**Программный комплекс «Проверка строки на принадлежность множеству формул языка логики высказываний»**

**Формулировка задачи**

При выполнении данной работы была поставлена следующая цель: разработать программный комплекс, определяющий, является ли введенная строка формулой языка логики высказываний.

Для достижения этой цели были выделены следующие задачи:

* анализ подходов к решению: знакомство с языком разработки, составление спецификации агентов
* проектирование системы: графическое представление логики работы программы (диаграммы потоков данных, схемы алгоритмов работы агентов)
* программная реализация: реализация спроектированной системы на требуемом языке программирования
* тестирование системы: разработка комплекса тестов для корректировки работы программного комплекса

**Анализ подходов к решению**

Результатом анализа подходов к решению поставленной задачи стала спецификация агентов, требуемых для разработки программного комплекса. Среди агентов выделены типов агентов:

* тип I – замена одного символа на другой символ
* тип II – замена одного символа на комбинацию символов
* тип III – замена комбинации двух символов на один символ
* тип IV – замена комбинации трёх символов на один символ
* тип V – рекурсивная замена n символов на один символ
* тип VI – агенты сравнения
* тип VII – агенты управления

Данная спецификация включает:

1. Супервизор (тип VII)

**2. Замена 1-9 на 1** (тип I)

3. Рекурсивная замена 10 на 1 (тип V)

4. Рекурсивная замена 11 на 1 (тип V)

5. Замена .L1 на 9 (тип IV)

6. *Замена .L на A* (тип III)

**7. Замена A-Z на А** (тип I)

**8. Замена А1 на А** (тип III)

9. *Замена -> на ~* (тип III)

10*. Замена \/ и /\ на ~* (тип III)

**11. Замена ! на А ~** (тип II)

12. Установка в очередь (тип VII)

13. Разделение потоков данных (тип VII)

14. Рекурсивная замена (А ~ А) на А (тип V)

15. Посимвольное сравнение строк (тип VI)

16. Сравнение получившейся строки с А (тип VI)

**Проектирование системы**

Граф потока управления

**Агент 1. Супервизор**

Входные данные: {True, False}

Принцип работы: на данный момент не определён.

**Агент 2. Простая замена [1, 9] на 1**

Шаблон входных данных: X = [1, 9], в ASCII X = [49, 57]

Выходные данные: z = {1}, в ASCII z = {49}

Для сравнения входящего символа на принадлежность множеству была использована математическая формула:

x’ = x - (min(max(49, x), 57) - 49) \* 57 / max(57, x)

Принцип работы:

1. Поставить флажок “могу получить”
2. Принять на вход битовую строку
3. Поставить флажок “могу получить”
4. Проверка значения:

a. если значение символа из промежутка [1, 9]:

* 1. заменить значение на 1
  2. отправить символ на вход следующего агента

если значение символа не из промежутка [1, 9], то отправить символ на вход следующего агента

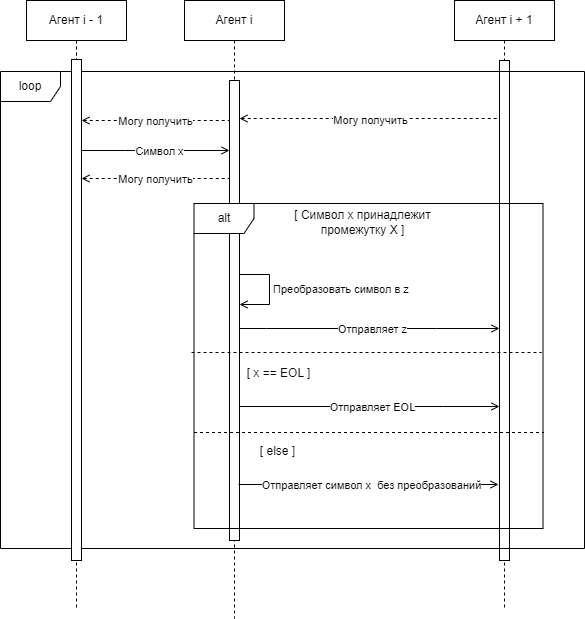


Схема работы агента 2

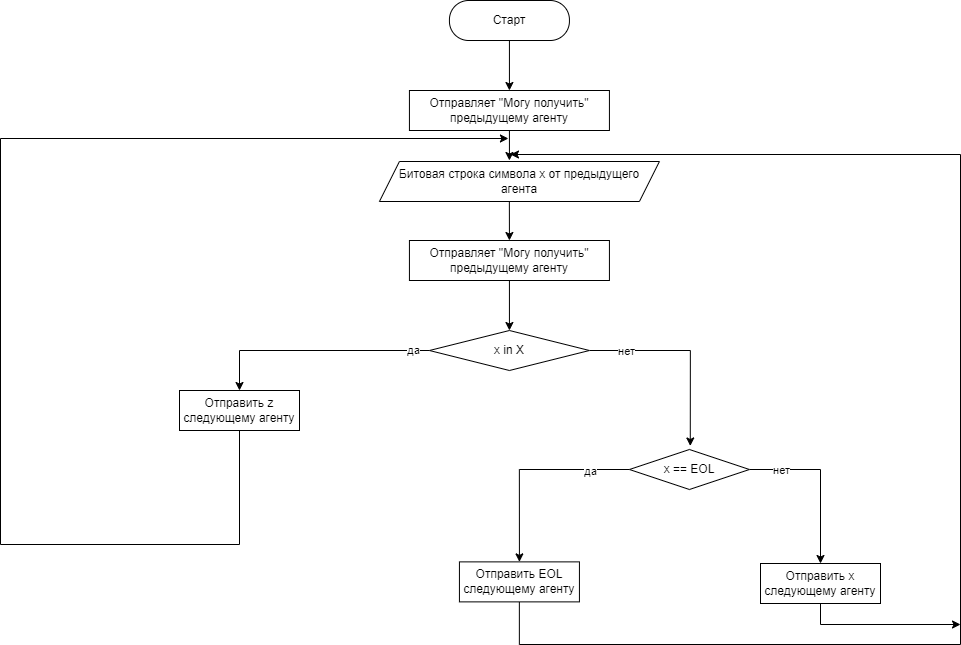


Диаграмма последовательности работы агента 2

**Агент 3. Рекурсивная замена 10 на 1**

Шаблон входных данных: X = {10}

Выходные данные: z = {1}

Принцип работы:

1. Поставить флажок “могу получить”
2. Принять на вход битовую строку
3. Поставить флажок “могу получить”
4. Проверка буфера:
   1. если буфер пустой, то сравниваем с 1:
      1. если 1, то добавляем в буфер, переходим на шаг 2;
      2. если EOL - то отправляем EOL на следующий агент, переходим на шаг 2;
      3. если не 1, то отправляем символ дальше, переходим на шаг 2.
   2. если буфер не пустой, то сравниваем с 0:
      1. если 0, то удаляем 0 из очереди, переходим на шаг 2.
      2. если EOL - то отправляем буфер и EOL на следующий агент, переходим на шаг 2.
      3. если не 0, то отправляем буфер и символ дальше, переходим на шаг 2.

**Агент 4. Рекурсивная замена 11 на 1**

Шаблон входных данных: X = {11}

Выходные данные: z ={1}

Принцип работы:

1. на вход агент получает символ, закодированный в виде битовой строки, и добавляет его в буфер;
2. если количество элементов в буфере равно 2:
   1. если содержимое буфера соответствует шаблону, то удалить одну 1 из буфера
   2. если содержимое буфера не соответствует шаблону, то первый элемент в буфере подается на вход агента №5
3. если количество элементов в буфере меньше 2
4. если в буфере EOL, то EOL подается на вход агента №5 и агент №4 заканчивает работу
5. если в буфере не EOL, повтор пунктов 1-3

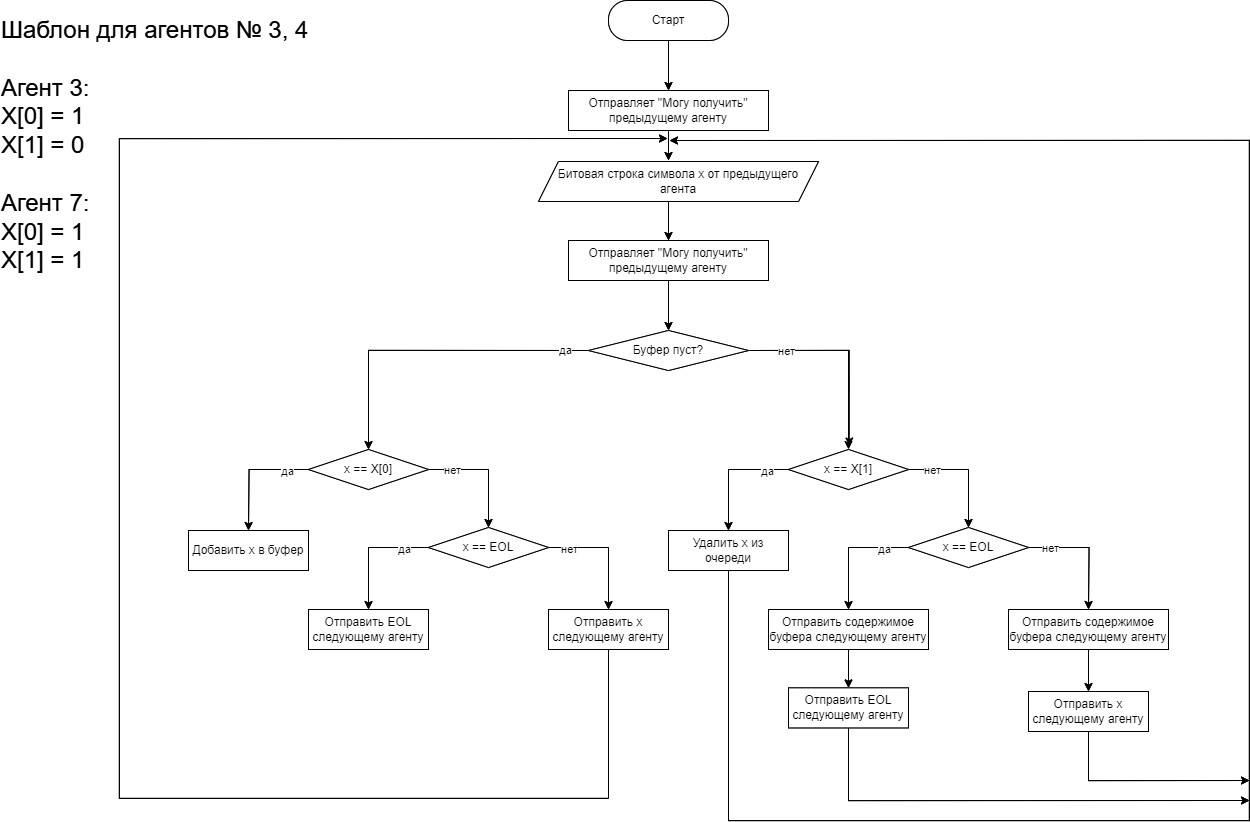


Схема работы агентов 3, 4

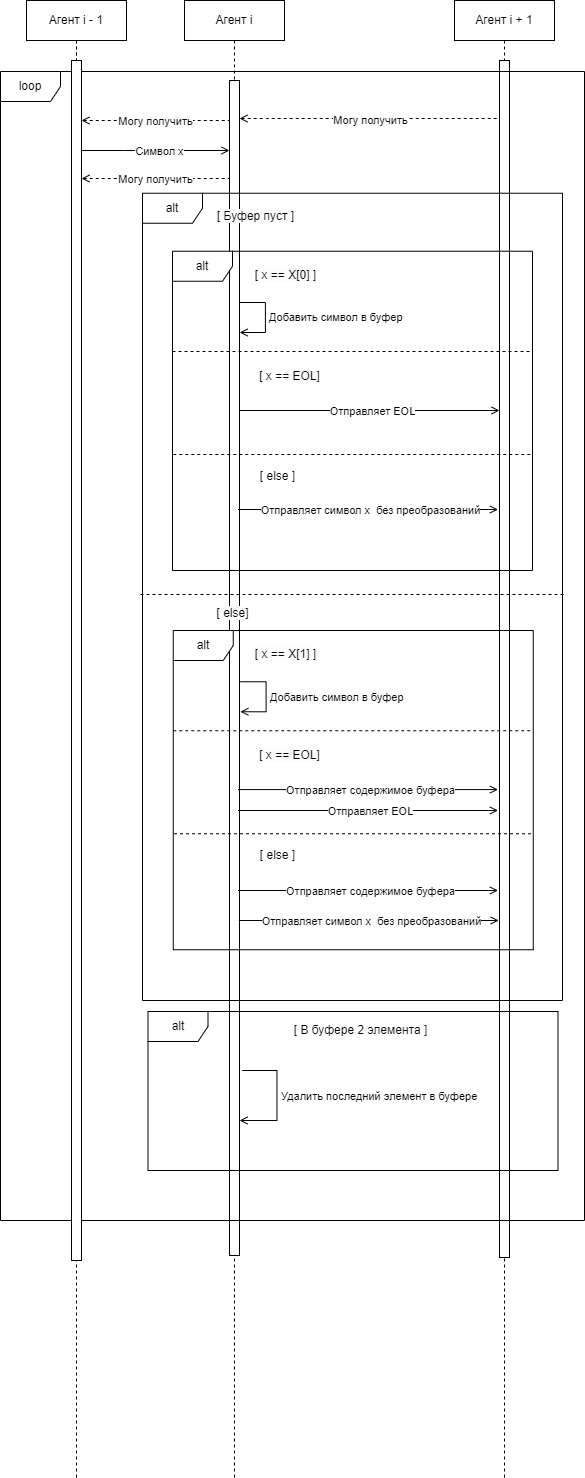


Диаграмма последовательности работы агентов 3, 4

**Агент 5. Контекстная замена .L1 на 9**

Шаблон входных данных: {.L1}

Выходные данные: {.91}

Принцип работы:

1. на вход агент получает символ, закодированный в виде битовой строки, и добавляет его в буфер;
2. если количество элементов в буфере равно 3:
   1. если содержимое буфера соответствует шаблону, то удалить все элементы буфера и добавить в буфер .91
   2. если содержимое буфера не соответствует шаблону, то первый элемент в буфере подается на вход агента №6
3. если количество элементов в буфере меньше 3:
   1. если в буфере только EOL, то EOL подается на вход агента №6 и агент №5 заканчивает работу
   2. если в буфере не EOL, повтор пунктов 1-3

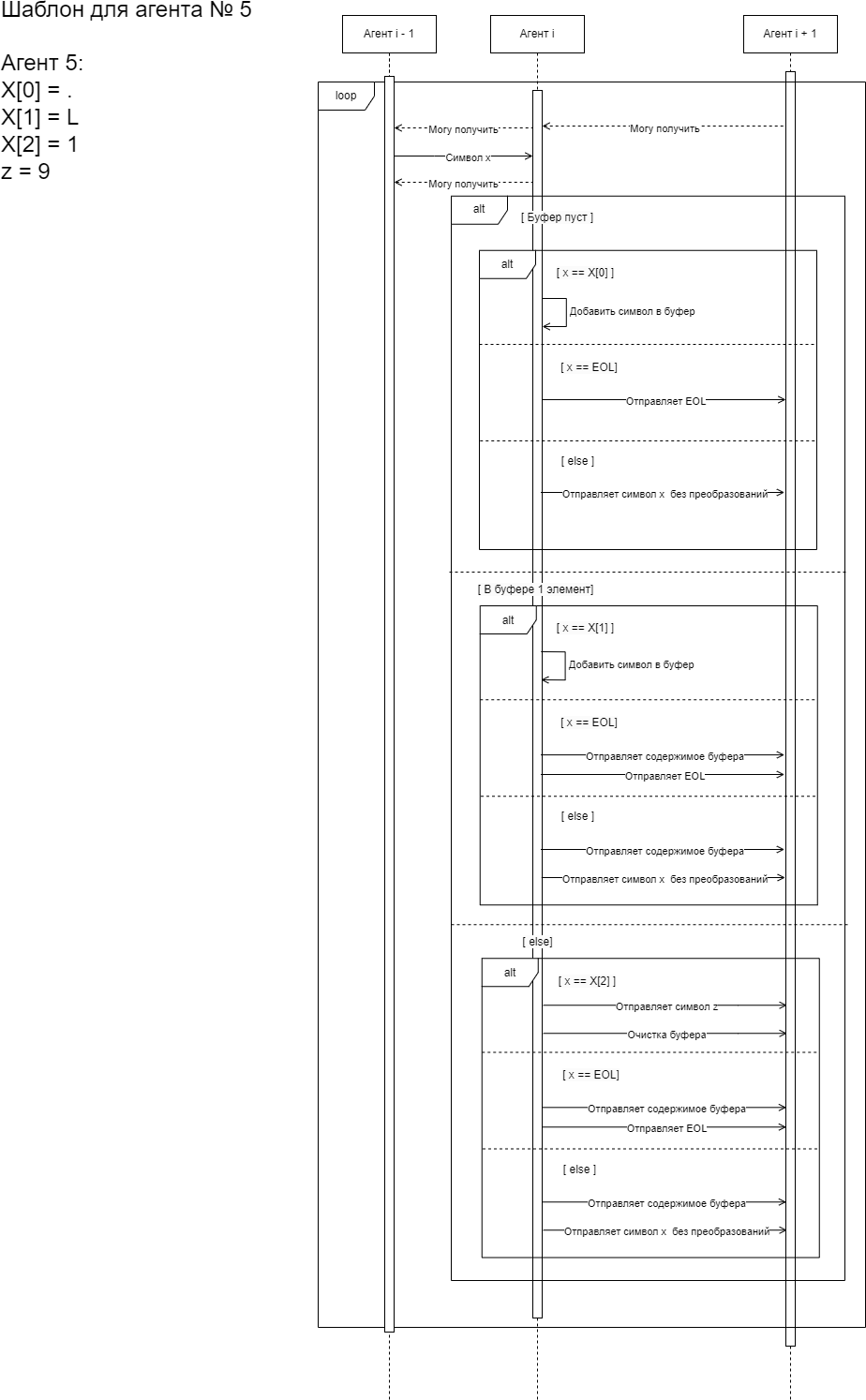


Схема работы агента 5

**Агент 6. Замена .L на A**

Шаблон входных данных: {.L}

Выходные данные: {A}

Принцип работы:

1. на вход агент получает символ, закодированный в виде битовой строки, и добавляет его в буфер;
2. если количество элементов в буфере равно 2:
   1. если содержимое буфера соответствует шаблону, то удалить все элементы буфера и добавить в буфер А
   2. если содержимое буфера не соответствует шаблону, то первый элемент в буфере подается на вход агента №7
3. если количество элементов в буфере меньше 2:
   1. если в буфере EOL, то EOL подается на вход агента №7 и агент №6 заканчивает работу
   2. если в буфере не EOL, повтор пунктов 1-3

**Агент 7. Замена A-Z на A**

Шаблон входных данных: [A, Z] , в ASCII X = [65, 90]

Выходные данные: z = {A}, в ASCII z = {65}

Для сравнения входящего символа на принадлежность множеству была использована математическая формула:

x’ = x - (min(max(65, x), 90) - 65) \* 90 / max(90, x)

Принцип работы:

1. на вход агент получает символ, закодированный в виде битовой строки;
2. если на вход поступил EOL, то на вход агенту №8 передается EOL;
3. если символ соответствует шаблону входных данных, символ заменяется на A и передается на вход агенту №8;
4. если символ не соответствует шаблону входных данных, символ передается на вход агенту №8.

**Агент 8. Замена A1 на A**

Шаблон входных данных: {A1}

Выходные данные: {A}

Принцип работы:

1. на вход агент получает символ, закодированный в виде битовой строки, и добавляет его в буфер;
2. если количество элементов в буфере равно 2:
   1. если содержимое буфера соответствует шаблону, то удалить 1 из буфера
   2. если содержимое буфера не соответствует шаблону, то первый элемент в буфере подается на вход агента №9
3. если количество элементов в буфере меньше 2:
   1. если в буфере EOL, то EOL подается на вход агента №9 и агент №8 заканчивает работу
   2. если в буфере не EOL, повтор пунктов 1-3

**Агент 9. Замена -> на ~**

Шаблон входных данных: {->}

Выходные данные: {~}

Принцип работы:

1. на вход агент получает символ, закодированный в виде битовой строки, и добавляет его в буфер;
2. если количество элементов в буфере равно 2:
   * если содержимое буфера соответствует шаблону, то удалить все элементы буфера и добавить ~
   * если содержимое буфера не соответствует шаблону, то первый элемент в буфере подается на вход агента №10
3. если количество элементов в буфере меньше 2:
   * если в буфере EOL, то EOL подается на вход агента №10 и агент №9 заканчивает работу
   * если в буфере не EOL, повтор пунктов 1-3

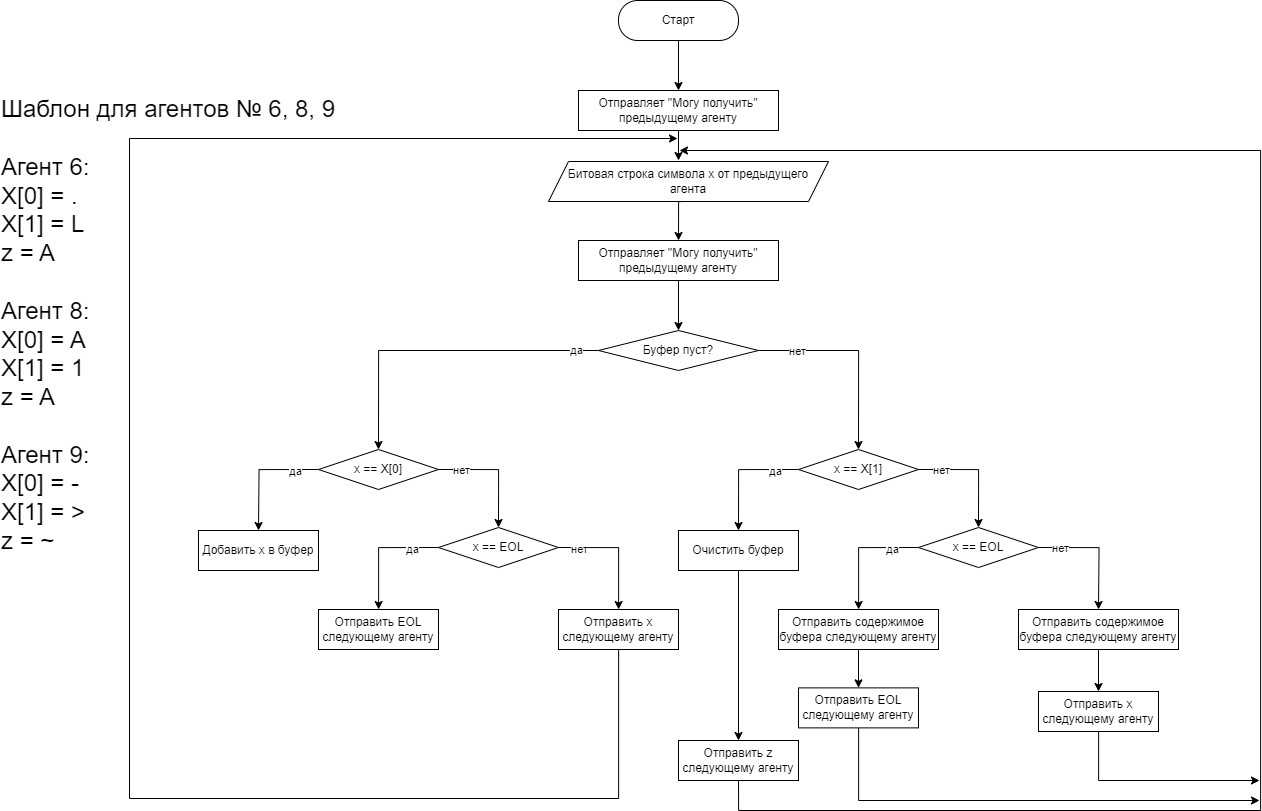


Схема работы агентов 6, 8, 9

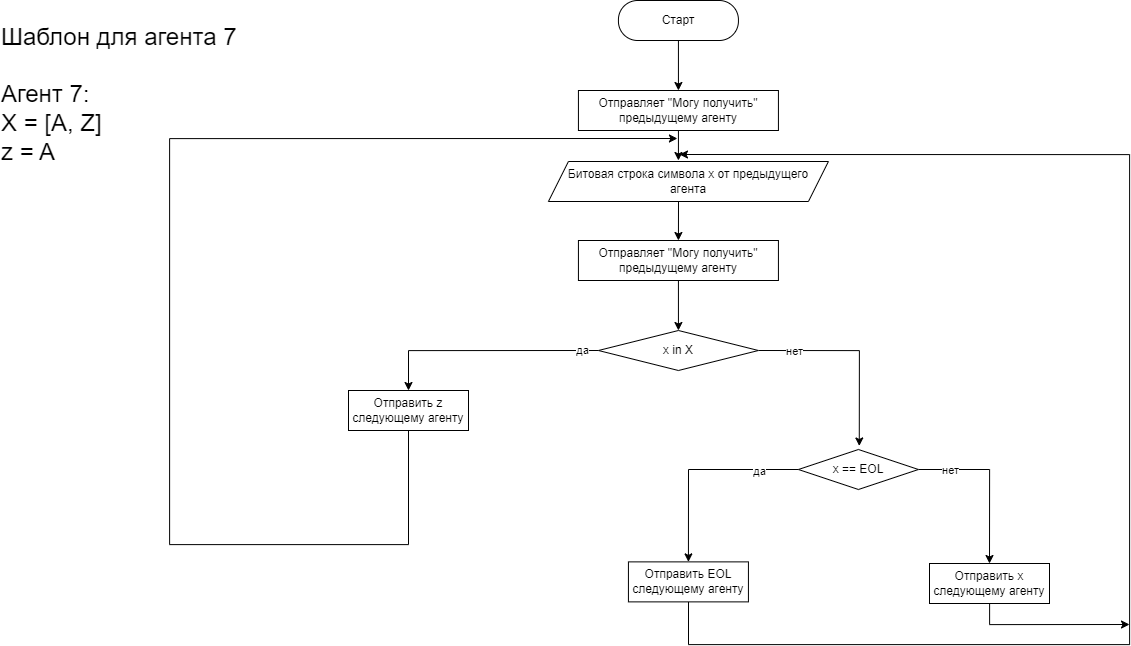


Схема работы агента 7

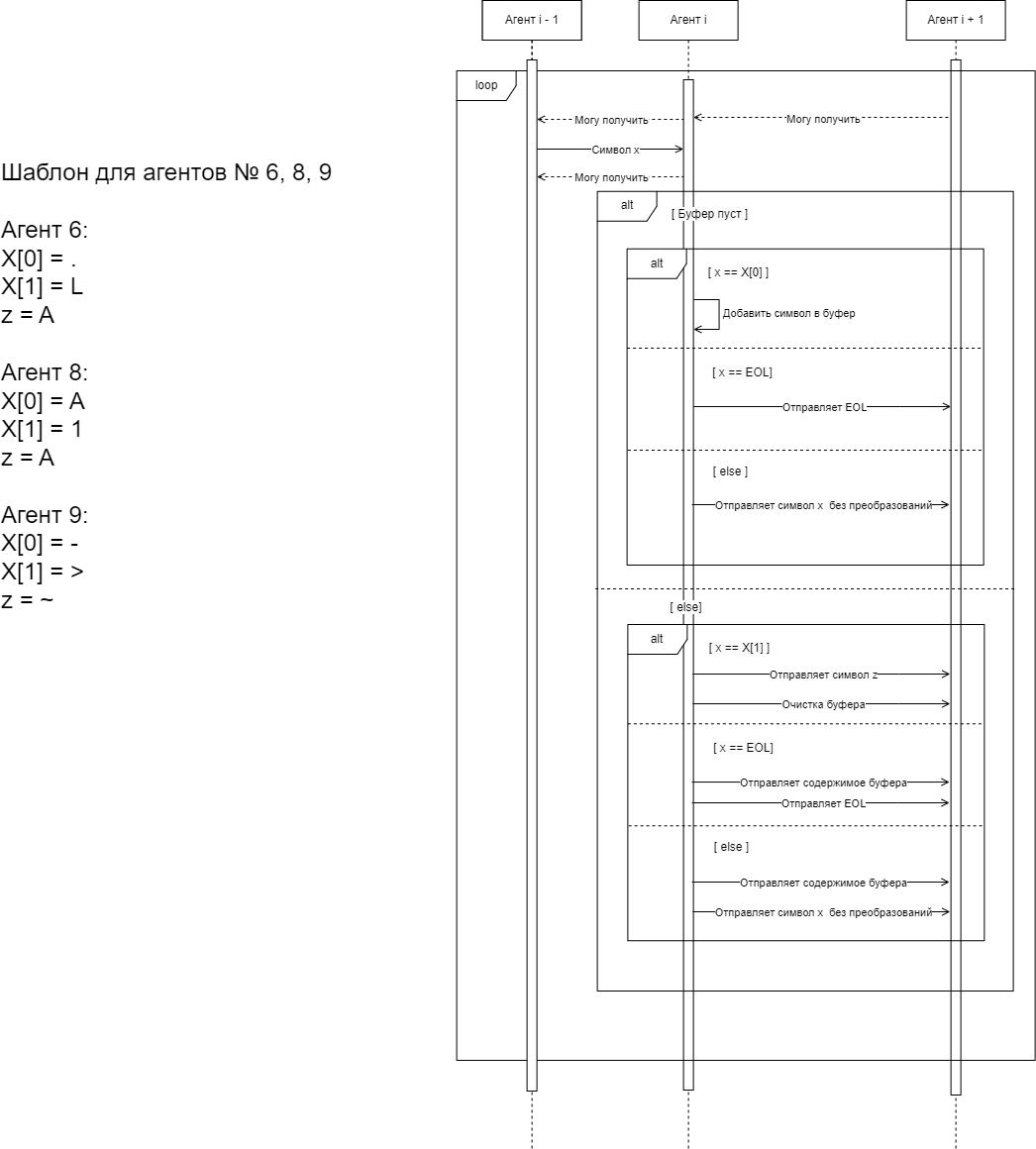


Диаграмма последовательности работы агентов 6, 8, 9

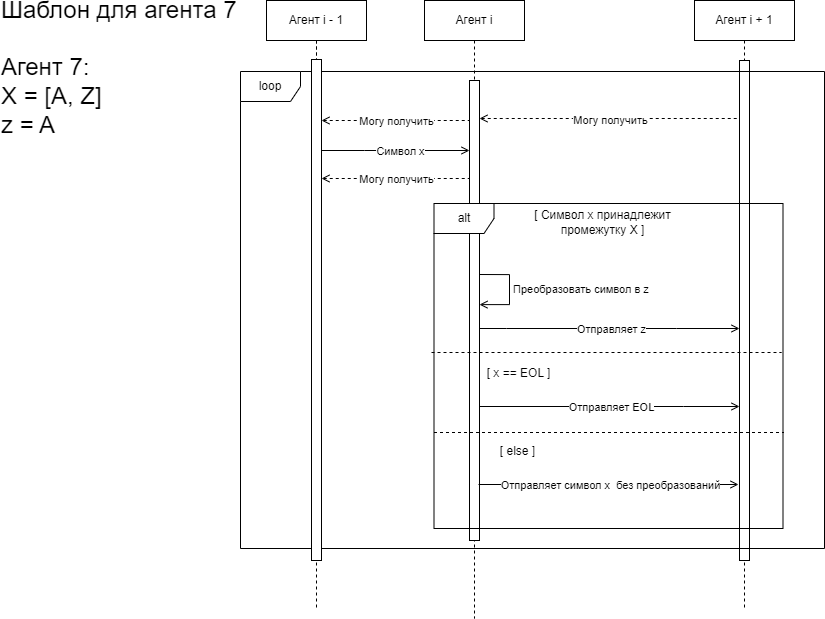


Диаграмма последовательности работы агента 7

**Агент 10. Замена \/, /\ на ~**

Шаблон входных данных: {\/, /\}

Выходные данные: {~}

Принцип работы:

1. на вход агент получает символ, закодированный в виде битовой строки, и добавляет его в буфер;
2. если количество элементов в буфере равно 2:
   1. если содержимое буфера соответствует одному из шаблонов, то удалить все элементы буфера и добавить ~
   2. если содержимое буфера не соответствует шаблону, то первый элемент в буфере подается на вход агента №11
3. если количество элементов в буфере меньше 2:
   1. если в буфере EOL, то EOL подается на вход агента №11 и агент №10 заканчивает работу
   2. если в буфере не EOL, повтор пунктов 1-3

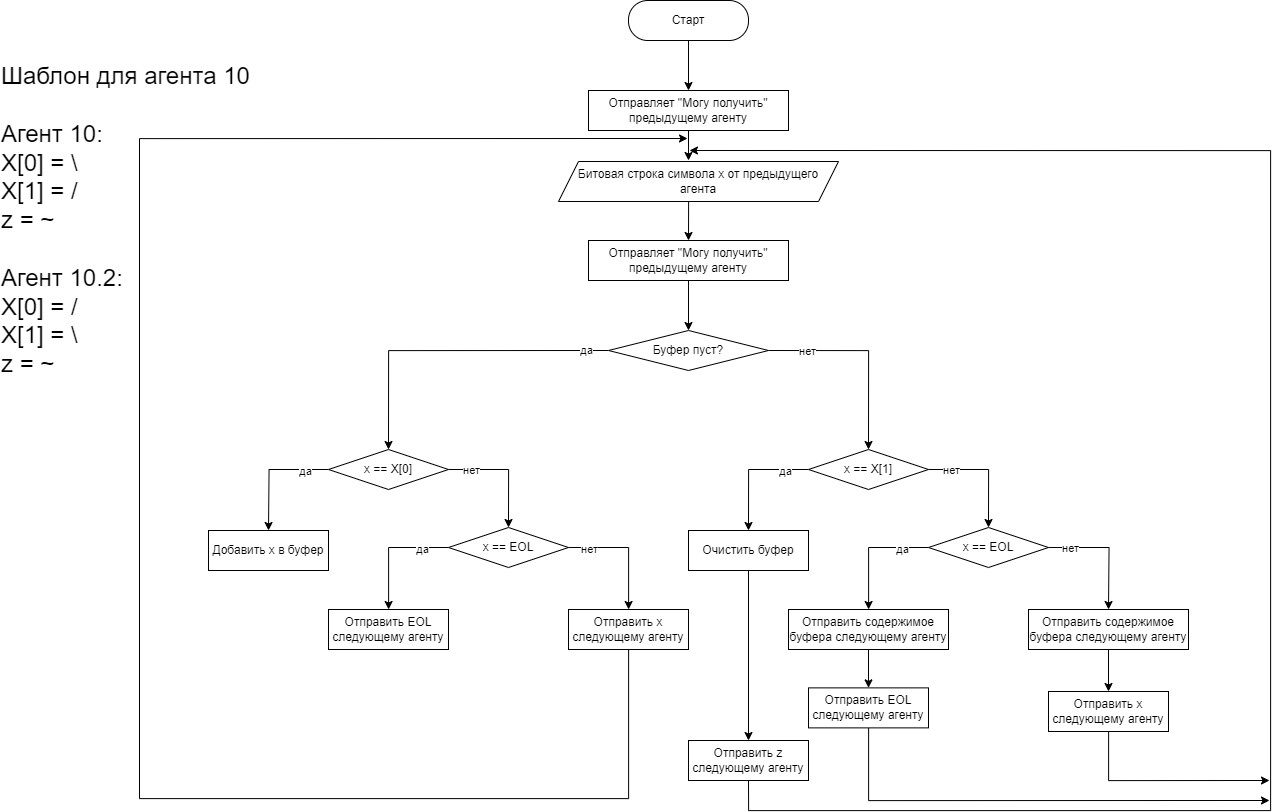


Схема работы агента 10 и 10.2

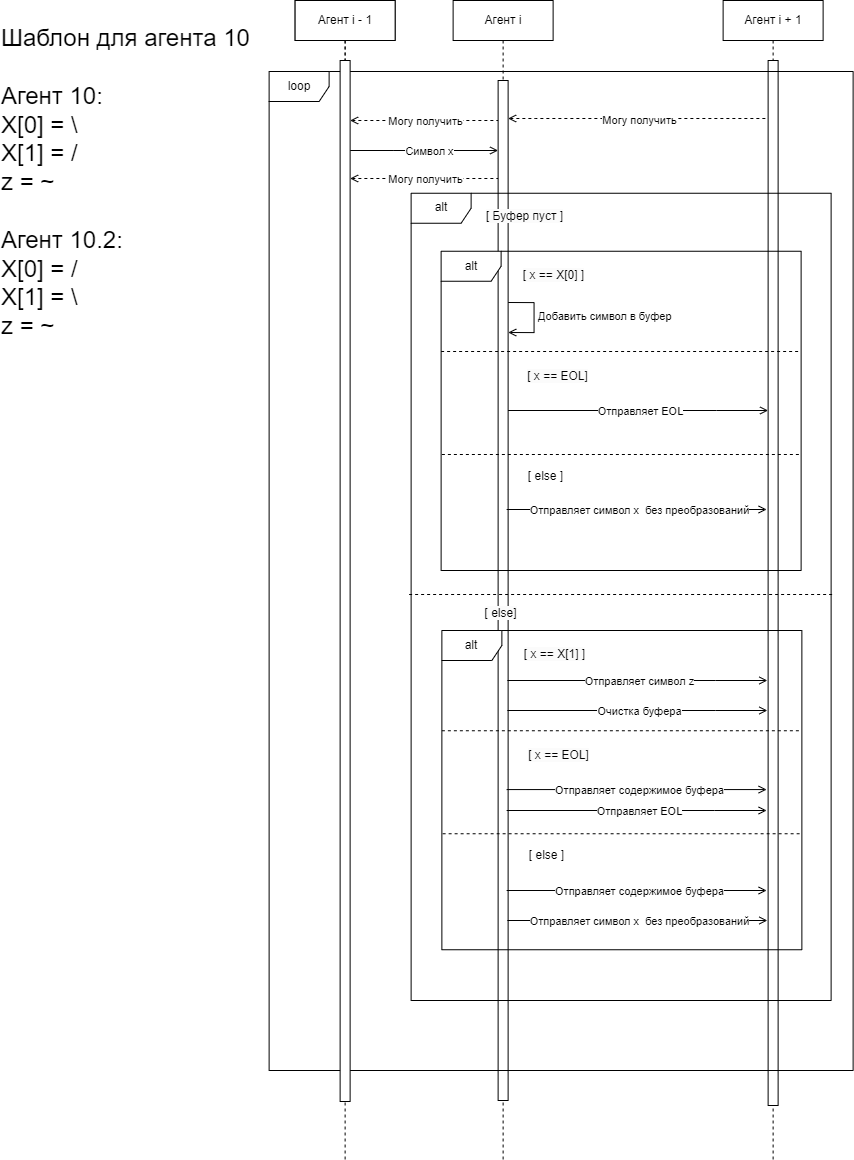


Диаграмма последовательности агента 10

**Агент 11. Замена ! на A~**

Шаблон входных данных: {!}

Выходные данные: {A~}

Принцип работы:

1. на вход агент получает символ, закодированный в виде битовой строки;
2. если на вход поступил EOL, то на вход агенту №12 передается EOL;
3. если символ соответствует шаблону входных данных, символ заменяется на A~ и передается на вход агенту №12;
4. если символ не соответствует шаблону входных данных, символ передается на вход агенту №12.

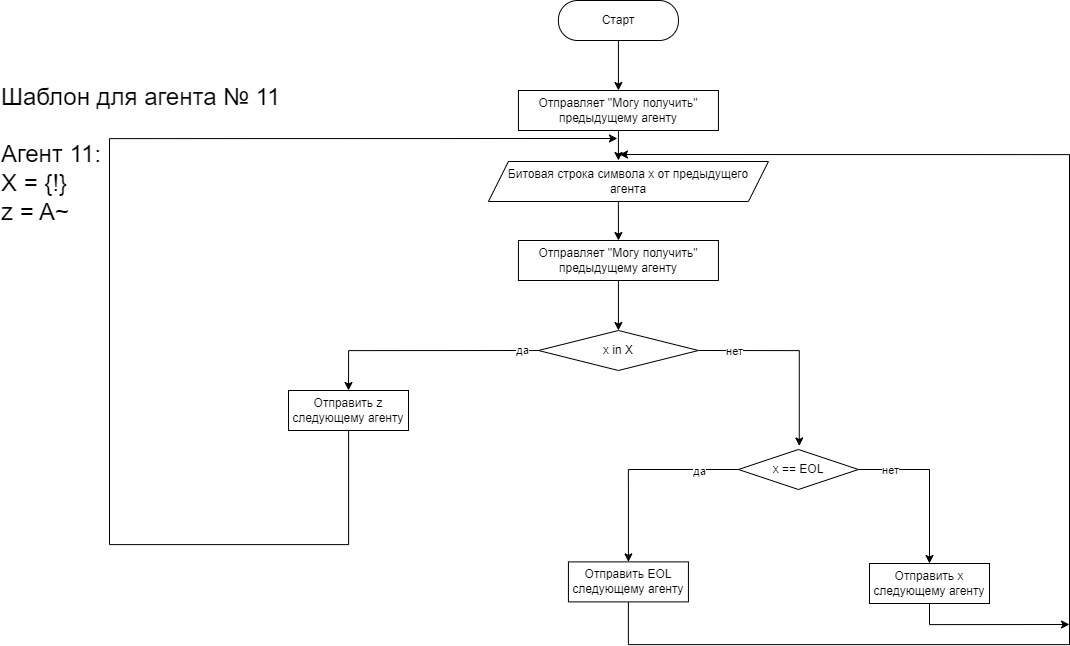


Схема работы агента 11

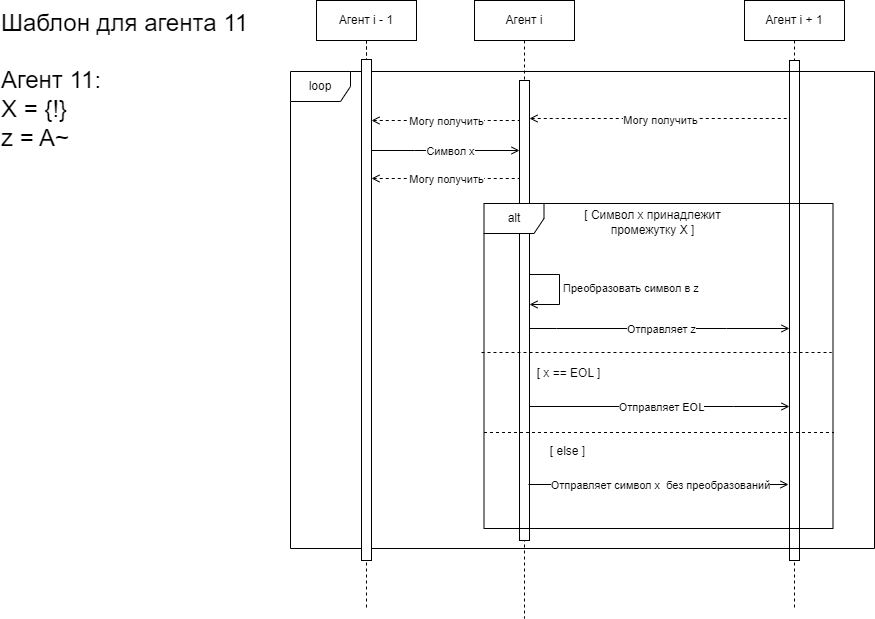


Диаграмма последовательности агента 11

**Агент 12. Установка в очередь**

Входные данные: символы строки из двух потоков: priority и исходного

Выходные данные: символы строки

Принцип работы:

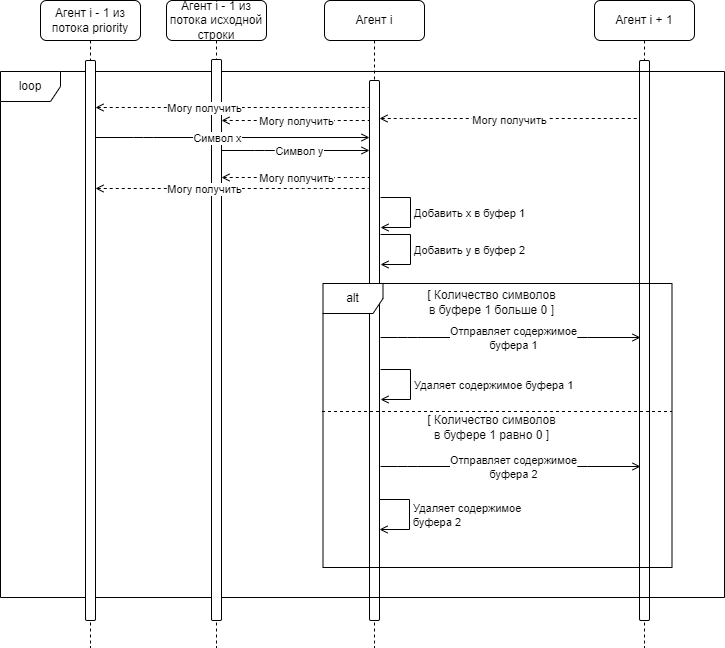
1. на вход из потока priority агент получает символ, закодированный в виде битовой строки, и добавляет его в буфер 1.
2. на вход из исходного потока получает символ в виде битовой строки и добавляет его в буфер 2.
3. если количество элементов в буфере 1 больше 0:
   1. отправляет агенту 13 содержимое буфера 1
4. если количество элементов в буфере равно 0
   1. ****отправляет агенту 13 содержимое буфера 2

Диаграмма последовательности работы агента 12

**Агент 13. Разделение на потоки**

Входные данные: символы строки

Выходные данные: символы строки, идущие по двум потокам данных: исходному и priority

Принцип работы:

1. на вход агент получает символ, закодированный в виде битовой строки
2. агент отправляет этот символ по потоку данных priority агенту 14 замены (A~A) на А
3. агент также отправляет этот символ по потоку исходных данных агенту 15 посимвольного сравнения строк

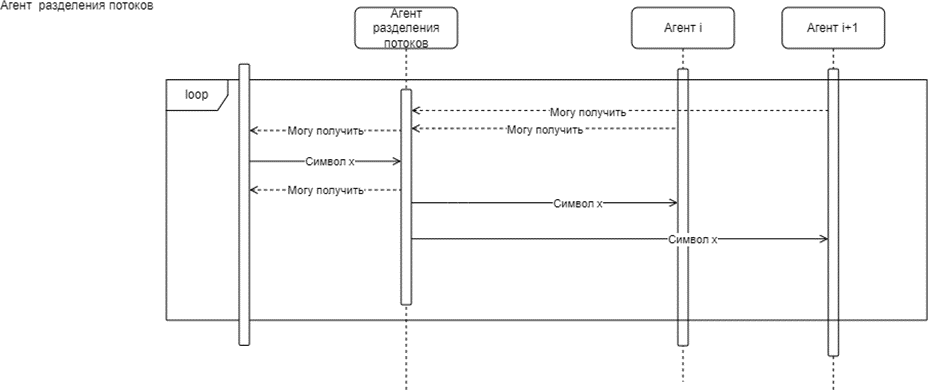


Диаграмма последовательности работы агента 13

**Агент 14. Замена (А~A) на А**

Шаблон входных данных: (A~A)

Выходные данные: A

Принцип работы:

1. на вход агент получает символ, закодированный в виде битовой строки, и добавляет его в буфер.
2. если количество элементов в буфере равно 5:
   1. если содержимое буфера соответствует шаблону, то удалить все элементы буфера и добавить в буфер А
   2. если содержимое буфера не соответствует шаблону, то первый элемент в буфере подается на вход агента №15
3. если количество элементов в буфере меньше 5:
   1. если в буфере только EOL, то EOL подается на вход агента №15 и агент №14 заканчивает работу
   2. если в буфере не EOL, повтор пунктов 1-3

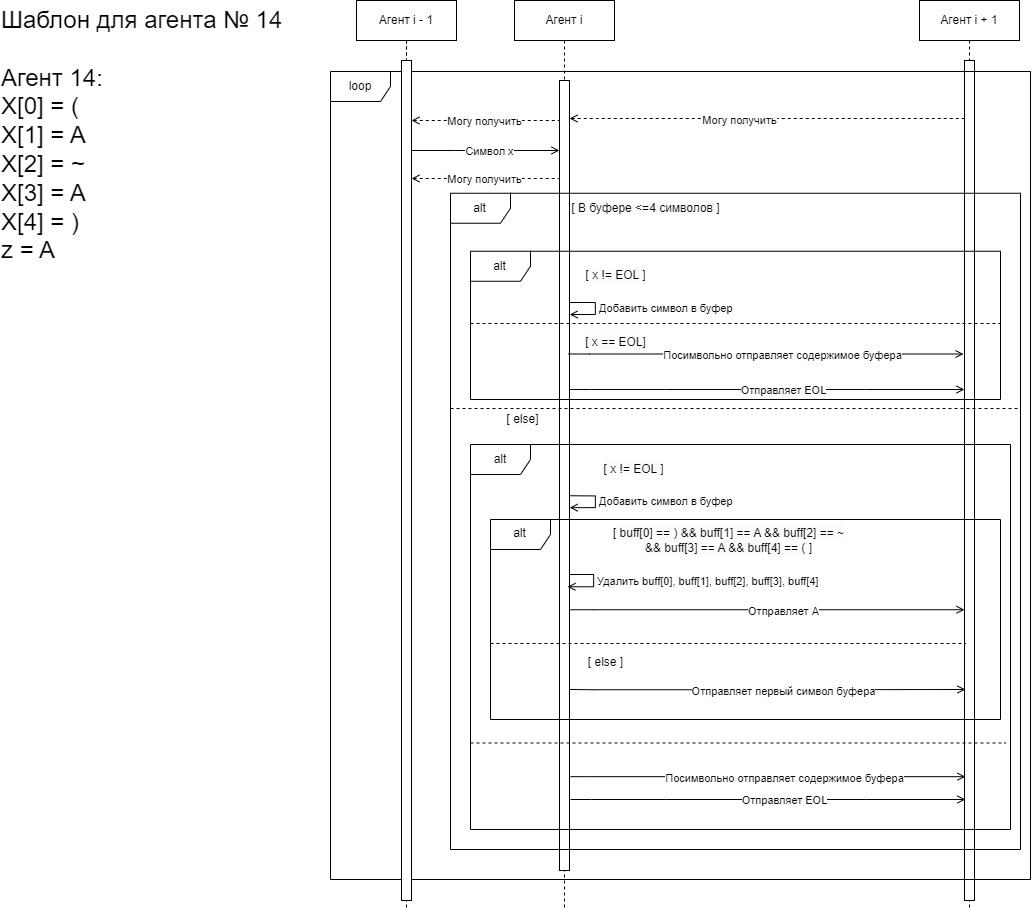


Диаграмма последовательности работы агента 14

**Агент 15. Посимвольное сравнение строк**

Шаблон входных данных: символы строки из потока исходных данных и потока priority

Выходные данные: символы строки потока priority

Принцип работы:

1. на вход агент со стороны потока данных priority получает символ в виде битовой строки и записывает его в буфер 1.
2. на вход агент со стороны потока данных исходной строки получает символ в виде битовой строки и записывает его в буфер 2.
3. если первые элементы обоих буферов не равны между собой:
   1. буфер 2 очищается
   2. первый элемент буфера 1 удаляется из буфера 1 и отправляется к агенту 12
4. если последний элемент буфера 2 является EOL и последний элемент буфера 1 является EOL:
   1. если содержимое буферов 1 и 2 совпадает:
      1. отправляет первый элемент буфера 1 агенту 16 и удаляет его из буфера
5. если буфер 1 не пуст, повторяем шаги 1-4

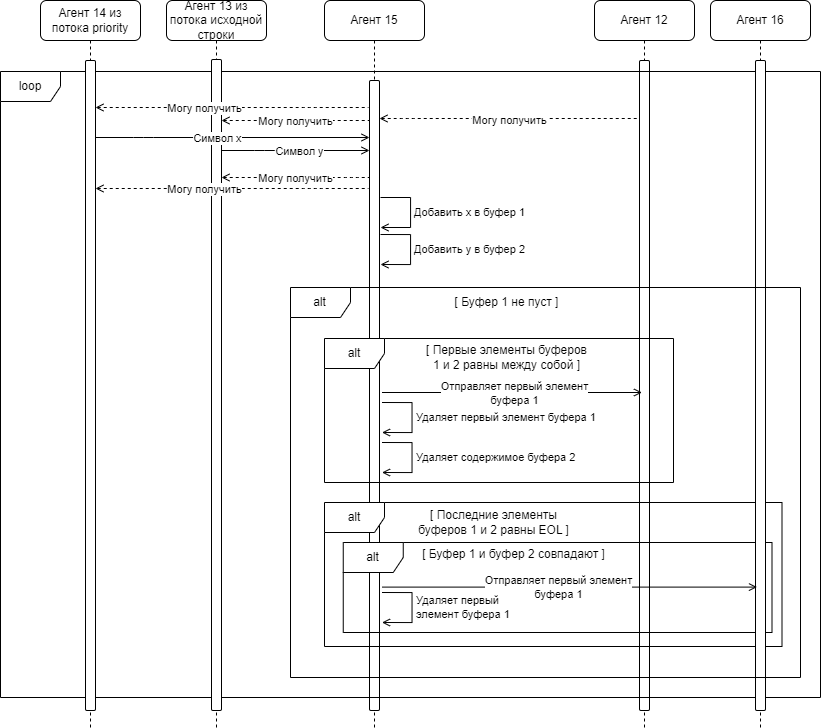


Диаграмма последовательности работы агента 15

**Агент 16. Сравнение конечной формулы с А**

Входные данные: символы строки

Выходные данные: {True, False}

Принцип работы:

1. на вход агент получает символ, закодированный в виде битовой строки.
2. если буфер пуст:
   1. если символ является EOL:
      * супервизору отправляется False (исходная строка не является формулой)
   2. если символ не является EOL:
      * символ добавляется в буфер
3. если буфер не пуст:
   1. если символ не является EOL:
      * символ добавляется в буфер
   2. если символ является EOL:
      * если в буфере 1 символ и он равен А:
        + супервизору отправляется True (исходная строка является формулой)
      * если в буфере не 1 элемент или первый элемент не равен А:
        + супервизору отправляется False (исходная строка не является формулой)

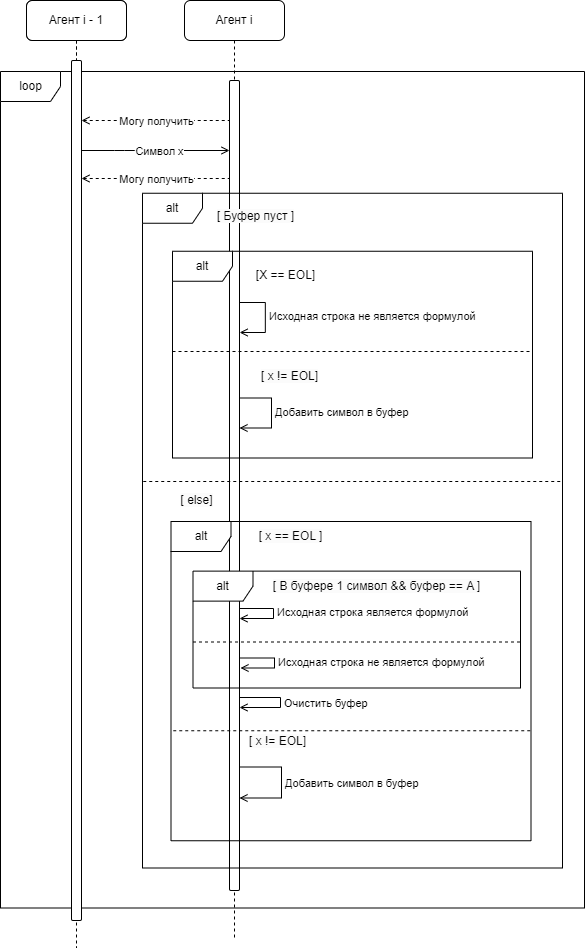


Диаграмма последовательности работы агента 16

**Программная реализация**

* **Команды простой замены**

|  |  |  |
| --- | --- | --- |
| Команда | Операнд | Пояснение |
| pattern\_from\_1\_9\_to\_1 |  | Замена [1, 9] на 1 |
| pattern\_from\_A\_Z\_to\_A |  | Замена [A, Z] на A |
| pattern\_from\_no\_to\_A\_equal |  | Замена ! на A~ |
| pattern\_from\_A1\_to\_A |  | Замена A1 на A |

Примеры

|  |  |  |
| --- | --- | --- |
| Стек до | Код | Стек после |
| 52 12 13 | #formula-linter /new\_pattern\_from\_1\_9\_to\_1 | 49 12 13 |
| 86 34 12 | #formula-linter /new\_pattern\_from\_A\_Z\_to\_A | 65 34 12 |
| 33 66 49 | #formula-linter /pattern\_from\_no\_to\_A\_equal | 126 65 66 49 |
| [1 49 65] | #formula-linter /pattern\_from\_A1\_to\_A | [1 65] |

* **Пакет вспомогательных команд a1\_to\_a (аналогичные для false\_to\_a, if\_to\_equal, and\_to\_equal, or\_to\_equal)**

|  |  |  |
| --- | --- | --- |
| Команда | Операнд | Пояснение |
| a1\_to\_a\_agent |  | Routine по запуску агента |
| a1\_to\_a\_agent\_src |  | Запуск агента |
| a1\_to\_a\_step |  | Routine по выполнению шага агента |
| a1\_to\_a\_step\_src |  | Выполнение шага агента |
| check\_pattern\_in\_stack |  | Проверка на принадлежность шаблона |
| compare\_to\_1 |  | Сравнение с единицей (49 в ASCII) |
| compare\_to\_a |  | Сравнение с A (65 в ASCII) |
| compare\_to\_not\_eol |  | Сравнение с концом строки (терминальная 1) |
| extract\_buffer |  | Routine для получения элемента из буфера |
| extract\_buffer\_src |  | Набор команд для получения элемента из буфера |
| master |  | Routine для подготовки агента к запуску |
| master\_src |  | Подготовка агента к запуску |
| result\_src |  | Перемещение результата из буфера в вывод |
| result |  | Routine для перемещения результата из буфера в вывод |
| swap\_src |  | Перемещение элементов между стеками буфера и ввода-вывода |
| swap |  | Routine для перемещения элементов между стеками буфера и ввода-вывода |

**Примеры**

|  |  |  |
| --- | --- | --- |
| Стек до | Команда | Стек после |
| [a1\_to\_a\_step] [input] [output] | #formula-linter/a1\_to\_a/a1\_to\_a\_agent | [a1\_to\_a\_agent] [a1\_to\_a\_step] [input] [output] |
| [a1\_to\_a\_step] [input] [output] | #formula-linter/a1\_to\_a/a1\_to\_a\_agent\_src | [a1\_to\_a\_agent] [a1\_to\_a\_step] [input] [output] |
| [input] [output] | #formula-linter/a1\_to\_a/a1\_to\_a\_step | [a1\_to\_a\_step] [input] [output] |
| [a1\_to\_a\_step] [input] [output] | #formula-linter/a1\_to\_a/a1\_to\_a\_step\_src | [a1\_to\_a\_step] [input] [output] |
| [input] [output] | #formula-linter/a1\_to\_a/check\_pattern\_in\_stack | 1 |
| 65 | #formula-linter/a1\_to\_a/compare\_to\_1 | 0 65 |
| 65 | #formula-linter/a1\_to\_a/compare\_to\_a | 1 65 |
| 1 | #formula-linter/a1\_to\_a/compare\_to\_not\_eol | 0 1 |
| [1 49 65] | #formula-linter/a1\_to\_a/extract\_buffer | 1 [49 65] |
| [1 49 65] | #formula-linter/a1\_to\_a/extract\_buffer\_src | 1 [49 65] |
| [input] [output] | #formula-linter/a1\_to\_a/master | [master] [input] [output] |
| [input] [output] | #formula-linter/a1\_to\_a/master\_src | [master] [input] [output] |
| [1 49] [] [65] | #formula-linter/a1\_to\_a/result | [1 49] [65] [] |
| [1 49] [] [65] | #formula-linter/a1\_to\_a/result\_src | [1 49] [65] [] |
| [1 49] [] [65] | #formula-linter/a1\_to\_a/swap | [1] [65] [49] |
| [1 49] [] [65] | #formula-linter/a1\_to\_a/swap\_src | [1] [65] [49] |

* **Пакет вспомогательных команд utility**

|  |  |  |
| --- | --- | --- |
| Команда | Операнд | Пояснение |
| compare |  | Сравнение двух лежащих на вершине стека чисел (операция ==) |
| not\_compare |  | Определение неравенства двух лежащих на вершине стека чисел (операция !=) |
| extract |  | Извлечение из узла, лежащего на вершине стека, первого в очереди числа |
| if\_full |  | Проверка буфера на наличие элементов |
| prepare\_for |  | Подготовка стека для условного оператора |

**Примеры**

|  |  |  |
| --- | --- | --- |
| Стек до | Код | Стек после |
| 56 56 18 | #formula-linter/utility/compare | 1 56 56 18 |
| 56 56 18 | #formula-linter/utility/not\_compare | 0 56 56 18 |
| [48 76 54] | #formula-linter/utility/extract | 48 [48 76 54] |
| [23] | #formula-linter/utility/if\_full | 1 [23] |
| 1/0 | #formula-linter/utility/prepare\_for | [0 1]/[] |