

Формальные языки

Егор Елисеев

С использованием любого инструмента реализовать парсер для языка P . Намеренно плохое описание конкретного синтаксиса ниже.

- Парсер должен быть встроен в консольное приложение, принимающее на вход путь к входному файлу.
- В результате работы парсера на некорректном входе пользователь должен получить цивилизованное сообщение об ошибке (пробросить исключение наружу — плохая идея).
- В результате работы парсера на корректном входе должно получаться абстрактное синтаксическое дерево, которое соответствует описанию абстрактного синтаксиса ниже. Его можно либо вывести в консоль, либо сохранить в файл — как удобнее.
- Конкретный синтаксис может быть вообще любым, учитывайте, что парсить этот язык вам же.
- Необходимо написать тесты.

Конкретный синтаксис языка P

Программа на языке P состоит из возможно пустого множества определений отношений, за которыми следует цель, отделенная символом $?$.

Отношение состоит из нескольких строк, его задающих.

Каждая строка состоит из головы и тела, разделенных символом $:-$, в конце стоит точка $(.)$.

Голова является атомом.

Атом состоит из идентификатора (название отношения или конструктора) и списка аргументов, разделенных пробельными символами. Сложные аргументы обрамляются в круглые скобки $((,))$.

Аргументом может быть либо переменная, либо атом.

Список аргументов атома может быть пуст.

Переменная всегда начинается с заглавной латинской буквы. Идентификатор — со строчной латинской буквы. Имена переменных и идентификаторов содержат только латинские буквы либо цифры $0 - 9$.

Тело является целью, также может быть пустым.

Цель — арифметика над атомами с операциями конъюнкция $(,)$ и дизъюнкция $(;)$, может содержать скобки $((,))$. Конъюнкция обладает большим приоритетом, чем дизъюнкция. Обе операции правоассоциативны.

Если тело пусто, символ $:-$ опускается.

Инdentация не является значащей.

Пример программы на языке P

```
eval St (var X) U :- elem X St U.  
eval St (conj X Y) true :- eval St X true, eval St Y true.  
eval St (conj X Y) false :- eval St X false; eval St Y false.  
eval St (disj X Y) true :- eval St X true; eval St Y true.  
eval St (disj X Y) false :- eval St X false, eval St Y false.  
eval St (not X) true :- eval St X false.  
eval St (not X) false :- eval St X true.
```

```
elem zero (cons H T) H.  
elem (succ N) (cons H T) V :- elem N T V.
```

```
? eval St (conj (disj X Y) (not (var Z))) true.
```

Еще пример программы на языке P

```
eval St (var X) U :- elem X St U.  
eval St (conj X Y) U :- eval St X V, eval St Y W, and V W U.  
eval St (disj X Y) U :- eval St X V, eval St Y W, or V W U.  
eval St (not X) U :- eval St X V, neg U V.
```

```
elem zero (cons H T) H.  
elem (succ N) (cons H T) V :- elem N T V.
```

```
nand false false true.  
nand false true true.  
nand true false true.  
nand true true false.
```

```
neg X R :- nand X X R.
```

```
or X Y R :- nand X X Xx, nand Y Y Yy, nand Xx Yy R.
```

```
and X Y R :- nand X Y Xy, nand Xy Xy R.
```

```
? eval St (conj (disj X Y) (not (var Z))) true.
```