**Тесты для парсинга**

Пусть {a1 = tg1, …, an = tgn} обозначает атрибутон, который определен в в состоянии s и имеет в этом состоянии атрибуты a1, …, an со значениями tg1, … tgn. Пусть (tg1, …, tgn) обозначает последовательность с элементами tg1, …, tgn в состоянии s.

|  |  |
| --- | --- |
| **Вход** | **Выход** |
| **Тесты для атомов** | |
| 3 | IntegerAtom(3) |
| true | BooleanAtom(true) |
| 4.2 | DoubleAtom(4.2) |
| -2.6 | DoubleAtom(-2.6) |
| 3.2e-5 | DoubleAtom(3.2e-5) |
| ′′abc′′ | StringAtom(′′abc′′) |
| abc | QNameAtom(′′abc′′, true)  Заметим, что  qname(′′abc′′) = QNameAtom(′′abc′′, false) ≠  QNameAtom(′′abc′′, true) |
| abc:2:ef | QNameAtom(′′abc:2:ef′′, true) |
| undef | UndefAtom() |
| **Переменные** | |
| $x | {QNameAtom(′′Variable′′, true) = BooleanAtom(true),  QNameAtom(′′name′′, true) = QNameAtom(′′x′′, true)  } |
| #x | {QNameAtom(′′AttributonVariable′′, true) = BooleanAtom(true),  QNameAtom(′′name′′, true) = QNameAtom(′′x′′, true)  } |
| **Выражение в скобках** | |
| (4.2) | DoubleAtom(4.2) |
| **Вызов функции** | |
| f(3, true, 4.2, ′′abc′′, abc) | {QNameAtom(′′FunctionCall′′, true) = BooleanAtom(true),  QNameAtom(′′function′′, true) = QNameAtom(′′f′′, true),  QNameAtom(′′arguments′′, true) =  (IntegerAtom(3), BooleanAtom(true), DoubleAtom(4.2),  StringAtom(′′abc′′), QNameAtom(′′abc′′, true)  )  } |
| **Синтаксические расширения** | |
| defun f($x, $y) {$x + $y;} | {QNameAtom(′′FunctionCall′′, true) = BooleanAtom(true),  QNameAtom(′′function′′, true) = QNameAtom(′′defun′′, true),  QNameAtom(′′arguments′′, true) = (QNameAtom(′′f′′, true),  {QNameAtom(′′Variable′′, true) = BooleanAtom(true),  QNameAtom(′′name′′, true) = QNameAtom(′′x′′, true)  },  {QNameAtom(′′Variable′′, true) = BooleanAtom(true),  QNameAtom(′′name′′, true) = QNameAtom(′′y′′, true)  },  {QNameAtom(′′FunctionCall′′, true) = BooleanAtom(true),  QNameAtom(′′function′′, true) = QNameAtom(′′add′′, true),  QNameAtom(′′arguments′′, true) =  ({QNameAtom(′′Variable′′, true) = BooleanAtom(true),  QNameAtom(′′name′′, true) = QNameAtom(′′x′′, true)  },  {QNameAtom(′′Variable′′, true) = BooleanAtom(true),  QNameAtom(′′name′′, true) = QNameAtom(′′y′′, true)  }  )  }  )  } |
| defun f($x, $y) {$x + $y} | как для  defun f($x, $y) {$x + $y;} |
| defun f($x, $y) {$x + $y; 3;} | {QNameAtom(′′FunctionCall′′, true) = BooleanAtom(true),  QNameAtom(′′function′′, true) = QNameAtom(′′defun′′, true),  QNameAtom(′′arguments′′, true) = (QNameAtom(′′f′′, true),  {QNameAtom(′′Variable′′, true) = BooleanAtom(true),  QNameAtom(′′name′′, true) = QNameAtom(′′x′′, true)  },  {QNameAtom(′′Variable′′, true) = BooleanAtom(true),  QNameAtom(′′name′′, true) = QNameAtom(′′y′′, true)  },  {QNameAtom(′′FunctionCall′′, true) = BooleanAtom(true),  QNameAtom(′′function′′, true) = QNameAtom(′′progn′′, true),  QNameAtom(′′arguments′′, true) =  ({QNameAtom(′′FunctionCall′′, true) = BooleanAtom(true),  QNameAtom(′′function′′, true) = QNameAtom(′′add′′, true),  QNameAtom(′′arguments′′, true) =  ({QNameAtom(′′Variable′′, true) = BooleanAtom(true),  QNameAtom(′′name′′, true) = QNameAtom(′′x′′, true)  },  {QNameAtom(′′Variable′′, true) = BooleanAtom(true),  QNameAtom(′′name′′, true) = QNameAtom(′′y′′, true)  }  )  },  IntegerAtom(3)  )  }  )  } |
| defun f($x, $y) {$x + $y; 3} | как для  defun f($x, $y) {$x + $y; 3;} |
| Остальные синтаксические расширения определяются аналогичным образом (также сводятся к вызову функции). Нужно составить тесты для всех синтаксических выражений. Для простоты проверки реализовать функцию toString и визуально или автоматически сверять результаты Входа и Выхода, к которому применена функция toString. | |
| **Предвычисляемые выражения** | |
| `3+5` | eval(  {QNameAtom(′′FunctionCall′′, true) = BooleanAtom(true),  QNameAtom(′′function′′, true) = QNameAtom(′′add′′, true),  QNameAtom(′′arguments′′, true) = (IntegerAtom(3), IntegerAtom(5))  }  ) =  IntegerAtom(8) |
| `{% type = pencil, color = green%}` | eval(  {QNameAtom(′′FunctionCall′′, true) = BooleanAtom(true),  QNameAtom(′′function′′, true) = QNameAtom(′′conz′′, true),  QNameAtom(′′arguments′′, true) =  (QNameAtom(type, true),  QNameAtom(pencil, true),  QNameAtom(color, true),  QNameAtom(green, true)  )  }  ) =  {QNameAtom(type, true) = QNameAtom(pencil, true),  QNameAtom(color, true) = QNameAtom(green, true)} |
| `(% 1 + 2, undef, true %)` | eval(  {QNameAtom(′′FunctionCall′′, true) = BooleanAtom(true),  QNameAtom(′′function′′, true) = QNameAtom(′′seqCreate′′, true),  QNameAtom(′′arguments′′, true) =  (IntegerAtom(3), UndefAtom(), BooleanAtom(true))  }  ) =  (IntegerAtom(3), UndefAtom(), BooleanAtom(true)) |
| Другие предвычисляемые выражения вычисляются аналогичным образом, т. е. парсер преобразует аргумент предвычисляемого выражения в его онтологическую модель и затем интерпретатор eval вычисляет эту онтологическую модель. | |