i). N]$

86 : SAVESTATE: 17

86 : i:l.w(i>i)[ti.i:i+i.p(i). i:E.N]$

87 : :l.w(i>i)[ti.i:i+i.p(i).] :E.N]$

88 : l.w(i>i)[ti.i:i+i.p(i).]e E.N]$

89 : E->l l.w(i>i)[ti.i:i+i.p(i).]e E.N]$

89 : SAVESTATE: 18

89 : l.w(i>i)[ti.i:i+i.p(i).]e l.N]$

90 : .w(i>i)[ti.i:i+i.p(i).]e[ .N]$

91 : w(i>i)[ti.i:i+i.p(i).]e[i N]$

92 : N->w(K)[C]e[C]N w(i>i)[ti.i:i+i.p(i).]e[i N]$

92 : SAVESTATE: 19

92 : w(i>i)[ti.i:i+i.p(i).]e[i w(K)[C]e[C]N]$

93 : (i>i)[ti.i:i+i.p(i).]e[i: (K)[C]e[C]N]$

94 : i>i)[ti.i:i+i.p(i).]e[i:f K)[C]e[C]N]$

95 : K->iBi i>i)[ti.i:i+i.p(i).]e[i:f K)[C]e[C]N]$

95 : SAVESTATE: 20

95 : i>i)[ti.i:i+i.p(i).]e[i:f iBi)[C]e[C]N]$

96 : >i)[ti.i:i+i.p(i).]e[i:f( Bi)[C]e[C]N]$

97 : B->> >i)[ti.i:i+i.p(i).]e[i:f( Bi)[C]e[C]N]$

97 : SAVESTATE: 21

97 : >i)[ti.i:i+i.p(i).]e[i:f( >i)[C]e[C]N]$

98 : i)[ti.i:i+i.p(i).]e[i:f(i i)[C]e[C]N]$

99 : )[ti.i:i+i.p(i).]e[i:f(i) )[C]e[C]N]$

100 : [ti.i:i+i.p(i).]e[i:f(i). [C]e[C]N]$

101 : ti.i:i+i.p(i).]e[i:f(i).] C]e[C]N]$

102 : C->ti:E.C ti.i:i+i.p(i).]e[i:f(i).] C]e[C]N]$

102 : SAVESTATE: 22

102 : ti.i:i+i.p(i).]e[i:f(i).] ti:E.C]e[C]N]$

103 : i.i:i+i.p(i).]e[i:f(i).]p i:E.C]e[C]N]$

104 : .i:i+i.p(i).]e[i:f(i).]p( :E.C]e[C]N]$

105 : TS\_NOK/NS\_NORULECHAIN

105 : RESSTATE

105 : ti.i:i+i.p(i).]e[i:f(i).] C]e[C]N]$

106 : C->ti:E. ti.i:i+i.p(i).]e[i:f(i).] C]e[C]N]$

106 : SAVESTATE: 22

106 : ti.i:i+i.p(i).]e[i:f(i).] ti:E.]e[C]N]$

107 : i.i:i+i.p(i).]e[i:f(i).]p i:E.]e[C]N]$

108 : .i:i+i.p(i).]e[i:f(i).]p( :E.]e[C]N]$

109 : TS\_NOK/NS\_NORULECHAIN

109 : RESSTATE

109 : ti.i:i+i.p(i).]e[i:f(i).] C]e[C]N]$

110 : C->ti:P.C ti.i:i+i.p(i).]e[i:f(i).] C]e[C]N]$

110 : SAVESTATE: 22

110 : ti.i:i+i.p(i).]e[i:f(i).] ti:P.C]e[C]N]$

111 : i.i:i+i.p(i).]e[i:f(i).]p i:P.C]e[C]N]$

112 : .i:i+i.p(i).]e[i:f(i).]p( :P.C]e[C]N]$

113 : TS\_NOK/NS\_NORULECHAIN

113 : RESSTATE

113 : ti.i:i+i.p(i).]e[i:f(i).] C]e[C]N]$

114 : C->ti:P. ti.i:i+i.p(i).]e[i:f(i).] C]e[C]N]$

114 : SAVESTATE: 22

114 : ti.i:i+i.p(i).]e[i:f(i).] ti:P.]e[C]N]$

115 : i.i:i+i.p(i).]e[i:f(i).]p i:P.]e[C]N]$

116 : .i:i+i.p(i).]e[i:f(i).]p( :P.]e[C]N]$

117 : TS\_NOK/NS\_NORULECHAIN

117 : RESSTATE

117 : ti.i:i+i.p(i).]e[i:f(i).] C]e[C]N]$

118 : C->ti.C ti.i:i+i.p(i).]e[i:f(i).] C]e[C]N]$

118 : SAVESTATE: 22

118 : ti.i:i+i.p(i).]e[i:f(i).] ti.C]e[C]N]$

119 : i.i:i+i.p(i).]e[i:f(i).]p i.C]e[C]N]$

120 : .i:i+i.p(i).]e[i:f(i).]p( .C]e[C]N]$

121 : i:i+i.p(i).]e[i:f(i).]p(i C]e[C]N]$

122 : C->i:E. i:i+i.p(i).]e[i:f(i).]p(i C]e[C]N]$

122 : SAVESTATE: 23

122 : i:i+i.p(i).]e[i:f(i).]p(i i:E.]e[C]N]$

123 : :i+i.p(i).]e[i:f(i).]p(i) :E.]e[C]N]$

124 : i+i.p(i).]e[i:f(i).]p(i). E.]e[C]N]$

125 : E->i i+i.p(i).]e[i:f(i).]p(i). E.]e[C]N]$

125 : SAVESTATE: 24

125 : i+i.p(i).]e[i:f(i).]p(i). i.]e[C]N]$

126 : +i.p(i).]e[i:f(i).]p(i).i .]e[C]N]$

127 : TS\_NOK/NS\_NORULECHAIN

127 : RESSTATE

127 : i+i.p(i).]e[i:f(i).]p(i). E.]e[C]N]$

128 : E->i,E i+i.p(i).]e[i:f(i).]p(i). E.]e[C]N]$

128 : SAVESTATE: 24

128 : i+i.p(i).]e[i:f(i).]p(i). i,E.]e[C]N]$

129 : +i.p(i).]e[i:f(i).]p(i).i ,E.]e[C]N]$

130 : TS\_NOK/NS\_NORULECHAIN

130 : RESSTATE

130 : i+i.p(i).]e[i:f(i).]p(i). E.]e[C]N]$

131 : TNS\_NORULECHAIN/NS\_NORULE

131 : RESSTATE

131 : i:i+i.p(i).]e[i:f(i).]p(i C]e[C]N]$

132 : C->i:E.C i:i+i.p(i).]e[i:f(i).]p(i C]e[C]N]$

132 : SAVESTATE: 23

132 : i:i+i.p(i).]e[i:f(i).]p(i i:E.C]e[C]N]$

133 : :i+i.p(i).]e[i:f(i).]p(i) :E.C]e[C]N]$

134 : i+i.p(i).]e[i:f(i).]p(i). E.C]e[C]N]$

135 : E->i i+i.p(i).]e[i:f(i).]p(i). E.C]e[C]N]$

135 : SAVESTATE: 24

135 : i+i.p(i).]e[i:f(i).]p(i). i.C]e[C]N]$

136 : +i.p(i).]e[i:f(i).]p(i).i .C]e[C]N]$

137 : TS\_NOK/NS\_NORULECHAIN

137 : RESSTATE

137 : i+i.p(i).]e[i:f(i).]p(i). E.C]e[C]N]$

138 : E->i,E i+i.p(i).]e[i:f(i).]p(i). E.C]e[C]N]$

138 : SAVESTATE: 24

138 : i+i.p(i).]e[i:f(i).]p(i). i,E.C]e[C]N]$

139 : +i.p(i).]e[i:f(i).]p(i).i ,E.C]e[C]N]$

140 : TS\_NOK/NS\_NORULECHAIN

140 : RESSTATE

140 : i+i.p(i).]e[i:f(i).]p(i). E.C]e[C]N]$

141 : TNS\_NORULECHAIN/NS\_NORULE

141 : RESSTATE

141 : i:i+i.p(i).]e[i:f(i).]p(i C]e[C]N]$

142 : C->i:P. i:i+i.p(i).]e[i:f(i).]p(i C]e[C]N]$

142 : SAVESTATE: 23

142 : i:i+i.p(i).]e[i:f(i).]p(i i:P.]e[C]N]$

143 : :i+i.p(i).]e[i:f(i).]p(i) :P.]e[C]N]$

144 : i+i.p(i).]e[i:f(i).]p(i). P.]e[C]N]$

145 : P->i i+i.p(i).]e[i:f(i).]p(i). P.]e[C]N]$

145 : SAVESTATE: 24

145 : i+i.p(i).]e[i:f(i).]p(i). i.]e[C]N]$

146 : +i.p(i).]e[i:f(i).]p(i).i .]e[C]N]$

147 : TS\_NOK/NS\_NORULECHAIN

147 : RESSTATE

147 : i+i.p(i).]e[i:f(i).]p(i). P.]e[C]N]$

148 : P->iMP i+i.p(i).]e[i:f(i).]p(i). P.]e[C]N]$

148 : SAVESTATE: 24

148 : i+i.p(i).]e[i:f(i).]p(i). iMP.]e[C]N]$

149 : +i.p(i).]e[i:f(i).]p(i).i MP.]e[C]N]$

150 : M->+ +i.p(i).]e[i:f(i).]p(i).i MP.]e[C]N]$

150 : SAVESTATE: 25

150 : +i.p(i).]e[i:f(i).]p(i).i +P.]e[C]N]$

151 : i.p(i).]e[i:f(i).]p(i).i: P.]e[C]N]$

152 : P->i i.p(i).]e[i:f(i).]p(i).i: P.]e[C]N]$

152 : SAVESTATE: 26

152 : i.p(i).]e[i:f(i).]p(i).i: i.]e[C]N]$

153 : .p(i).]e[i:f(i).]p(i).i:( .]e[C]N]$

154 : p(i).]e[i:f(i).]p(i).i:(i ]e[C]N]$

155 : TS\_NOK/NS\_NORULECHAIN

155 : RESSTATE

155 : i.p(i).]e[i:f(i).]p(i).i: P.]e[C]N]$

156 : P->iMP i.p(i).]e[i:f(i).]p(i).i: P.]e[C]N]$

156 : SAVESTATE: 26

156 : i.p(i).]e[i:f(i).]p(i).i: iMP.]e[C]N]$

157 : .p(i).]e[i:f(i).]p(i).i:( MP.]e[C]N]$

158 : TNS\_NORULECHAIN/NS\_NORULE

158 : RESSTATE

158 : i.p(i).]e[i:f(i).]p(i).i: P.]e[C]N]$

159 : TNS\_NORULECHAIN/NS\_NORULE

159 : RESSTATE

159 : +i.p(i).]e[i:f(i).]p(i).i MP.]e[C]N]$

160 : TNS\_NORULECHAIN/NS\_NORULE

160 : RESSTATE

160 : i+i.p(i).]e[i:f(i).]p(i). P.]e[C]N]$

161 : TNS\_NORULECHAIN/NS\_NORULE

161 : RESSTATE

161 : i:i+i.p(i).]e[i:f(i).]p(i C]e[C]N]$

162 : C->i:P.C i:i+i.p(i).]e[i:f(i).]p(i C]e[C]N]$

162 : SAVESTATE: 23

162 : i:i+i.p(i).]e[i:f(i).]p(i i:P.C]e[C]N]$

163 : :i+i.p(i).]e[i:f(i).]p(i) :P.C]e[C]N]$

164 : i+i.p(i).]e[i:f(i).]p(i). P.C]e[C]N]$

165 : P->i i+i.p(i).]e[i:f(i).]p(i). P.C]e[C]N]$

165 : SAVESTATE: 24

165 : i+i.p(i).]e[i:f(i).]p(i). i.C]e[C]N]$

166 : +i.p(i).]e[i:f(i).]p(i).i .C]e[C]N]$

167 : TS\_NOK/NS\_NORULECHAIN

167 : RESSTATE

167 : i+i.p(i).]e[i:f(i).]p(i). P.C]e[C]N]$

168 : P->iMP i+i.p(i).]e[i:f(i).]p(i). P.C]e[C]N]$

168 : SAVESTATE: 24

168 : i+i.p(i).]e[i:f(i).]p(i). iMP.C]e[C]N]$

169 : +i.p(i).]e[i:f(i).]p(i).i MP.C]e[C]N]$

170 : M->+ +i.p(i).]e[i:f(i).]p(i).i MP.C]e[C]N]$

170 : SAVESTATE: 25

170 : +i.p(i).]e[i:f(i).]p(i).i +P.C]e[C]N]$

171 : i.p(i).]e[i:f(i).]p(i).i: P.C]e[C]N]$

172 : P->i i.p(i).]e[i:f(i).]p(i).i: P.C]e[C]N]$

172 : SAVESTATE: 26

172 : i.p(i).]e[i:f(i).]p(i).i: i.C]e[C]N]$

173 : .p(i).]e[i:f(i).]p(i).i:( .C]e[C]N]$

174 : p(i).]e[i:f(i).]p(i).i:(i C]e[C]N]$

175 : C->p(E).C p(i).]e[i:f(i).]p(i).i:(i C]e[C]N]$

175 : SAVESTATE: 27

175 : p(i).]e[i:f(i).]p(i).i:(i p(E).C]e[C]N]$

176 : (i).]e[i:f(i).]p(i).i:(i+ (E).C]e[C]N]$

177 : i).]e[i:f(i).]p(i).i:(i+i E).C]e[C]N]$

178 : E->i i).]e[i:f(i).]p(i).i:(i+i E).C]e[C]N]$

178 : SAVESTATE: 28

178 : i).]e[i:f(i).]p(i).i:(i+i i).C]e[C]N]$

179 : ).]e[i:f(i).]p(i).i:(i+i) ).C]e[C]N]$

180 : .]e[i:f(i).]p(i).i:(i+i)\* .C]e[C]N]$

181 : ]e[i:f(i).]p(i).i:(i+i)\*( C]e[C]N]$

182 : TNS\_NORULECHAIN/NS\_NORULE

182 : RESSTATE

182 : i).]e[i:f(i).]p(i).i:(i+i E).C]e[C]N]$

183 : E->i,E i).]e[i:f(i).]p(i).i:(i+i E).C]e[C]N]$

183 : SAVESTATE: 28

183 : i).]e[i:f(i).]p(i).i:(i+i i,E).C]e[C]N]$

184 : ).]e[i:f(i).]p(i).i:(i+i) ,E).C]e[C]N]$

185 : TS\_NOK/NS\_NORULECHAIN

185 : RESSTATE

185 : i).]e[i:f(i).]p(i).i:(i+i E).C]e[C]N]$

186 : TNS\_NORULECHAIN/NS\_NORULE

186 : RESSTATE

186 : p(i).]e[i:f(i).]p(i).i:(i C]e[C]N]$

187 : C->p(E). p(i).]e[i:f(i).]p(i).i:(i C]e[C]N]$

187 : SAVESTATE: 27

187 : p(i).]e[i:f(i).]p(i).i:(i p(E).]e[C]N]$

188 : (i).]e[i:f(i).]p(i).i:(i+ (E).]e[C]N]$

189 : i).]e[i:f(i).]p(i).i:(i+i E).]e[C]N]$

190 : E->i i).]e[i:f(i).]p(i).i:(i+i E).]e[C]N]$

190 : SAVESTATE: 28

190 : i).]e[i:f(i).]p(i).i:(i+i i).]e[C]N]$

191 : ).]e[i:f(i).]p(i).i:(i+i) ).]e[C]N]$

192 : .]e[i:f(i).]p(i).i:(i+i)\* .]e[C]N]$

193 : ]e[i:f(i).]p(i).i:(i+i)\*( ]e[C]N]$

194 : e[i:f(i).]p(i).i:(i+i)\*(i e[C]N]$

195 : [i:f(i).]p(i).i:(i+i)\*(i- [C]N]$

196 : i:f(i).]p(i).i:(i+i)\*(i-i C]N]$

197 : C->i:E. i:f(i).]p(i).i:(i+i)\*(i-i C]N]$

197 : SAVESTATE: 29

197 : i:f(i).]p(i).i:(i+i)\*(i-i i:E.]N]$

198 : :f(i).]p(i).i:(i+i)\*(i-i) :E.]N]$

199 : f(i).]p(i).i:(i+i)\*(i-i). E.]N]$

200 : E->f(E) f(i).]p(i).i:(i+i)\*(i-i). E.]N]$

200 : SAVESTATE: 30

200 : f(i).]p(i).i:(i+i)\*(i-i). f(E).]N]$

201 : (i).]p(i).i:(i+i)\*(i-i).t (E).]N]$

202 : i).]p(i).i:(i+i)\*(i-i).ti E).]N]$

203 : E->i i).]p(i).i:(i+i)\*(i-i).ti E).]N]$

203 : SAVESTATE: 31

203 : i).]p(i).i:(i+i)\*(i-i).ti i).]N]$

204 : ).]p(i).i:(i+i)\*(i-i).ti: ).]N]$

205 : .]p(i).i:(i+i)\*(i-i).ti:f .]N]$

206 : ]p(i).i:(i+i)\*(i-i).ti:f( ]N]$

207 : p(i).i:(i+i)\*(i-i).ti:f(i N]$

208 : N->p(E).N p(i).i:(i+i)\*(i-i).ti:f(i N]$

208 : SAVESTATE: 32

208 : p(i).i:(i+i)\*(i-i).ti:f(i p(E).N]$

209 : (i).i:(i+i)\*(i-i).ti:f(i, (E).N]$

210 : i).i:(i+i)\*(i-i).ti:f(i,i E).N]$

211 : E->i i).i:(i+i)\*(i-i).ti:f(i,i E).N]$

211 : SAVESTATE: 33

211 : i).i:(i+i)\*(i-i).ti:f(i,i i).N]$

212 : ).i:(i+i)\*(i-i).ti:f(i,i) ).N]$

213 : .i:(i+i)\*(i-i).ti:f(i,i). .N]$

214 : i:(i+i)\*(i-i).ti:f(i,i).p N]$

215 : N->i:P.N i:(i+i)\*(i-i).ti:f(i,i).p N]$

215 : SAVESTATE: 34

215 : i:(i+i)\*(i-i).ti:f(i,i).p i:P.N]$

216 : :(i+i)\*(i-i).ti:f(i,i).p( :P.N]$

217 : (i+i)\*(i-i).ti:f(i,i).p(i P.N]$

218 : P->(P) (i+i)\*(i-i).ti:f(i,i).p(i P.N]$

218 : SAVESTATE: 35

218 : (i+i)\*(i-i).ti:f(i,i).p(i (P).N]$

219 : i+i)\*(i-i).ti:f(i,i).p(i) P).N]$

220 : P->i i+i)\*(i-i).ti:f(i,i).p(i) P).N]$

220 : SAVESTATE: 36

220 : i+i)\*(i-i).ti:f(i,i).p(i) i).N]$

221 : +i)\*(i-i).ti:f(i,i).p(i). ).N]$

222 : TS\_NOK/NS\_NORULECHAIN

222 : RESSTATE

222 : i+i)\*(i-i).ti:f(i,i).p(i) P).N]$

223 : P->iMP i+i)\*(i-i).ti:f(i,i).p(i) P).N]$

223 : SAVESTATE: 36

223 : i+i)\*(i-i).ti:f(i,i).p(i) iMP).N]$

224 : +i)\*(i-i).ti:f(i,i).p(i). MP).N]$

225 : M->+ +i)\*(i-i).ti:f(i,i).p(i). MP).N]$

225 : SAVESTATE: 37

225 : +i)\*(i-i).ti:f(i,i).p(i). +P).N]$

226 : i)\*(i-i).ti:f(i,i).p(i).r P).N]$

227 : P->i i)\*(i-i).ti:f(i,i).p(i).r P).N]$

227 : SAVESTATE: 38

227 : i)\*(i-i).ti:f(i,i).p(i).r i).N]$

228 : )\*(i-i).ti:f(i,i).p(i).ri ).N]$

229 : \*(i-i).ti:f(i,i).p(i).ri. .N]$

230 : TS\_NOK/NS\_NORULECHAIN

230 : RESSTATE

230 : i)\*(i-i).ti:f(i,i).p(i).r P).N]$

231 : P->iMP i)\*(i-i).ti:f(i,i).p(i).r P).N]$

231 : SAVESTATE: 38

231 : i)\*(i-i).ti:f(i,i).p(i).r iMP).N]$

232 : )\*(i-i).ti:f(i,i).p(i).ri MP).N]$

233 : TNS\_NORULECHAIN/NS\_NORULE

233 : RESSTATE

233 : i)\*(i-i).ti:f(i,i).p(i).r P).N]$

234 : TNS\_NORULECHAIN/NS\_NORULE

234 : RESSTATE

234 : +i)\*(i-i).ti:f(i,i).p(i). MP).N]$

235 : TNS\_NORULECHAIN/NS\_NORULE

235 : RESSTATE

235 : i+i)\*(i-i).ti:f(i,i).p(i) P).N]$

236 : TNS\_NORULECHAIN/NS\_NORULE

236 : RESSTATE

236 : (i+i)\*(i-i).ti:f(i,i).p(i P.N]$

237 : P->(P)MP (i+i)\*(i-i).ti:f(i,i).p(i P.N]$

237 : SAVESTATE: 35

237 : (i+i)\*(i-i).ti:f(i,i).p(i (P)MP.N]$

238 : i+i)\*(i-i).ti:f(i,i).p(i) P)MP.N]$

239 : P->i i+i)\*(i-i).ti:f(i,i).p(i) P)MP.N]$

239 : SAVESTATE: 36

239 : i+i)\*(i-i).ti:f(i,i).p(i) i)MP.N]$

240 : +i)\*(i-i).ti:f(i,i).p(i). )MP.N]$

241 : TS\_NOK/NS\_NORULECHAIN

241 : RESSTATE

241 : i+i)\*(i-i).ti:f(i,i).p(i) P)MP.N]$

242 : P->iMP i+i)\*(i-i).ti:f(i,i).p(i) P)MP.N]$

242 : SAVESTATE: 36

242 : i+i)\*(i-i).ti:f(i,i).p(i) iMP)MP.N]$

243 : +i)\*(i-i).ti:f(i,i).p(i). MP)MP.N]$

244 : M->+ +i)\*(i-i).ti:f(i,i).p(i). MP)MP.N]$

244 : SAVESTATE: 37

244 : +i)\*(i-i).ti:f(i,i).p(i). +P)MP.N]$

245 : i)\*(i-i).ti:f(i,i).p(i).r P)MP.N]$

246 : P->i i)\*(i-i).ti:f(i,i).p(i).r P)MP.N]$

246 : SAVESTATE: 38

246 : i)\*(i-i).ti:f(i,i).p(i).r i)MP.N]$

247 : )\*(i-i).ti:f(i,i).p(i).ri )MP.N]$

248 : \*(i-i).ti:f(i,i).p(i).ri. MP.N]$

249 : M->\* \*(i-i).ti:f(i,i).p(i).ri. MP.N]$

249 : SAVESTATE: 39

249 : \*(i-i).ti:f(i,i).p(i).ri. \*P.N]$

250 : (i-i).ti:f(i,i).p(i).ri.] P.N]$

251 : P->(P) (i-i).ti:f(i,i).p(i).ri.] P.N]$

251 : SAVESTATE: 40

251 : (i-i).ti:f(i,i).p(i).ri.] (P).N]$

252 : i-i).ti:f(i,i).p(i).ri.] P).N]$

253 : P->i i-i).ti:f(i,i).p(i).ri.] P).N]$

253 : SAVESTATE: 41

253 : i-i).ti:f(i,i).p(i).ri.] i).N]$

254 : -i).ti:f(i,i).p(i).ri.] ).N]$

255 : TS\_NOK/NS\_NORULECHAIN

255 : RESSTATE

255 : i-i).ti:f(i,i).p(i).ri.] P).N]$

256 : P->iMP i-i).ti:f(i,i).p(i).ri.] P).N]$

256 : SAVESTATE: 41

256 : i-i).ti:f(i,i).p(i).ri.] iMP).N]$

257 : -i).ti:f(i,i).p(i).ri.] MP).N]$

258 : M->- -i).ti:f(i,i).p(i).ri.] MP).N]$

258 : SAVESTATE: 42

258 : -i).ti:f(i,i).p(i).ri.] -P).N]$

259 : i).ti:f(i,i).p(i).ri.] P).N]$

260 : P->i i).ti:f(i,i).p(i).ri.] P).N]$

260 : SAVESTATE: 43

260 : i).ti:f(i,i).p(i).ri.] i).N]$

261 : ).ti:f(i,i).p(i).ri.] ).N]$

262 : .ti:f(i,i).p(i).ri.] .N]$

263 : ti:f(i,i).p(i).ri.] N]$

264 : N->ti:E.N ti:f(i,i).p(i).ri.] N]$

264 : SAVESTATE: 44

264 : ti:f(i,i).p(i).ri.] ti:E.N]$

265 : i:f(i,i).p(i).ri.] i:E.N]$

266 : :f(i,i).p(i).ri.] :E.N]$

267 : f(i,i).p(i).ri.] E.N]$

268 : E->f(E) f(i,i).p(i).ri.] E.N]$

268 : SAVESTATE: 45

268 : f(i,i).p(i).ri.] f(E).N]$

269 : (i,i).p(i).ri.] (E).N]$

270 : i,i).p(i).ri.] E).N]$

271 : E->i i,i).p(i).ri.] E).N]$

271 : SAVESTATE: 46

271 : i,i).p(i).ri.] i).N]$

272 : ,i).p(i).ri.] ).N]$

273 : TS\_NOK/NS\_NORULECHAIN

273 : RESSTATE

273 : i,i).p(i).ri.] E).N]$

274 : E->i,E i,i).p(i).ri.] E).N]$

274 : SAVESTATE: 46

274 : i,i).p(i).ri.] i,E).N]$

275 : ,i).p(i).ri.] ,E).N]$

276 : i).p(i).ri.] E).N]$

277 : E->i i).p(i).ri.] E).N]$

277 : SAVESTATE: 47

277 : i).p(i).ri.] i).N]$

278 : ).p(i).ri.] ).N]$

279 : .p(i).ri.] .N]$

280 : p(i).ri.] N]$

281 : N->p(E).N p(i).ri.] N]$

281 : SAVESTATE: 48

281 : p(i).ri.] p(E).N]$

282 : (i).ri.] (E).N]$

283 : i).ri.] E).N]$

284 : E->i i).ri.] E).N]$

284 : SAVESTATE: 49

284 : i).ri.] i).N]$

285 : ).ri.] ).N]$

286 : .ri.] .N]$

287 : ri.] N]$

288 : N->rE. ri.] N]$

288 : SAVESTATE: 50

288 : ri.] rE.]$

289 : i.] E.]$

290 : E->i i.] E.]$

290 : SAVESTATE: 51

290 : i.] i.]$

291 : .] .]$

292 : ] ]$

293 : $

294 : LENTA\_END

295 : -------> NS\_LENTA\_END