**Спецификация ядра языка ASL**

ASL (Attribute Systems Language) – императивный язык описания атрибутных систем.

# Атрибутные системы типа ASL

Класс атрибутных систем, специфицируемых в языке ASL (далее атрибутные системы типа ASL) расширяет атрибутные системы типа s+a+tov+tj+c+oob за счет описания переходов с помощью функций.

Пусть seq(vγ1, …, vγn) обозначает атрибутон, являющийся последовательностью с элементами vγ1, …, vγn.

Пусть upd(f, x, y) обозначает функцию f′, которая совпадает с функцией f для всех аргументов кроме возможно аргумента x, и f′(x) = y.

Пусть upd(f, <x>, y) – сокращение для upd(f, x, y).

Пусть upd(f, <x1, …, xn>, y), где n > 1, – сокращение для upd(f, x1, upd(f(x1), <x2, …, xn>, y).

Пусть upd(f, x1, y1, x2, y2, …, xn, yn) – сокращение для upd(upd(f, x1, y1), x2, y2, …, xn, yn).

Ниже атрибутоны также рассматриваются как функции.

Атрибутная система типа ASL – это кортеж (Aα, Aβ, Cα, Cβ, Bβ), где

* (Aα, Aβ, Cα, Cβ, Oδ, Bγ) – атрибутная система типа oob;
* String ∪ Integer ∪ Double ∪ SQName ⊆ Aα;
* множество строк String определяется следующим образом: если u - Unicode-строка, то ′′u′′ ∈ String. Как обычно, в строке u могут экранироваться символы ′′, \ , aα, n и t, например, \t;
* элементами множества Integer являются целые числа в стандартной нотации, например -2, 6, 0 и т. п.;
* элементами множества Double являются числа с фиксированной точкой в стандартной нотации, например, -2.6, 3.2e-5 и т. п.;
* множество простых квалифицированных имен SQName строится следующим образом: если u1, u2, …, un – строки, которые состоят из цифр 0-9, латинских букв (a-z, A-Z) и прочерка (\_) и начинаются не с цифры, то u1:u2: … :un ∈ SQName. В частности, u1 ∈ SQName;
* value, tvalue, jvalue, jtype, icontext, lcontext ∈ Cα;
* gcontext, actual, arguments, body, special, varied ∈ Cβ;
* любой элемент из Vα может рассматриваться как функция. Пусть Fα = Vα – множество функций;
* arguments ∈ Cα – компонент спецификатора определяемой функции;
* oα.arguments – спецификатор определяемой функции. Функция fα называется определяемой в состоянии (sβ, oα), если sβ(oα.arguments, fα) ∈ Aδ(sβ). Элементы последовательности sβ(oα.arguments, fα) называются аргументами функции fα, а длина этой последовательности – базовой местностью функции fα;
* body ∈ Cα – компонент спецификатора тела определяемой функции;
* oα.body – спецификатор тела определяемой функции. Значение sβ(oα.arguments, fα) называется телом функции fα в состоянии (sβ, oα);
* special ∈ Cα – компонент спецификатора тела специальной функции;
* oα.special – спецификатор специальной функции. Функция fα является специальной в состоянии (sβ, oα), если sβ(oα.special, fα) = special;
* varied ∈ Cα – компонент спецификатора функции с переменным числом аргументов;
* oα.varied – спецификатор функции с переменным числом аргументов. Функция fα является функцией с переменным числом аргументов в состоянии (sβ, oα), если sβ(oα.varied, fα) = varied;
* Bβ = ((fα1, n1, Bγ1), …, (fαm, nm, Bγm)) – базис предопределенных функций, где fαi – предопределенная функция, ni ∈ Nat – базовая местность функции fαi, Nat – множество натуральных чисел с нулем, Bγi ⊆ ((Sβ × Oα × Oβ) × Vαn\_i × (Sβ × Oγ × Oβ)) – отношение ветвления функции fαi;
* catch – предопределенная функция базовой местности 1;
* FCall(sβ) ⊆ Aβ – множество вызовов функций в состоянии sβ. Атрибутон aβ является вызовом функции в состоянии sβ, если sβ(aβ, functionCall) = concept, sβ(aβ, function) = fα, fα – определяемая функция в состоянии sβ, или fα – предопределенная функция, и sβ(aβ, arguments) – последовательность длины arity(fα) в случае, если fα – функция с постоянным числом аргументов в состоянии sβ и длины arity(fα) или большей, если fα – функция с переменным числом аргументов в состоянии sβ;
* Отношения ветвления Bα и перехода Tα определяются как минимальные отношения, удовлетворяющие следующим правилам (правила перечислены в порядке приоритета):
  + если oα.jvalue ≠ vβ, или oα.jtype ≠ vβ, и неверно, что iα ∈ Aβ, iα ∈ FCall(sβ), и sβ(iα, function) = catch, то ((sβ, oα, oβ), iα, (sβ, (oα), oβ)) ∈ Bα. Это правило просачивания джампов: джампы ловятся только функцией catch;
  + если iα ∉ Aβ, или iα ∈ Aβ, и sβ(iα, functionCall) ≠ concept, то ((sβ, oα, oβ), iα, (sβ, (upd(oα, value, iα)), oβ)) ∈ Bα. Это правило, когда входом является не вызов функции: состояние при переходе не меняется, а в выходе меняется только значение компонента value на вход;
  + если sβ(iα, functionCall) = concept, и iα ∉ FCall(sβ), то ((sβ, oα, oβ), iα, (sβ, (upd(oα, jtype, functionCallJump, jvalue, iα), oβ)) ∈ Bα. Это правило, когда входом является некорректный вызов функции: состояние при переходе не меняется, а в выходе меняется только значение компонентов jtype и jvalue на functionCallJump и вход, таким образом порождается джамп;
  + если iα ∈ FCall(sβ), sβ(iα, function) = fα, sβ(iα, arguments) = seq(vγ1, …, vγn), и (fα, m, Bγ) ∈ Bβ, то
    - sβ(oβ.special, fα) = special, sβ(oβ.varied, fα) ≠ varied, m = n, и ((sβ, oα, oβ), (vγ1, …, vγn), (sβ′, os, oβ)) ∈ Bγ, то ((sβ, oα, oβ), iα, (sβ′, os, oβ)) ∈ Bα;
    - если sβ(oβ.special, fα) ≠ special, sβ(oβ.varied, fα) ≠ varied, m = n, (sβ0, oγ0, oβ) = (sβ, (oα) , oβ), ((sβ0, oγ0, oβ), vγ1, (sβ1, oγ1, oβ)) ∈ Tα, …, ((sβn-1, oγn-1, oβ), vγn, (sβn, oγn, oβ)) ∈ Tα, oαi ∈ oγi, oαi.jvalue = vβ и oαi.jtype = vβ для 1 ≤ i ≤ n, и ((sβn, oαn, oβ), (oα1.value, …, oαn.value), (sβ′, oγ, oβ)) ∈ Bγ, то ((sβ, oα, oβ), iα, (sβ′, oγ, oβ)) ∈ Bα;
    - если sβ(oβ.special, fα) ≠ special, sβ(oβ.varied, fα) ≠ varied, m = n, (sβ0, oγ0, oβ) = (sβ, (oα) , oβ), ((sβ0, oγ0, oβ), vγ1, (sβ1, oγ1, oβ)) ∈ Tα, …, ((sβk-1, oγk-1, oβ), vγk, (sβk, oγk, oβ)) ∈ Tα, oαi ∈ oγi, oαi.jvalue = vβ и oαi.jtype = vβ для 1 ≤ i ≤ k-1, k ≤ n, oαk ∈ oγk, и неверно, что (oαk.jvalue = vβ и oαk.jtype = vβ), то ((sβ, oα, oβ), iα, (sβk, oγk, oβ)) ∈ Bα;
    - если sβ(oβ.special, fα) = special, sβ(oβ.varied, fα) = varied, n ≥ m - 1, и ((sβ, oα, oβ), (vγ1, …, vγm-1, seq(vγm, …, vγn)), (sβ′, oγ, oβ)) ∈ Bγ, то ((sβ, oα, oβ), iα, (sβ′, oγ, oβ)) ∈ Bα;
    - если sβ(oβ.special, fα) ≠ special, sβ(oβ.varied, fα) = varied, n ≥ m - 1, (sβ0, oγ0, oβ) = (sβ, (oα) , oβ), ((sβ0, oγ0, oβ), vγ1, (sβ1, oγ1, oβ)) ∈ Tα, …, ((sβn-1, oγn-1, oβ), vγn, (sβn, oγn, oβ)) ∈ Tα, oαi ∈ oγi, oαi.jvalue = vβ и oαi.jtype = vβ для 1 ≤ i ≤ n, и ((sβn, oαn, oβ), (oα1.value, …, oαm-1.value, seq(oαm.value, …, oαn.value)), (sβ′, oγ, oβ)) ∈ Bγ, то ((sβ, oα, oβ), iα, (sβ′, oγ, oβ)) ∈ Bα;
    - если sβ(oβ.special, fα) ≠ special, sβ(oβ.varied, fα) = varied, n ≥ m - 1, (sβ0, oγ0, oβ) = (sβ, (oα) , oβ), ((sβ0, oγ0, oβ), vγ1, (sβ1, oγ1, oβ)) ∈ Tα, …, ((sβk-1, oγk-1, oβ), vγk, (sβk, oγk, oβ)) ∈ Tα, oαi ∈ oγi, oαi.jvalue = vβ и oαi.jtype = vβ для 1 ≤ i ≤ k-1, k ≤ n, oαk ∈ oγk, и неверно, что (oαk.jvalue = vβ и oαk.jtype = vβ), то ((sβ, oα, oβ), iα, (sβk, oγk, oβ)) ∈ Bα;
  + если iα ∈ FCall(sβ), sβ(iα, function) = fα, sβ(iα, arguments) = seq(vγ1, …, vγn), и sβ(oα.arguments, fα) = seq(vγ′1, …, vγ′m), то
    - sβ(oβ.special, fα) = special, sβ(oβ.varied, fα) ≠ varied, m = n, sβ(oβ.actual, aβ) ≠ actual, и ((upd(sβ, (oβ.actual, aβ), actual, (aβ, vγ′1), vγ1, …, (aβ, vγ′n), vγn), upd(oα, lcontext, aβ), oβ)), sβ(oα.body, fα), (sβ′, oγ, oβ)) ∈ Bα, то ((sβ, oα, oβ), iα, (sβ′, oγ, oβ)) ∈ Bα;
    - если sβ(oβ.special, fα) ≠ special, sβ(oβ.varied, fα) ≠ varied, m = n, (sβ0, oγ0, oβ) = (sβ, (oα), oβ), ((sβ0, oγ0, oβ), vγ1, (sβ1, oγ1, oβ)) ∈ Tα, …, ((sβn-1, oγn-1, oβ), vγn, (sβn, oγn, oβ)) ∈ Tα, oαi ∈ oγi, oαi.jvalue = vβ и oαi.jtype = vβ для 1 ≤ i ≤ n, sβn(oβ.actual, aβ) ≠ actual, и ((upd(sβn, (oβ.actual, aβ), actual, (aβ, vγ′1), oα1.value, …, (aβ, vγ′n), oαn.value), upd(oα, lcontext, aβ)), sβ(oα.body, fα), (sβ′, oγ, oβ)) ∈ Bα, то ((sβ, oα, oβ), iα, (sβ′, oγ, oβ)) ∈ Bα;
    - если sβ(oβ.special, fα) ≠ special, sβ(oβ.varied, fα) ≠ varied, m = n, (sβ0, oγ0, oβ) = (sβ, (oα), oβ), ((sβ0, oγ0, oβ), vγ1, (sβ1, oγ1, oβ)) ∈ Tα, …, ((sβk-1, oγk-1, oβ), vγk, (sβk, oγk, oβ)) ∈ Tα, oαi ∈ oγi, oαi.jvalue = vβ и oαi.jtype = vβ для 1 ≤ i ≤ k-1, k ≤ n, oαk ∈ oγk, и неверно, что (oαk.jvalue = vβ и oαk.jtype = vβ), то ((sβ, oα, oβ), iα, (sβk, oγk, oβ)) ∈ Bα;
    - если sβ(oβ.special, fα) = special, sβ(oβ.varied, fα) = varied, n ≥ m - 1, sβ(oβ.actual, aβ) ≠ actual, и ((upd(sβ, (oβ.actual, aβ), actual, (aβ, vγ′1), vγ1, …, (aβ, vγ′m-1), vγm-1, (aβ, vγ′m), seq(vγm, …, vγn)), upd(oα, lcontext, aβ), oβ)), sβ(oα.body, fα), (sβ′, oγ, oβ)) ∈ Bα, то ((sβ, oα, oβ), iα, (sβ′, oγ, oβ)) ∈ Bα;
    - если sβ(oβ.special, fα) ≠ special, sβ(oβ.varied, fα) = varied, n ≥ m - 1, (sβ0, oγ0, oβ) = (sβ, (oα), oβ), ((sβ0, oγ0, oβ), vγ1, (sβ1, oγ1, oβ)) ∈ Tα, …, ((sβn-1, oγn-1, oβ), vγn, (sβn, oγn, oβ)) ∈ Tα, oαi ∈ oγi, oαi.jvalue = vβ и oαi.jtype = vβ для 1 ≤ i ≤ n, sβn(oβ.actual, aβ) ≠ actual, и ((upd(sβn, (oβ.actual, aβ), actual, (aβ, vγ′1), oα1.value, …, (aβ, vγ′m-1), oαm-1.value, (aβ, vγ′m), seq(oαm.value, …, oαn.value)), upd(oα, lcontext, aβ)), sβ(oα.body, fα), (sβ′, oγ, oβ)) ∈ Bα, то ((sβ, oα, oβ), iα, (sβ′, oγ, oβ)) ∈ Bα;
    - если sβ(oβ.special, fα) ≠ special, sβ(oβ.varied, fα) = varied, n ≥ m - 1, (sβ0, oγ0, oβ) = (sβ, (oα), oβ), ((sβ0, oγ0, oβ), vγ1, (sβ1, oγ1, oβ)) ∈ Tα, …, ((sβk-1, oγk-1, oβ), vγk, (sβk, oγk, oβ)) ∈ Tα, oαi ∈ oγi, oαi.jvalue = vβ и oαi.jtype = vβ для 1 ≤ i ≤ k-1, k ≤ n, oαk ∈ oγk, и неверно, что (oαk.jvalue = vβ и oαk.jtype = vβ), то ((sβ, oα, oβ), iα, (sβk, oγk, oβ)) ∈ Bα.

Последние два правила определяют вызов предопределенной и определяемой функций, соответственно. Правило для определяемой функции использует локальный контекст для хранения значений аргументов функции. Для каждого из этих правил рассматриваются 4 случая в зависимости от значений компонентов special и varied. Компонент special определяет, нужно ли вычислять аргументы вызова функции. Компонент varied определяет, может ли функция иметь переменное число аргументов. В случае, если может, и местность функции равна m, первые m-1 аргументов вызова функции соответствуют первым m-1 аргументам функции, а аргументы вызова функции, начиная с m-го, объединяются в атрибутон-последовательность, и полученное значение соответствует m-му аргументу функции.

# Синтаксис языка ASL

Язык ASL – регистро-зависимый язык.

Пусть SAtom = String ∪ Integer ∪ Double ∪ SQName – множество синтаксических атомов.

Любая синтаксическая конструкция языка ASL является выражением.

Множество выражений Eα определяется следующим образом:

* если eα ∈ SAtom, то eα ∈ Eα;
* если eα0, eα1, …, eαn ∈ Eα, то eαo(eα1, …, eαn) ∈ Eα – вызов функции eαo с аргументами eα1, …, eαn;
* если eα ∈ Eα, то (eα) ∈ Eα – выражение в скобках.

# Онтологические модели выражений

Вычисление выражения означает выполнение перехода в атрибутной системе типа ASL со входом, соответствующим этому выражению.

Вход, соответствующий выражению, называется онтологической моделью выражения.

Пусть {aγ1 = vα1, …, aγn = vαn} обозначает актуальный атрибутон с атрибутами aγ1, … aγn со значениями vα1, …, vαn в состоянии (sβ, oα, oβ).

Пусть seq(vα1, …, vαn) обозначает актуальный атрибутон-последовательность с элементами vα1, …, vαn в состоянии (sβ, oα, oβ).

Онтологическая модель om(eα) выражения eα в состоянии (sβ, oα, oβ) определяется следующим образом:

* если eα ∈ SAtom, то om(eα) = eα;
* если eα0, eα1, …, eαn ∈ Eα, то om(eαo(eα1, …, eαn)) = {functionCall = concept, function = om(eα0), arguments = seq(om(eα1), …, om(eαn))};
* если eα ∈ Eα, то om((eα)) = om(eα).

# Трансляция выражений в онтологические модели

В разделе 3 сделано предположение, что все атрибутоны, используемые в онтологической модели выражения, определены в текущем состоянии.

В этом разделе описан алгоритм трансляции omt(eα, sβ, oα, oβ) выражения eα в состоянии (sβ, oα, oβ) в его онтологическую модель. Результатом трансляции является множество троек (iα, sβ′, oα′), где (sβ′, oα′, oβ) – состояние, в котором уже все атрибутоны, используемые в онтологической модели выражения еα, определены, iα – вход, являющийся онтологической моделью выражения eα в состоянии (sβ′, oα′, oβ).

Алгоритм трансляции omt(eα, sβ, oα) определяется правилами:

* если eα ∈ SAtom, то omt(u, sβ, oα) = {(eα, sβ, oα)};
* если eα0, eα1, …, eαn ∈ Eα, (vα0, sβ0, oα0) ∈ omt(eα0, sβ0, oα0), (vα1, sβ1, oα1) ∈ omt(eα1, sβ0, oα0), …, (vαn, sβn, oαn) ∈ omt(eαn, sβn-1, oαn-1), и fc, args ∈ Q – потенциальные атрибутоны в состоянии (sβn, oαn, oβ), то (fc, upd(sβn, (fc, functionCall), concept, (fc, function), vα0, (fc, arguments), args, (args, seqLen), n, (args, start), 1, (args, 1), vα1, …, (args, n), vαn,)), upd(oβ, <actual, fc>, actual, <actual, args>, actual)) ∈ om(eαo(eα1, …, eαn));
* если eα ∈ Eα, то omt((eα)) = omt(eα).



