**Предопределенные функции языка ASL (в алфавитном порядке)**

Пусть состояние sα имеет вид (sβ, oα, oβ).

Будем говорить, что функция fα определяется как (vα1, vα2, n, Bγ) в состоянии sα, если sβ(οβ.special, fα) = vα1, sβ(οβ.varied, fα) = vα2, и (fα, n, Bγ) ∈ Bβ.

Пусть cδ обозначает вызов предопределенной функции.

# add

Функция add определяется как (undef, varied, 1, Bγ), где

* если n = 0, то ((sβ, oα, oβ), (seq(vγ1, …, vγn)), (sβ, upd(oα, value, 0), oβ)) ∈ Bγ;
* если n > 0, и vγ1, …, vγn ∈ Integer ∪ Double, и vγ – сумма чисел vγ1, …, vγn, то ((sβ, oα, oβ), (seq(vγ1, …, vγn)), (sβ, upd(oα, value, vγ), oβ)) ∈ Bγ;
* если n > 0, и существует 1 ≤ i ≤ n такое, что vγi ∉ Integer ∪ Double, то ((sβ, oα, oβ), (seq(vγ1, …, vγn)), (sβ, upd(oα, jtype, addJump, jvalue, cδ), oβ)) ∈ Bγ.

# and

Функция and имеет аргументы (x1, …, xn) и определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если n = 0, то возвратить значение true.

Если n > 0, то

Если u1, …, un ∈ {true}, то возвратить значение true.

Если u1, …, ui-1 ∈ {true}, и ui = false, то не вычислять xi+1, …, xn и возвратить значение false

Если u1, …, ui-1 ∈ {true}, и ui ∉ Boolean, то не вычислять xi+1, …, xn и возвратить джамп типа andJump.

# aref

Функция aref имеет аргументы (x0, x1, …, xn) и определяется следующим образом:

Пусть x0 возвращает значение u0.

Если n = 0, то возвратить значение u0.

Если n > 0, то

Если u0 ∉ Attributon, то возвратить джамп типов arefJump и Undef.

Если u0 ∈ Attributon, то.

Пусть выражения x1, …, xn возвращают значения u1, …, un.

Пусть vi = s(ui-1, ui).

Если v1, …, vi-1 ∈ Attributon, vi ∉ Attributon, и 1≤ i<n, то xi+1, …, xn не вычислять, и возвратить джамп типов arefJump и Undef.

Если v1, …, vn-1 ∈ Attributon, то возвратить значение vn.

# aset

Функция aset имеет аргументы (x0, x1, …, xn, y, z) и определяется следующим образом:

Пусть aref(x0, x1, …, xn) возвращает значение u.

Если u ∉ Attributon, то возвратить джамп типа asetJump.

Если u ∈ Attributon, то

Пусть у возвращает значение uy.

Если z возвратить джамп типа Undef, то удалить атрибут uy у атрибутона u.

Если z возвращает джамп w, который не имеет тип Undef, то возвратить джамп w.

Если z не возвращает джамп и возвращает значение uz, то присвоить значение uz атрибуту uy атрибутона u и возвратить значение uz.

# assert

Функция assert имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u = type, то возвратить текущее состояние.

Если u ∉ {true}, то возвратить джамп типа assertJump.

assertEq

Функция assertEq имеет аргументы (x, y) и определяется следующим образом:

Пусть x и y возвращают значения ux и uy.

Если ux = uy, то возвратить значение true.

Если ux ≠ uy, то возвратить джамп типа assertEqJump.

# catch

Функция catch имеет аргументы (x, y1, …, yn) и определяется следующим образом:

Если s(lc, jumping) = undef, то возвратить текущее состояние.

Если s(lc, jumping) ≠ undef, то

Присвоить атрибуту jumping локального контекста значение undef.

Пусть s(lc, jumpType) = u, и s(lc, jumpValue) = v.

Вычислить x.

Если x возвращает джамп w, то возвратить джамп w.

Если x не возвращает джамп, то

Пусть x возвращает значение ux.

Если u = ux, то вычислить progn(y1, …, yn).

Если u не имеет тип ux, то возвратить джамп u типа u со значением v.

# catchAll

Функция catchAll имеет аргументы (y1, …, yn) и определяется следующим образом:

Если s(lc, jumping) = undef, то возвратить текущее состояние.

Если s(lc, jumping) ≠ undef, то

Присвоить атрибуту jumping локального контекста значение undef.

Вычислить progn(y1, …, yn).

# locaref

Функция locaref имеет аргументы (x) и определяется следующим образом:

Вычислить aref(quote(oα.lcontext), x).

# locaset

Функция locaset имеет аргументы (x) и определяется следующим образом:

Вычислить aset(quote(oα.lcontext), x).

# coalesce

Функция coalesce имеет аргументы (x1, …, xn) и определяется следующим образом:

Если n = 0, то coalesce возвращает текущее состояние.

Если n > 0, то

Вычислить x1, …, xn. Перед вычислением каждого xi, где i > 1, атрибут jump удаляется из текущего состояния.

Если 0 < i ≤ n, x1, …, xi-1 возвращают джампы типа Undef, xi возвращает значение u, то возвратить значение u.

Если 0 < i ≤ n, x1, …, xi-1 возвращают джампы типа Undef, и xi возвращает джамп w, который не имеет тип Undef, то возвратить джамп w.

Если x1, …, xn возвращают джампы типа Undef, и xn возвращает джамп w, то возвратить джамп w.

# cond

Функция cond имеет аргументы (x1, y1, …, xn, yn, opt z), где n>1. Аргументы x1, …. xn называются условиями. Эта функция определяется следующим образом:

Пусть x1, …. xn возвращают значения u1, …, un.

Если u1, …, ui-1 ∈ {false}, ui = type, 1≤ i≤ n, то вычислить yi.

Если u1, …, ui-1 ∈ {false}, ui ∉ Boolean, и 1≤ i≤ n, то не вычислять xi+1, …, xn и возвратить джамп типа condJump.

Если u1, …, un ∈ {false}, и z присутствует, то вычислить z.

Если u1, …, un ∈ {false}, и z отсутствует, то возвратить текущее состояние.

# conz

Функция conz имеет аргументы (x1, y1, …, xn, yn) и определяется следующим образом:

Породить новый атрибутон u.

Вычислить последовательно x1, y1, …, xn, yn.

Пусть x1, y1, …, xn, yn возвращают значения ux1, uy1, …, uxn, uyn.

Если yi возвращает значение undef, то удалить атрибут uxi атрибутона u.

Если yi не возвращает значение undef, то присвоить атрибуту uxi атрибутона u значение uyi.

# copyDeep

Функция copyDeep имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u ∉ Attributon, то copyDeep возвращает значение u.

Если u ∈ Attributon, то породить новый атрибутон u1, определить те же самые атрибуты у u1 как у u, и рекурсивно применить copyDeep к значениям этих атрибутов.

Возвратить значение u1.

# copySeq

Функция copySeq имеет аргументы (x) и эквивалентна copyShallow(x).

# copyShallow

Функция copyShallow имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u ∉ Attributon, то возвратить значение u.

Если u ∈ Attributon, то породить новый атрибутон u1, определить те же самые атрибуты у u1 как у u, и присвоить им те же самые значения.

Возвратить значение u1.

# copyState

Функция copyState имеет аргументы () и определяется следующим образом:

Возвратить значение, которое является копией текущего состояния s.

Копия имеет те же самые атрибуты, что и s.

Значениями атрибутов variableValue, jump и previousJump копии являются копии значений соответствующих атрибутов состояния s.

Значения остальных атрибутов совпадают со значениями соответствующих атрибутов состояния s.

# defun

Функция defun определяется как (special, varied, 5, Bγ), где

* ((sβ, oα, oβ), (vγ1, vγ2, vγ3, vγ4, vγ′1, …, vγ′n), ((upd(sβ, (oβ.special, vγ3), vγ1, (oβ.varied, vγ3), vγ2, (oβ.arguments, vγ3), seq(vγ′1, …, vγ′n), (oβ.body, vγ3), vγ4), oα, oβ))) ∈ Bγ.

# div

Функция div имеет аргументы (x, y) и определяется следующим образом:

Пусть x, y возвращают значения ux, uy.

Если ux ∉ Integer ∪ Double, то не вычислять y и возвратить джамп типа Div.

Если ux ∈ Integer ∪ Double, то

Если uy ∉ Integer ∪ Double, то возвратить джамп типа divJump.

Если uy ∈ Integer ∪ Double, то

Если uy ≠ 0, то

Если ux, uy ∈ Integer, то возвратить значение, которое является остатком от деления ux на uy.

Если ux ∉ Integer, или uy ∉ Integer, то возвратить значение, которое является частным от деления ux на uy.

Если uy = 0, то возвратить джамп типа divJump.

# eq

Функция eq имеет аргументы (x1, …, xn), где n > 0, и определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если 1<i≤ n, и u1, …, ui-1 попарно равны, и ui ≠ ui-1, то не вычислять xi+1, …, xn и возвратить значение false.

Если u1, …, un попарно равны, то возвратить значение true.

# eqDeep

Функция eqDeep имеет аргументы (x1, …, xn), где n > 0. Пусть свойство P(u1, u2) означает, что либо u1 и u2 – не атрибутоны и u1 = u2, либо u1 и u2 –атрибутоны, которые имеют один и тот же набор атрибутов и для значений соответствующих атрибутов функция eqDeep возвращает значение true. Тогда eqDeep определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если 1<i≤ n, и P(u1, uj) для 1 < j < i, и P(u1, ui), то не вычислять xi+1, …, xn и возвратить значение false.

Если P(u1, ui) для 1 < i ≤ n, то возвратить значение true.

# eqSeq

Функция eqSeq имеет аргументы (x1, …, xn), где n > 0. Пусть свойство P(u1, u2) означает, что либо u1 и u2 – последовательности длины 0, либо u1 и u2 – последовательности длины n > 0, которые имеют те же самые элементы и в том же самом порядке. Тогда eqSeq определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если 1<i≤ n, и P(u1, uj) для 1 < j < i, и P(u1, ui), то не вычислять xi+1, …, xn и возвратить значение false.

Если P(u1, ui) для 1 < i ≤ n, то возвратить значение true.

# eqShallow

Функция eqShallow имеет аргументы (x1, …, xn), где n > 0. Пусть свойство P(u1, u2) означает, что либо u1 и u2 – не атрибутоны и u1 = u2, либо u1 и u2 –атрибутоны, которые имеют один и тот же набор атрибутов и значения соответствующих атрибутов равны. Тогда eqShallow определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если 1<i≤ n, и P(u1, uj) для 1 < j < i, и P(u1, ui), то не вычислять xi+1, …, xn и возвратить значение false.

Если P(u1, ui) для 1 < i ≤ n, то возвратить значение true.

# eval

Функция eval имеет аргументы (x, opt y, opt z). По умолчанию аргументы y и z возвращают значения, которые являются текущим локальным контекстом и текущим глобальным контекстом. Эта функция определяется следующим образом:

Пусть y, z возвращают значения uy, uz.

Если y присутствует, то заменить текущий локальный контекст на uy.

Если z присутствует, то заменить текущий глобальный контекст на uz.

Вычислить x.

Вернуться к текущим локальному и глобальному контекстам.

Установить атрибуты value и jump текущего контекста равными значениям этих атрибутов контекста uy.

# evalArg

Функция evalArg имеет аргументы (x). Эта функция определяется следующим образом:

Пусть lc – текущий локальный контекст, и lcpar – родитель lc.

Вычислить выражение eval(x, lcpar).

# fileToString

Функция fileToString имеет аргументы (x), где x – строка, определяющая полное имя файла.

Пока не нужно реализовывать.

# functions

Функция functions имеет аргументы () и определяется следующим образом:

Возвратить значение s(gc, functions).

# getState

Функция getState имеет аргументы () и определяется следующим образом:

Возвратить значение, которое является копией текущего состояния s.

Копия имеет те же самые атрибуты, что и s.

Значениями атрибутов variableValue, jump и previousJump копии являются копии значений соответствующих атрибутов состояния s.

Значения остальных атрибутов совпадают со значениями соответствующих атрибутов состояния s.

# globaref

Функция globaref имеет аргументы (x) и определяется следующим образом:

Если

Вычислить aref(quote(oβ.gcontext), x).

# globaset

Функция globaset имеет аргументы (x) и определяется следующим образом:

Вычислить aset(quote(oβ.gcontext), x).

# gt

Функция gt имеет аргументы (x1, …, xn). Пусть > - бинарная операция «больше» на числах или строках. Функция gt определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если n ≤ 1, то возвратить значение true.

Если n > 1, то

Если u1, …, ui ∈ Integer ∪ Double ∪ String, uj-1 > uj для 1 < j < i < n, и неверно, что ui-1 > ui, то не вычислять xi+1, …, xn и возвратить значение false.

Если u1, …, ui ∈ Integer ∪ Double ∪ String, uj-1 > uj для 1 < j < i < n, и ui ∉ Integer ∪ Double ∪ String, то не вычислять xi+1, …, xn и возвратить джамп типа gtJump.

Если u1, …, un ∈ Numeric ∪ String, и ui-1 > ui для 1 < i ≤ n, то возвратить значение true.

# gte

Функция gte имеет аргументы (x1, …, xn). Пусть ≥ - бинарная операция «больше или равно» на числах или строках. Функция gte определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если n ≤ 1, то возвратить значение true.

Если n > 1, то

Если u1, …, ui ∈ Integer ∪ Double ∪ String, uj-1 ≥ uj для 1 < j < i < n, и неверно, что ui-1 ≥ ui, то не вычислять xi+1, …, xn и возвратить значение false.

Если u1, …, ui ∈ Integer ∪ Double ∪ String, uj-1 ≥ uj для 1 < j < i < n, и ui ∉ Integer ∪ Double ∪ String, то не вычислять xi+1, …, xn и возвратить джамп типа gteJump.

Если u1, …, un ∈ Integer ∪ Double ∪ String, и ui-1 ≥ ui для 1 < i ≤ n, то возвратить значение true.

# interef

Функция interef имеет аргументы (x) и определяется следующим образом:

Вычислить aref(quote(oα.icontext), x).

# inteset

Функция inteset имеет аргументы (x) и определяется следующим образом:

Вычислить aset(quote(oα.icontext), x).

# isAtom

Функция isAtom имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u ∈ Atom, то возвратить значение true.

Если u ∉ Atom, то возвратить значение false.

# isBoolean

Функция isBoolean имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u ∈ Boolean, то возвратить значение true.

Если u ∉ Boolean, то возвратить значение false.

# isDef

Функция isDef имеет аргументы (x) и определяется следующим образом:

Если x не возвращает значение undef, то возвратить значение true.

Если x возвращает значение undef, то возвратить значение false.

# isDouble

Функция isDouble имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u ∈ Double, то возвратить значение true.

Если u ∉ Double, то возвратить значение false

# isInteger

Функция isInteger имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u ∈ Integer, то возвратить значение true.

Если u ∉ Integer, то возвратить значение false.

# isNumeric

Функция isNumeric имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u ∈ Numeric, то возвратить значение true.

Если u ∉ Numeric, то возвратить значение false.

# isQName

Функция isQname имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u ∈ QName, то возвратить значение true.

Если u ∉ QName, то возвратить значение false.

# isString

Функция isString имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u ∈ String, то возвратить значение true.

Если u ∉ String, то возвратить значение false.

# isAttributon

Функция isAttributon имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u ∈ Attributon, то возвратить значение true

Если u ∉ Attributon, то возвратить значение false.

# isUndef

Функция isUndef имеет аргументы (x) и определяется следующим образом:

Если x возвращает значение undef, то возвратить значение true.

Если x не возвращает значение undef, то возвратить значение false.

# jump

Функция jump для аргументов (x) определяется следующим образом:

$jumpType = x; $jumping = type;.

Функция jump для аргументов (x, y) определяется следующим образом:

$jumpType = x; $jumpValue = y; $jumping = type;.

# load

Функция load имеет аргументы (x), где x – имя файла, и определяется следующим образом:

Пусть x возвращают значение u.

Если u ∉ String, то возвратить джамп типа loadJump.

Если файл с именем u не загружается, то возвратить джамп типа loadJump.

Если содержимое файла с именем u не является последовательностью ASL-выражений, то возвратить джамп типа loadJump.

Если e1, …, en – содержимое файла с именем u, то вычислить progn(e1, …, en).

# loadData

Функция loadData имеет аргументы (x), где x – имя файла, и определяется следующим образом:

Пусть x возвращают значение u

Если u ∉ String, то возвратить джамп типа loadDataJump.

Если файл с именем u не загружается, то возвратить джамп типа

# loadDataJump.

Если содержимое файла с именем u не является последовательностью ASL-выражений, то возвратить джамп типа loadDataJump.

Если e1, …, en – содержимое файла с именем u, то преобразовать e1, …, en в их онтологические модели m1, …, mn и возвратить значение mn.

# lt

Функция lt имеет аргументы (x1, …, xn). Пусть < - бинарная операция «меньше» на числах или строках. Функция lt определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если n ≤ 1, то возвратить значение true.

Если n > 1, то

Если u1, …, ui ∈ Integer ∪ Double ∪ String, uj-1 < uj для 1 < j < i < n, и неверно, что ui-1 < ui, то не вычислять xi+1, …, xn и возвратить значение false.

Если u1, …, ui ∈ Integer ∪ Double ∪ String, uj-1 < uj для 1 < j < i < n, и ui ∉ Integer ∪ Double ∪ String, то не вычислять xi+1, …, xn и возвратить джамп типа ltJump.

Если u1, …, un ∈ Integer ∪ Double ∪ String, и ui-1 < ui для 1 < i ≤ n, то возвратить значение true.

# lte

Функция lte имеет аргументы (x1, …, xn). Пусть ≤ - бинарная операция «меньше или равно» на числах или строках. Функция lte определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если n ≤ 1, то возвратить значение true.

Если n > 1, то

Если u1, …, ui ∈ Integer ∪ Double ∪ String, uj-1 ≤ uj для 1 < j < i < n, и неверно, что ui-1 ≤ ui, то не вычислять xi+1, …, xn и возвратить значение false.

Если u1, …, ui ∈ Integer ∪ Double ∪ String, uj-1 ≤ uj для 1 < j < i < n, и ui ∉ Integer ∪ Double ∪ String, то не вычислять xi+1, …, xn и возвратить джамп типа gteJump.

Если u1, …, un ∈ Integer ∪ Double ∪ String, и ui-1 ≤ ui для 1 < i ≤ n, то возвратить значение true.

# mapFirst

Функция mapFirst имеет аргументы (x, y) и определяется следующим образом:

Пусть x, y возвращают значения ux, uy.

Если ux – не функция, или ux не допускает один аргумент, то возвратить джамп типа mapFirstJump.

Если ux – функция, и ux допускает один аргумент, то

Если uy – пустая последовательность, то возвратить значение uy.

Если uy – непустая последовательность, то породить новую последовательность u, где элементами являются результаты применения функции ux слева направо к элементам последовательности uy, и возвратить значение u.

# mapKeyValues

Функция mapKeyValues имеет аргументы (x, y) и определяется следующим образом:

Пусть x, y возвращают значения ux, uy.

Если ux ∉ Function, или ux не допускает два аргумента, то возвратить джамп типа mapKeyValuesJump.

Если ux ∈ Function, и ux допускает два аргумента, то

Если uy ∉ Attributon, то возвратить значение uy.

Если uy ∈ Attributon, то породить новый атрибутон u с теми же атрибутами, что и uy, значениями которых являются результаты применения функции ux в произвольном порядке к парам (имя атрибута, значение атрибута), и возвратить значение u.

# mapSortedKeyValues

Функция mapSortedKeyValues имеет аргументы (x, y) и определяется следующим образом:

Пусть x, y возвращают значения ux, uy.

Если ux ∉ Function, или ux не допускает два аргумента, то возвратить джамп типа mapSortedKeyValuesJump.

Если ux ∈ Function, и ux допускает два аргумента, то

Если uy ∉ Attributon, то возвратить значение uy.

Если uy ∈ Attributon, то породить новый атрибутон u с теми же атрибутами, что и uy, значениями которых являются результаты применения функции ux слева направо (в возрастающем порядке имен атрибутов) к парам (имя атрибута, значение атрибута), и возвратить значение u.

# mod

Функция mod имеет аргументы (x, y) и определяется следующим образом:

Пусть x, y возвращают значения ux, uy.

Если ux ∉ Integer, то не вычислять y и возвратить джамп типа modJump.

Если ux ∈ Integer, то

Если uy ∉ Integer, то возвратить джамп типа modJump.

Если uy ∈ Integer, то

Если uy ≠ 0, то возвратить значение, которое является остатком от деления ux на uy.

Если uy = 0, то возвратить джамп типа modJump.

# mul

Функция mul имеет аргументы (x1, …, xn) и определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если n = 0, то возвратить значение 1.

Если n > 0, то

Если u1, …, un ∈ Integer ∪ Double, то возвратить значение, которое является произведением чисел u1, …, un.

Если u1, …, ui-1 ∈ Integer ∪ Double, и ui ∉ Integer ∪ Double, то не вычислять xi+1, …, xn и возвратить джамп типа mulJump.

# neq

Функция neq имеет аргументы (x, y) и определяется следующим образом:

Пусть x, y возвращают значения ux, uy.

Если ux ≠ uy, то возвратить значение true.

Если ux = uy, то возвратить значение false.

# nop

Функция nop имеет аргументы (x1, …, xn).

Пока не нужно реализовывать.

# not

Функция not имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u = type, то возвратить значение false.

Если u = false, то возвратить значение true.

Если u ∉ Boolean, то возвратить джамп типа notJump.

# or

Функция or имеет аргументы (x1, …, xn) и определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если n = 0, то возвратить значение false.

Если n > 0, то

Если u1, …, un ∈ {false}, то возвратить значение false.

Если u1, …, ui-1 ∈ {false}, и ui = type, то не вычислять xi+1, …, xn и возвратить значение true.

Если u1, …, ui-1 ∈ {false}, и ui ∉ Boolean, то не вычислять xi+1, …, xn и возвратить джамп типа orJump.

# interef

Функция interef имеет аргументы (x) и определяется следующим образом:

Вычислить aref(quote(lcp), x).

# interefq

Функция interefq имеет аргументы (x) и определяется следующим образом:

Вычислить aref(quote(lcp), quote(x)).

# inteset

Функция plocaset имеет аргументы (x) и определяется следующим образом:

Вычислить aset(quote(lcp), x).

# print

Функция print имеет аргументы (x1, …, xn) и определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если v1, …, vn – строковые представления элементов u1, …, un, то выдать на печать строку v1, …, vn.

# println

Функция println имеет аргументы (x1, …, xn) и определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если v1, …, vn – строковые представления элементов u1, …, un, то выдать на печать строку v1, …, vn и выполнить перевод строки.

# progn

Функция progn имеет аргументы (x1, …, xn) и определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если i > 0, xi-1 возвращает джамп u, xi – не catch, то возвратить джамп u.

Если xn не возвращает джамп, то возвратить значение un.

Если xn возвращает джамп u, то возвратить джамп u.

# qname

Функция qname имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u ∉ QName ∪ String, то возвратить джамп типа qnameJump.

Если u ∈ QName, то возвратить значение u.

Если u ∈ String, и для u порождалось квалифицированное имя v, то возвратить значение v.

Если u ∈ String, и для u не порождалось квалифицированное имя, то породить новое квалифицированное имя для u и возвратить значение, которое является этим квалифицированным именем.

# quote

Функция quote имеет аргументы (quote x) и определяется следующим образом:

Возвратить значение x.

# read

Функция read имеет аргументы (x1, …, xn) и определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если i = 0, то возвратить текущее состояние.

Если i > 0, то

u1, …, ui-1 ∈ String, и ui ∉ String, то возвратить джамп типа readJump.

Если u1, …, un ∈ String, то преобразовать u1, …, un в онтологические модели m1, …, mn и возвратить значение mn.

# return

Функция return имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Вычислить jump(conz(returnJump, true, value, quote(u))).

# seqAppend

Функция seqAppend имеет аргументы (x, x1, …, xn) и определяется следующим образом:

Пусть x1, …, xn, x возвращают значения u1, …, un, u.

Если w1 – первый элемент последовательности (u1, …, un, u), и w1 возвращает джамп w2, то не вычислять остальные элементы этой последовательности и возвратить джамп w2.

Если u – не последовательность, то возвратить джамп типа seqAppendJump.

В противном случае, если u = {seqLen = m, start = i, i+0 = v1, …, i+m-1 = vm, w}, где w – остальные пары атрибутов, то заменить содержимое u на {seqLen = m+n, start = i, i+0 = v1, …, i+m-1 = vm, i+m = u1, …, i+m+n-1 = un, w} и возвратить значение u.

# seqApply

Функция seqApply имеет аргументы (x, y) и определяется следующим образом:

Пусть x, y возвращают значения ux, uy.

Если ux – не функция, то возвратить джамп типа seqApplyJump.

Если ux – функция, и uy – последовательность длины n, то

Если ux допускает последовательность аргументов длины n, то применить ux к последовательности аргументов, соответствующих элементам последовательности uy.

Если ux не допускает последовательность аргументов длины n, то возвратить джамп типа seqApplyJump.

# seqCons

Функция seqCons имеет аргументы (x, y) и определяется следующим образом:

Пусть x, y возвращают значения ux, uy.

Если x возвращает джамп u, то не вычислять y и возвратить джамп u.

Если y возвращает джамп u, то возвратить джамп u.

Если uy – не последовательность, то возвратить джамп типа sequenceJump.

Если uy = {seqLen = n, start = i, i+0 = u1, …, i+n-1 = un, w}, то заменить содержимое атрибутона uy на {seqLen = n+1, start = i, i+0 = ux, i+1 = u1, …, i+n = un, w} и возвратить значение uy.

# seqCopy

Функция seqCopy имеет аргументы (x) и эквивалентна copyShallow(x).

# seqCreate

Функция seqCreate имеет аргументы (x1, …, xn) и определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Породить новую последовательность w из элементов u1, …, un и возвратить значение w.

# seqFirst

Функция seqFirst имеет аргументы (x) и определяется следующим образом:

Если x – не последовательность, то возвратить джамп типа seqFirstJump.

В противном случае,

Если aref(x, start) возвращает значение u ∉ Integer ∪ Undef, то возвратить джамп типа seqFirstJump.

Если aref(x, start) возвращает undef, то вычислить aref(x, 1).

В противном случае, вычислить aref(x, aref(x, start)).

# seqLength

Функция seqLength имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u – последовательность, то возвратить значение, которое является длиной последовательности u.

В противном случае, возвратить джамп типа seqLengthJump.

# seqOne

Функция seqOne имеет аргументы (x1, …, xn) и определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Если x1, …, xi-1 возвращают значения, и xi возвращает джамп w, то возвратить джамп w.

В противном случае, вычислить conz(seqLen, n, seqStart, 1, 1, quote(u1), …, n, quote(un)).

# seqPrepend

Функция seqPrepend имеет аргументы (x1, …, xn, x) и определяется следующим образом:

Пусть x1, …, xn, x возвращают значения u1, …, un, u.

Если w1 – первый элемент последовательности (u1, …, un, u), и w1 возвращает джамп w2, то не вычислять остальные элементы этой последовательности и возвратить джамп w2.

Если u – не последовательность, то возвратить джамп типа seqPrependJump.

В противном случае, если u = {seqLen = m, start = i, i+0 = v1, …, i+m-1 = vm, w}, где w – остальные пары атрибутов, то заменить содержимое u на {seqLen = m+n, start = i, i+0 = u1, …, i+n-1 = un, i+n = v1, …, i+n+m-1 = vm, w} и возвратить значение u.

# seqRest

Функция seqRest имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u – не последовательность, то возвратить джамп типа seqRestJump.

Если u – пустая последовательность, то возвратить джамп типа seqRestJump.

В противном случае, если u = {seqLen = n, start = i, i+0 = u1, …, i+n-1 = un, w}, то заменить содержимое u на {seqLen = n-1, start = i, i+0 = u2, …, i+n-2 = un, w} и возвратить u.

# seqReverse

Функция seqReverse имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если x возвращает джамп w, то возвратить джамп w.

Если u – не последовательность, то возвратить джамп типа seqReverseJump.

В противном случае, если u = {seqLen= n, start = i, i+0 = u1, …, i+n-1 = un, w}, то заменить содержимое u на {seqLen= n, start = i, i+0 = un, …, i+n-1 = u1, w} и возвратить значение u.

# setq

Функция setq имеет аргументы (x, y) и определяется следующим образом:

Пусть y возвращает значение u.

Если s(x, Variable) = type, то

Присвоить атрибуту x атрибутона s(lc, variables) значение u.

Возвратить значение u.

Если s(x, AttributonVariable) = type, то

Если u ∉ Atom, то

Присвоить атрибуту x атрибутона s(lc, attributonVariables) значение u.

Возвратить значение u.

В противном случае, возвратить джамп типа setqJump.

В противном случае, не вычислять y и возвратить джамп типа setqJump.

# stop

Функция stop имеет аргументы () и определяется следующим образом:

Удалить текущий локальный контекст из последовательности локальных контекстов.

# strConcat

Функция strConcat имеет аргументы (x1, …, xn) и определяется следующим образом:

Пусть x1, …, xn возвращают значения u1, …, un.

Пусть w1, …, wn – результаты конвертации u1, …, un в строки.

Если n = 0, то возвратить значение undef.

Если n > 0, то

Если 0 < i ≤ n, u1, …, ui-1 могут быть конвертированы в строки, элемент ui не может быть конвертирован в строку, то возвратить джамп типа strConcatJump.

Если u1, …, un могут быть конвертированы в строки, то породить новую строку w, которая является конкатенацией строк w1, …, wn и возвратить значение w.

# strToFile

Функция strToFile имеет аргументы (…).

Пока не нужно реализовывать.

# sub

Функция sub имеет аргументы (x, y) и определяется следующим образом:

Пусть x, y возвращают значения ux, uy.

Если ux, uy ∈ Integer ∪ Double, то возвратить значение, которое является разностью w чисел ux и uy.

Если ux, uy ∈ Integer, то w ∈ Integer.

Если ux ∈ Double, или uy ∈ Double, то w ∈ Double.

Если ux ∉ Integer ∪ Double, то не вычислять y и возвратить джамп типа subJump.

Если ux ∈ Integer ∪ Double, и uy ∉ Integer ∪ Double, то возвратить джамп типа subJump.

# toBoolean

Функция toBoolean имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Пусть w – результат приведения u к Boolean.

Если u ∈ Boolean, то w = u.

Если u ∈ Integer ∪ Double, то

Если u = 0, то w = false.

Если u ≠ 0, то w = type.

Если u ∉ Boolean ∪ Integer ∪ Double, то u не приводится к Boolean.

Если u приводится к Boolean, то возвратить значение w.

Если u не приводится к Boolean, то возвратить джамп типа toBooleanJump.

# toDouble

Функция toDouble имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Пусть w – результат приведения u к Double.

Если u ∈ Double, то w = u.

Если u ∈ Integer, то u приводится к Double без потери точности.

Если u ∉ Integer ∪ Double, то u не приводится к Boolean.

Если u приводится к Double, то возвратить значение w.

Если u не приводится к Double, то возвратить джамп типа toDoubleJump.

# toInteger

Функция toInteger имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Пусть w – результат приведения u к Integer.

Если u ∈ Integer, то w = u.

Если u ∈ Double, то

Если u приводится к Integer без округления, то w – результат этого приведения.

Если u не приводится к Integer без округления, то u не приводится к Integer.

Если u ∉ Integer ∪ Double, то u не приводится к Integer.

Если u приводится к Integer, то возвратить значение w.

Если u не приводится к Integer, то возвратить джамп типа toIntegerJump.

# toString

Функция toString имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Пусть w – результат приведения u к String.

Если u приводится к String, то возвратить значение w.

Если u не приводится к String, то возвратить джамп типа toStringJump.

# uminus

Функция uminus имеет аргументы (x) и определяется следующим образом:

Пусть x возвращает значение u.

Если u ∈ Integer ∪ Double, то возвратить значение, которое является отрицанием числа u.

Если u ∉ Integer ∪ Double, то возвратить джамп типа uminusJump.

# variants

Функция variants имеет аргументы (x1, …, xn) и определяется следующим образом:

Пусть lcc1, …, lccn – копии текущего локального контекста.

Пусть x1, …, xn вычисляются в локальных контекстах lcc1, …, lccn в порядке слева направо и возвращают последовательности локальных контекстов lcs1, …, lcsn.

Возвратить состояние, которое получается из текущего состояния заменой последовательности локальных контекстов на конкатенацию последовательностей lcs1, …, lcsn.

# variantsForKeyValues

Функция variantsForKeyValues имеет аргументы (x, y, z) и определяется следующим образом:

Пусть z возвращает значение u.

Пусть {a1, …, an} – множество атрибутов элемента u.

Пусть lc1, …, lcn – копии текущего контекста lc, расширенные переменными x и y со значениями s(lc, a1), …, s(lc, an).

Если x, y – переменные, то

Если n = 0, то возвратить lc.

Если n > 0, то возвратить состояние, полученное из текущего состояния заменой lc на последовательность из локальных контекстов lc1, …, lcn, входящих в нее в произвольном порядке.

Если x или y – не переменная, то не вычислять z и возвратить джамп типа variantsForKeyValuesJump.

# while

Функция while имеет аргументы (x, y) и определяется следующим образом:

1: Вычислить x.

Пусть x возвращает значение ux.

Если ux = type, то вычислить u и перейти к шагу 1.

Если ux = false, то закончить вычисление.

Если ux ∉ Boolean, то возвратить джамп типа whileJump.