

# Документация SCL-machine

### **Программный вариант реализации машины логического вывода SCL**

```

:= [Машина логического вывода SCL]
:= [scl-машина]
:= [SCL-machine]
:= [ostis-inference]
∈ машина обработки знаний
⇐ программная модель*:
  Абстрактная scl-машина
⇒ внутренний язык*:
  Язык SCL
⇒ 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 применения правил вывода
      ⇒ примечание*:
        [Не реализовано.]
    • Агент эквивалентных преобразований логической формулы
      ⇒ примечание*:
        [Не реализовано.]
  }

```

### **Агент прямого логического вывода**

```

:= [sc-агент прямого логического вывода]

```

```

⇒ примечание*:

```

[Задачей sc-агента прямого логического вывода является генерация новых знаний на основе некоторых логических утверждений. Данный sc-агент активируется при появлении в sc-памяти инициированного действия, принадлежащего классу *действие прямого логического вывода*. После проверки sc-агентом условия инициирования выполняется процесс прямого логического вывода.]

```

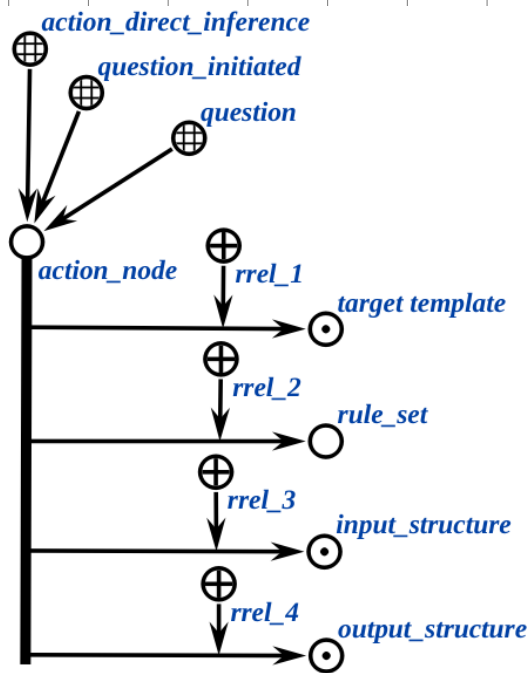
⇒ пример входной конструкции*:

```

```

[

```



⇒

аргументы агента\*:

⟨• *\_target\_template*

:= [targetTemplate]

:= [targetStatement]

:= [ожидаемый результат выполнения логического вывода]

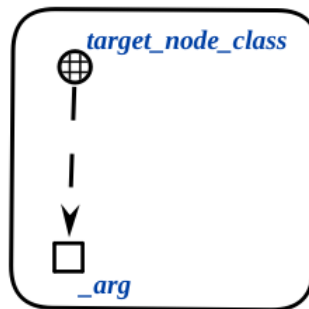
⇒ *пояснение\**:

[Шаблон, успешный поиск которого показывает, что цель логического вывода достигнута и применение правил можно прекратить.]

⇒ *описание примера\**:

[

*direct\_inference\_target*



]

• *\_rule\_set*

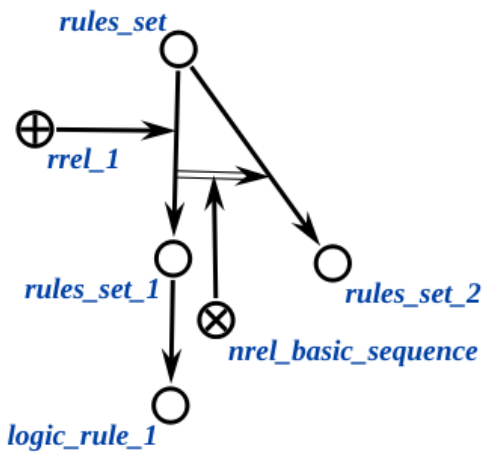
:= [ruleSet]

⇒ *пояснение\**:

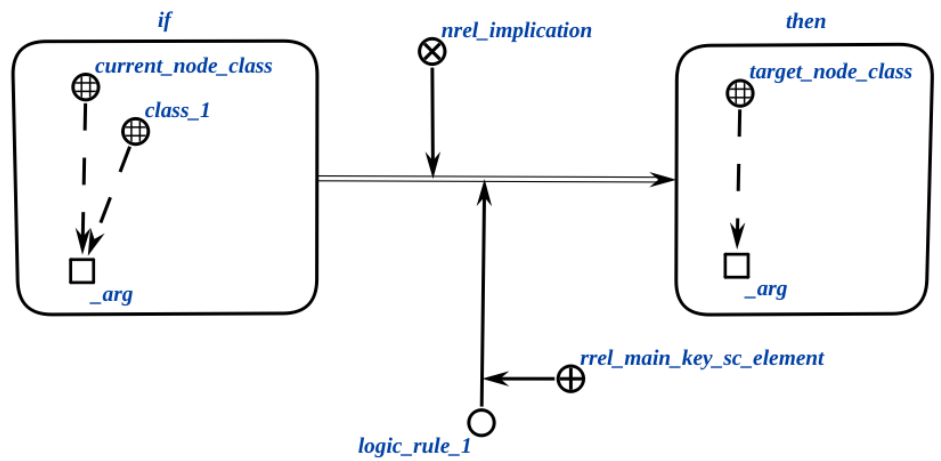
[Ориентированное множество множеств правил, применяя которые требуется совершить логический вывод. Первым элементом множества является множество правил, которые применяются в первую очередь, а каждое следующее множество правил применяется после предыдущего. Таким образом указываются приоритеты множеств правил.]

⇒ *описание примера\**:

[



⇒ описание примера\*:  
[



• `_input_structure`

:= [inputStructure]

⇒ пояснение\*:

[Структура, элементы которой используются при применении правил. Каждый sc-узел этой структуры добавляется во множество аргументов, которые должны быть подставлены как значение переменных шаблона цели.]

⇒ аналог\*:

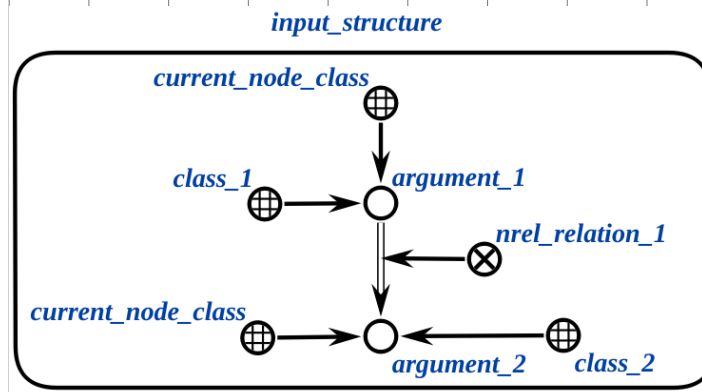
`_argument_set`

⇒ примечание\*:

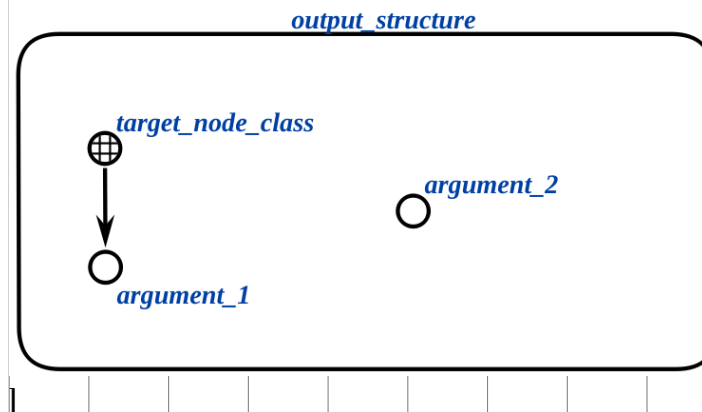
[Можно использовать структуру всей базы знаний системы, например, sc-узел **База знаний IMS**.]

⇒ описание примера\*:

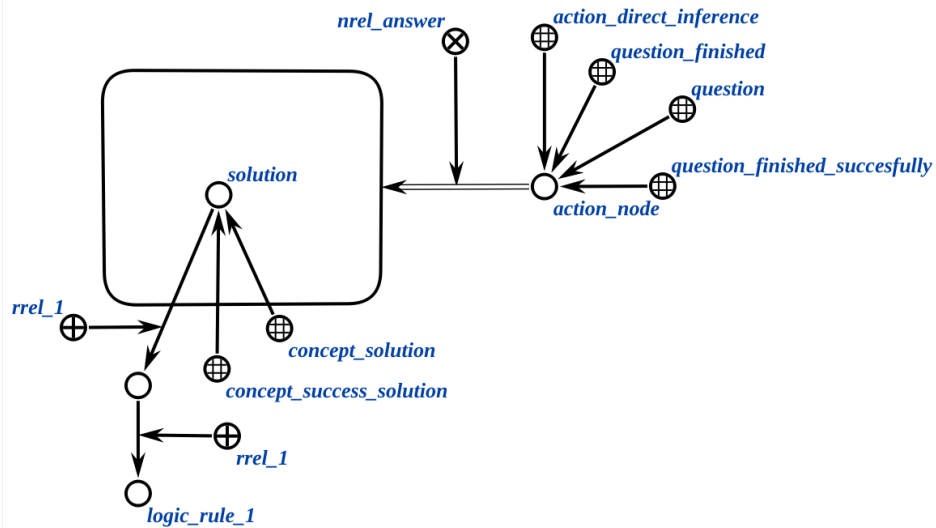
[



- *\_output\_structure*  
 := [outputStructure]  
 ⇒ *пояснение\**:  
 [Структура, в которую добавляются сгенерированные в ходе логического вывода конструкции.]  
 ⇒ *примечание\**:  
 [можно использовать структуру всей базы знаний системы, например, sc-узел *База знаний IMS*.]  
 ⇒ *описание примера\**:  
 [



- ⇒ *ответ агента\**:  
*ответ агента прямого логического вывода*  
 ⇒ *примечание\**:  
 [В результате выполнения агентом логического вывода действия, в sc-памяти формируется sc-структура, представляющая собой дерево решения. Это дерево состоит из последовательности узлов, представляющих собой применённые правила, которые привели к появлению в sc-памяти требуемых знаний. Такое дерево может быть пустым в случае, если требуемую структуру не удалось сгенерировать в ходе логического вывода.]  
 ⇒ *описание примера\**:  
 [



⇒ *примечание\**:

[Работа агента заключается в последовательном применении правил из входного множества правил, генерируя структуры, если атомарная формула принадлежит классу формул для генерации (*concept\_formula\_for\_generation*). Если правило применилось успешно, то создаётся отношение *nrel\_satisfiable\_formula* между правилом и моделью, на которой это правило выполнимо. Если правило применилось безуспешно, то оно добавляется во множество безуспешно применённых правил, которые применяются повторно в случае успешного применения какого-либо другого правила. Также после каждого успешного применения правила проверяется, достигнута ли цель (если она передана), и, если цель достигнута, выполнение агента завершается успешно и остальные правила не применяются.]

⇒ *обобщённый алгоритм\**:

- [Получение параметров агента, вызов агента;]
- [Получение всех sc-узлов из inputStructure, если структура валидна, заполнение ими списка аргументов;]
- [Проверка, достигнута ли уже цель в базе знаний с полученными аргументами;]

⇒ *примечание\**:

[Выполняется поиск по шаблону target template с параметрами структуры \_input\_structure. Если шаблон найден, агент завершает работу, возвращает узел, принадлежащий *concept\_success\_solution*.]

- [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 ответа агента.]

- [Формирование дерева применённых правил.]

⇒ *недостатки текущего состояния\**:

- [В текущем состоянии не реализован механизм применения правил вывода, вместо него указываются формулы для генерации, используя класс *concept\_formula\_for\_generation*.]
- [Генерируются только атомарные формулы.]
- [Логическая связка отрицания некорректно работает с подстановками.]
- [В структуру ответа агента входит только узел solution, а не вся структура решения.]
-

⇒

```

    }
    [Входная структура интерпретируется как параметры, которые подставляются в логические
    формулы, а не как структура, в которой нужно искать.]
    }
    преимущества текущего состояния*:
    {
    • [Агент работает корректно при передаче параметров в соответствии с предыдущим вариантом
    его реализации.]
    • [Проверка входных параметров не только по невалидности sc-узла, но и проверка на непустое
    множество.]
    }
  
```