

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.]

```

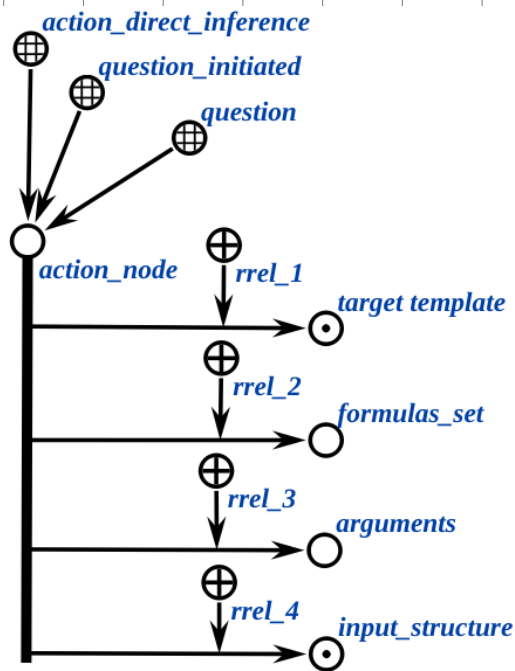
⇒ пример входной конструкции*:

```

```

[

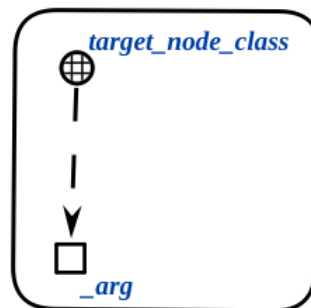
```



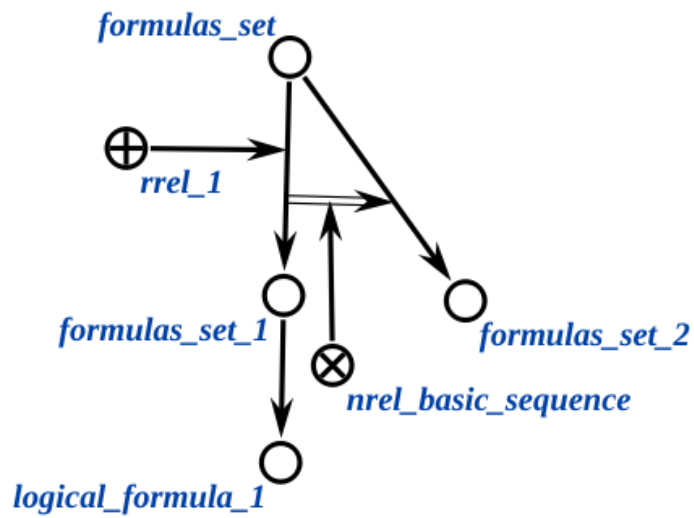
⇒ *аргументы агента**:

- *_target_template*
 - := [targetTemplate]
 - := [targetStatement]
 - := [шаблон цели]
 - := [ожидаемый результат выполнения логического вывода]
- ⇒ *пояснение**:
- ⇒ [Шаблон, успешный поиск которого показывает, что цель логического вывода достигнута и применение правил можно прекратить.]
- ⇒ *описание примера**:
- [

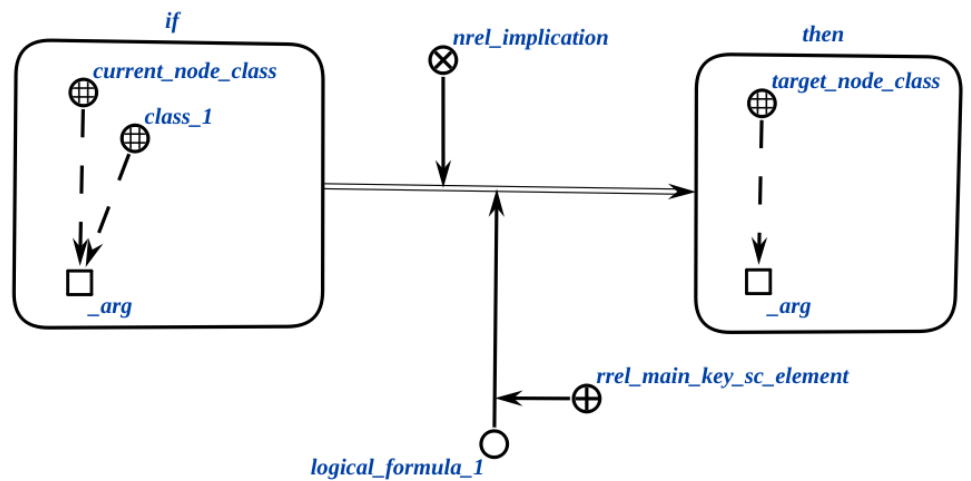
direct_inference_target



- *_formulas_set*
 - := [formulasSet]
 - ⇒ *пояснение**:
 - ⇒ [Ориентированное множество множеств формул, применяя которые требуется совершить логический вывод. Первым элементом множества является множество формул, которые применяются в первую очередь, а каждое следующее множество формул применяется после предыдущего. Таким образом указываются приоритеты множеств формул.]
 - ⇒ *описание примера**:
 - [



⇒ описание примера*:



- arguments

⇒ пояснение*:

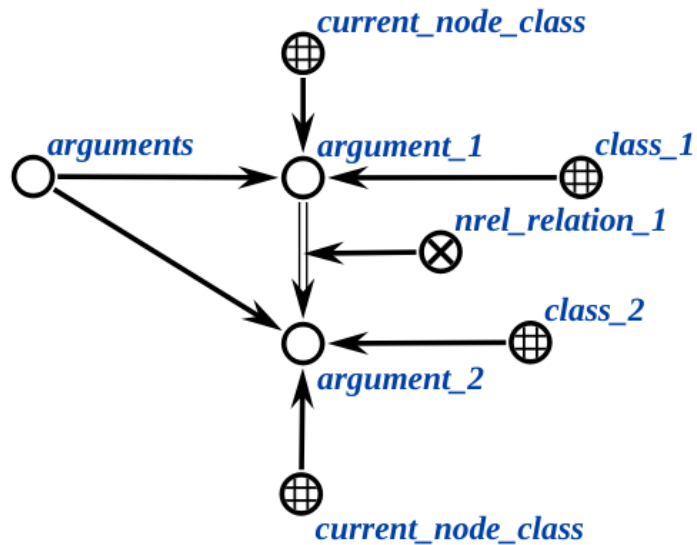
[Множество, элементы которого используются при применении правил. Каждый sc-узел этого множества подставляется как значение переменных атомарных логических формул (в том числе шаблона цели).]

⇒ примечание*:

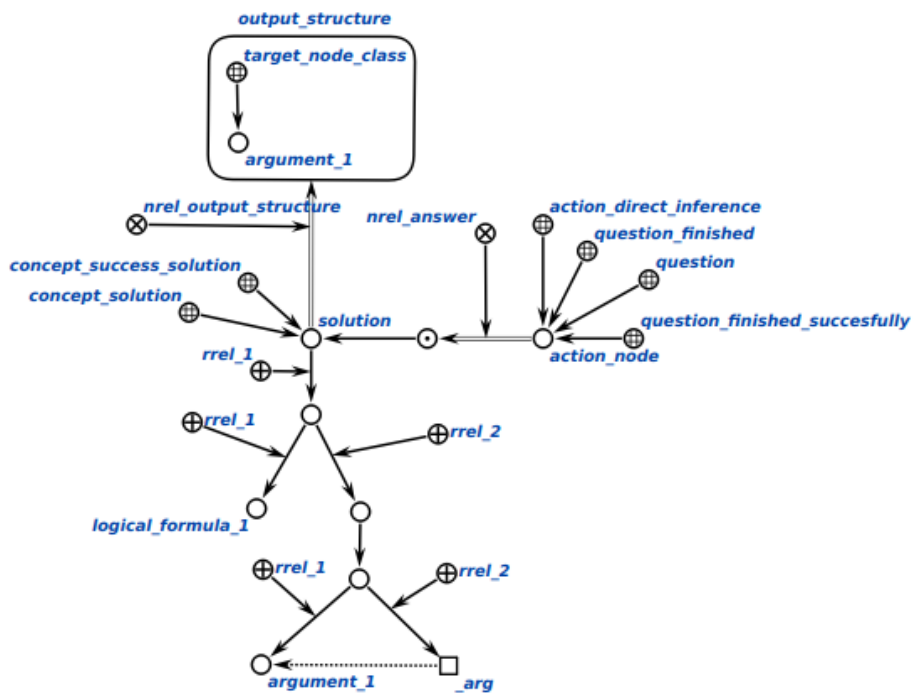
[Можно использовать структуру всей базы знаний системы, например, sc-узел **База знаний IMS**.]

⇒ описание примера*:

[



- `_input_structure`
 \Rightarrow *пояснение**:
 [Структура, в которой происходит поиск при проверке истинности атомарных логических формул.]
- \Rightarrow *ответ агента**:
ответ агента прямого логического вывода
- \Rightarrow *примечание**:
 [В результате выполнения агентом логического вывода действия, в sc-памяти формируется sc-структура, представляющая собой дерево решения. Это дерево состоит из последовательности пар, в которых первым элементом является применённое правило, а вторым - множество пар подстановок и переменных, соединённых временной дугой принадлежности, выходящей из переменного узла, при которых данное правило выполняется. Такое дерево может быть пустым в случае, если требуемую структуру не удалось сгенерировать в ходе логического вывода. При достижении цели вывода узел дерева решения позитивно принадлежит классу `concept_success_solution`, при недостижении – негативно. Корень этого дерева находится в связке под отношением `nrel_output_structure` со структурой, в которую добавляются сгенерированные в ходе логического вывода конструкции.]
- \Rightarrow *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 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 arguments, если множество валидно, заполнение ими списка аргументов;]
- [Проверка, достигнута ли уже цель в базе знаний с полученными аргументами;]

⇒ примечание*:

[Выполняется поиск по шаблону target template с параметрами arguments. Если шаблон найден, агент завершает работу, возвращает узел, принадлежащий *concept_success_solution*.]

- [Построение вектора очереди формул на основе множества формул. Цикл по всем правилам и пока не достигнута цель;]

⇒ циклические операции*:

- ⟨ • [Получение посылки логической формулы;]
- [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 формулы.]
- ⇒ примечание*:
[Смотрите пример ответа агента.]
- ⟩

- [Формирование дерева применённых формул.]

⇒ 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.]
- [Не реализована логика для входной структуры, поиск осуществляется по всей базе знаний.]

}

⇒

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.]

}