

Документация SCL-machine

Программный вариант реализации логической машины интерпретации логических sc-моделей компьютерных систем

```

:= [SCL-machine]
:= [ostis-inference]
⇒ декомпозиция программной системы*:
{
• База знаний SCL-machine
• Решатель задач SCL-machine
• Интерфейс SCL-machine
}
⇒ реализованные логические связи*:
{
• импликация*
• дизъюнкция*
• конъюнкция*
• отрицание*
}
⇒ не реализованные логические связи*:
{
• эквиваленция*
• строгая дизъюнкция*
}

```

Решатель задач SCL-machine

```

⇒ обобщённая декомпозиция*:
{
• Агент прямого логического вывода
• Агент обратного логического вывода
⇒ примечание*:
[Не реализовано.]
}

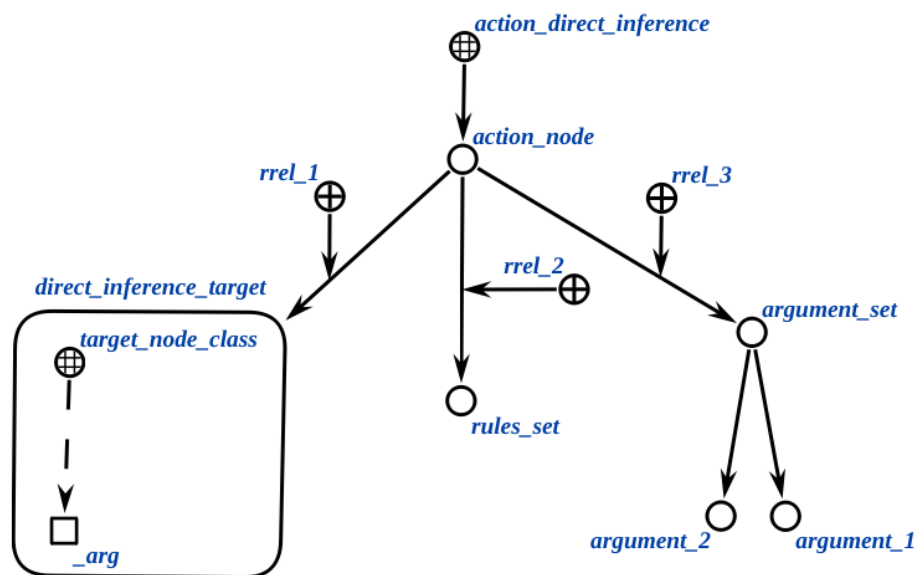
```

Агент прямого логического вывода

```

⇒ пример входной конструкции*:
[

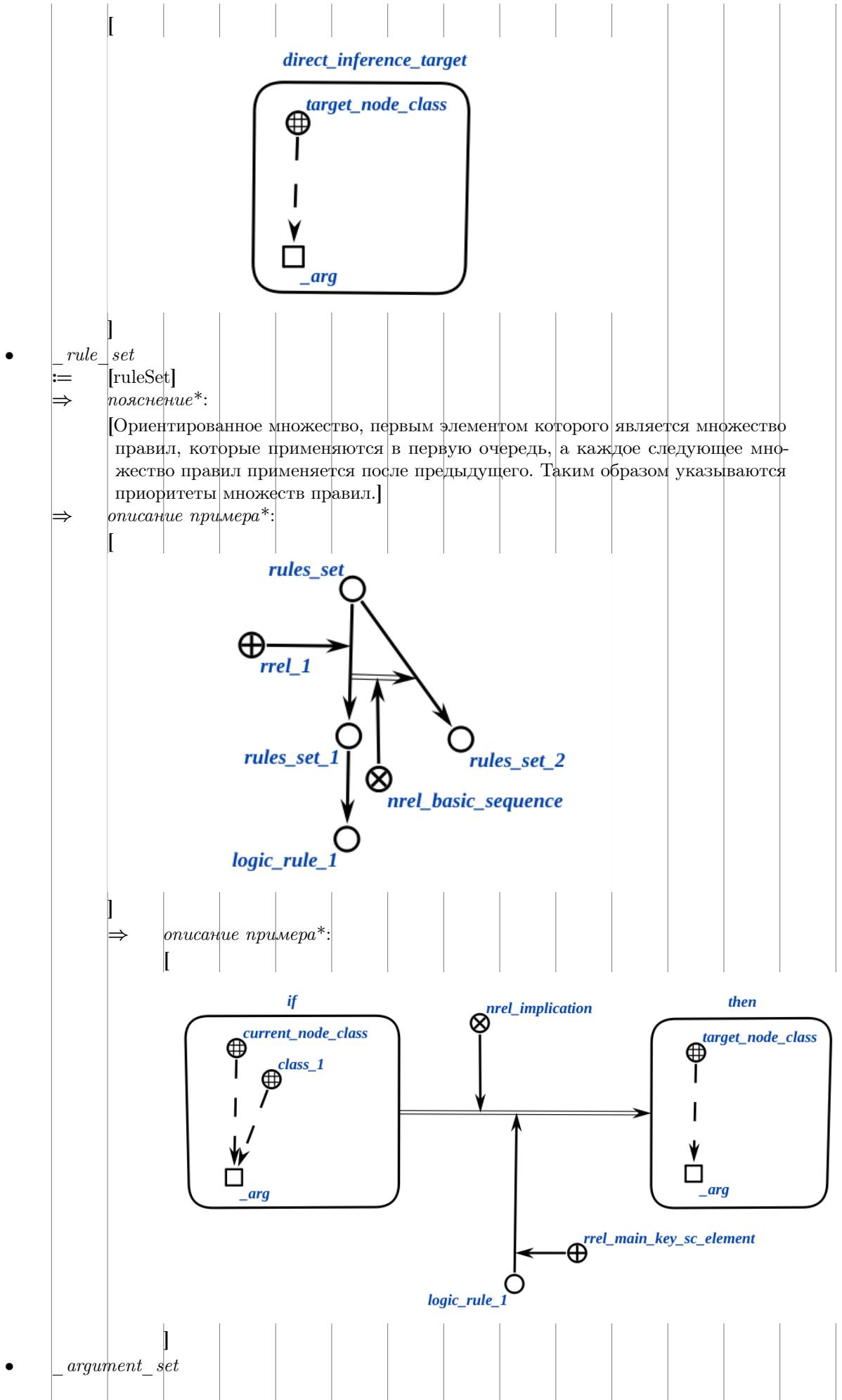
```



```

]
⇒ аргументы агента*:
<• _target_template
:= [targetTemplate]
:= [targetStatement]
⇒ пояснение*:
[Шаблон, успешный поиск которого показывает, что цель логического вывода
достигнута и применение правил можно прекратить.]
⇒ описание примера*:

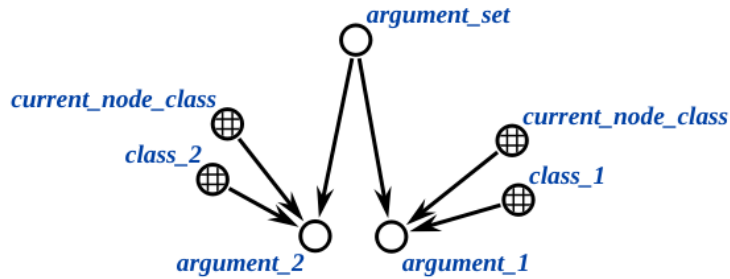
```



```

:= [argumentSet]
⇒ пояснение*:
  [Множество тех элементов, которые должны быть подставлены как значение
  переменных шаблона цели.]
⇒ пояснение*:
  [В данном примере значением переменной _x в шаблоне цели может
  быть sc-узел argument_1, тогда значением _y будет argument_2. Также
  значением переменной _x может быть sc-узел argument_2, а значением
  _y – argument_1.]
⇒ описание примера*:
  [

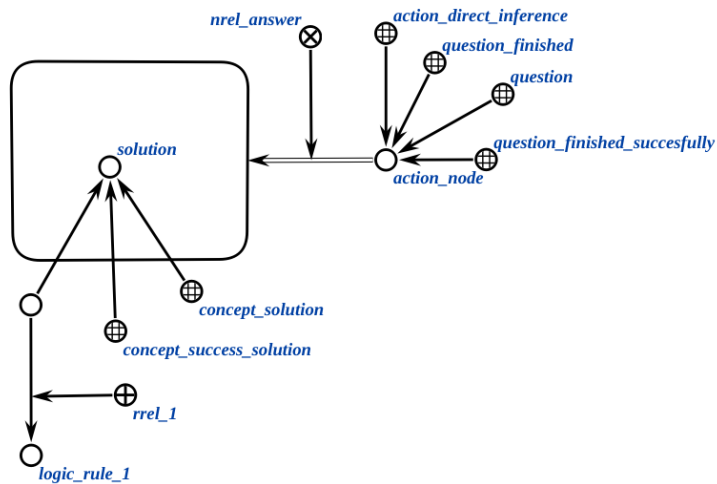
```



```

  ]
}
⇒ пример выходной конструкции*:
  [

```



```

  ]
⇒ обобщённый алгоритм*:
  {
  • [Получение параметров агента, проверка их валидности. Вызов агента;]
  • [Проверка, достигнута ли уже цель в базе знаний;]
  ⇒ примечание*:
    [Выполняется поиск по шаблону target template с параметрами шаблона arguments
    set.]
  • [Построение вектора очереди правил на основе множества правил. Цикл по всем
    правилам и пока не достигнута цель;]
  ⇒ циклические операции*:
    {
    • [Получение посылки логического правила;]
    • [Определение типа посылки (связка конъюнкции, дизъюнкции, отрицания
      или атомарная логическая формула);]
    • [Проверка истинности посылки в зависимости от её типа;]
    ⇒ замечание*:
      [Конъюнкция, дизъюнкция, отрицание работают нестабильно.]
    }
  }

```

• [Генерация по шаблону следствия;]
 • [Добавление в дерево решений узла правила.]
 ⇒ *примечание**:
 [Смотрите пример выходной конструкции.]
 }
 • [Возврат дерева применённых правил.]
 }
 ⇒ *недостатки текущего состояния**:
 {
 • [Импликация в текущем состоянии интерпретируется не как логическая связка, а как отношение **выводимости***, то есть импликация не возвращает логическую константу, а генерирует новые знания. Вместо этого должны использоваться правила вывода, например, *Modes ponens* и другие правила, в процессе логического вывода.]
 • [В структуру ответа агента входит только узел *solution*, а не вся структура решения.]
 }