Разработка операционной семантики языков программирования

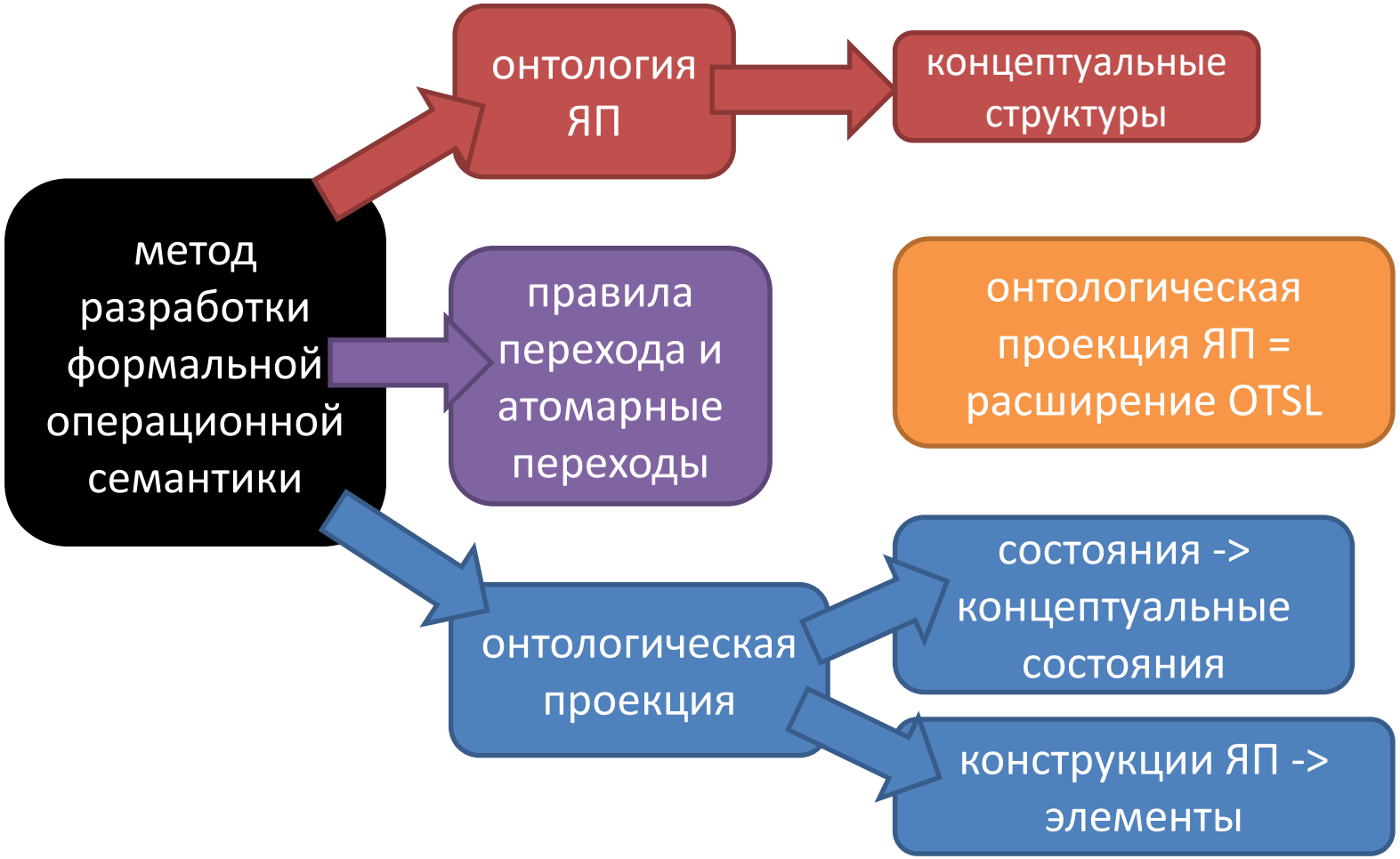
~~на основе двухэтапного метода концептуального проектирования информационных систем~~

на основе концептуальных систем переходов

Ануреев Игорь Сергеевич
Институт систем информатики имени А.П. Ершова
Новосибирск


формальные методы
программа на ЯП
формальная операционная семантика
надежность
эффективность
оптимальность ...
естественный язык
матем. формализм
системы автомат. доказательства
неявные
методика
паттерны
обозначения
терминология
прокрустово
ложе
естественная терминология
обозримость
понятность
быстрота изменения семантики
технологичность
предметно-ориентированный язык
ускорение разработки инструментов
ИКР ?
состояния
конструкции ЯП
выполнимость семант.
~~программирование~~
дедуктивная
верификация
тестирование


библиотека базовых элементов
0
атом
атрибут:значение
1
элемент::сорт
(элемент, …, элемент)
CTSL
язык
2
концептуали: целые -> элементы
(-1:value, 0:x, 1:variable)
(0:value)
концептуальные системы переходов
концептуальные структуры (элементы)
(0:program) = (элемент, …)
(…,
-2:пространство атрибутов,
-1:атрибут,
0:индивид,
1:понятие,
2:пространство понятий,
…)
отношение перехода
эндогенные отношения переходов
экзогенные атомарные переходы (transition образец … then (fun состояния x подстановки -> множества состояний))
концептуальные состояния: концептуали -> элементы
3
правила (rule образец var переменные seq переменные then тело)
((-1:value, 0:x, 1:variable):3,
(-1:type, 0:x, 1:variable):int)


метод разработки формальной операционной семантики
онтология ЯП
концептуальные структуры
правила перехода и атомарные переходы
онтологическая проекция ЯП = расширение OTSL
онтологическая проекция
состояния -> концептуальные состояния
конструкции ЯП -> элементы

Язык MPL-1 = расширение CTSL

типы: int и element (супертип);

константы: целые числа, true, und;

типизированные переменные;

выражения: +, -, *, div, mod, =, !=, <, >, <=, >=, and, or, not;

операторы: присваивание, блоки, условный, цикл while.

Онтология MPL-1

(rule (x is name) var (x) abn then (x is normal));

(0:имя, 1:variable);
(-1:type, 0:имя, 1:variable);
(-1:value, 0:имя, 1:variable);

(rule (x is variable) var (x) abn where (x is name)
 then (0:x, 1:variable));

(rule (x is type) var (x) abn then (x::q = int::q));
(rule (x is type) var (x) abn then (x::q = element::q)).

Правила для элементов MPL-1 Программы

(rule (program x y) var (x) seq (y) abn where (x is name)
  then (collect-variables y), y);

Декларации переменных

```
(rule (collect-variables (var x y), z) var (x, y) seq (z) abn (x, y) abn
 where ((x is name) and (not (x is variable)) and (y is type))
 then ((-1:type, 0:x, 1:variable) ::= y::q),
  ((0:x, 1:variable) ::= true), (collect-variables z));

(rule (collect-variables x, z) var (x) seq (z) abn
 then (collect-variables z));

(rule (collect-variables) abn then);

(rule (var x y) var (x, y) abn (x, y) abn then);
```

Переменные и константы

rule x var (x) abn (x) abn where (x is variable)
 then (-1:value, 0:x, 1:variable);

rule x var (x) abn (x) abn where (x is int) then x::q;

(rule (type in x) var (x) abn (x) abn where (x is variable)
 then (-1:type, 0:x, 1:variable);

(rule (type in x) var (x) abn (x) abn where (x is int) then
int::q);

(rule (type in x) var (x) abn (x) abn then element::q);

Операторы

```
(rule (x := y) var (x, y) val (y) abn (x) exc (y) abn
 where ((x is variable) and (let w be (type in x) in (y::* is w))
   and (not (y::* is exception)))
 then ((-1:value, 0:x, 1:variable) ::= y::*::q));

(rule (block x) seq (x) abn then x);

(rule (if::m x then y else z) var (x) seq (y, z) abn (x) abn
 then (if::exc x then (block y) else (block z)));

(rule (while::m x do y) var (x) seq (y) abn (x) abn
 then (while::exc x do (block y))).
```

Язык MPL-2 = расширение MPL-1

области видимости переменных (scopes)

(относительный) уровень вложенности x = число блоков, охватывающих x (scope)

Онтология MPL-2. Часть 1

(0:current-scope);

(-2:уровень, 0:имя, 1:variable);

(-2:уровень, -1:type, 0:имя, 1:variable);

(-2:уровень, -1:value, 0:имя, 1:variable);

Онтология MPL-2. Часть 2

```
(rule current-scope abn then (0:current-scope));

(rule (index in x) var (x) abn (x) abn where (x is name)
 then (let w be current-scope in (index in x, w)));

(rule (index in x, y) var (x, y) abn (x, y) abn then
 (if (-2:y, 0:x, 1:variable) then y::q
  else (if (y = 0) then und else (let w be (y - 1) in (index in x, w)))));

(rule (x is variable) var (x) abn then (index in x)).
```

Декларации переменных

```
(rule (collect-variables x) seq (x) abn
 then (collect-variables-1 () x);

(rule (collect-variables-1 (u), (var x y), z) var (x, y) seq (u, z)
  abn (x, y) abn
  where ((x is name) and (y is type)) then
  (let w be current-scope in
   (if (-2:w, 0:x, 1:variable) then und
    else ((-2:w, -1:type, 0:x, 1:variable) ::= y::q),
     ((-2:w, 0:x, 1:variable) ::= true),
     (collect-variables-1 (u, x) z))));
(rule (collect-variables-1 (u), x, z) var (x) seq (u, z) abn
 then (collect-variables-1 (u), z));
(rule (collect-variables-1 (u)) seq (u) abn then (u));
```

Переменные и константы

```
(rule x var (x) abn (x) abn then
 (let::und w be (index in x) in
  (-2:w, -1:value, 0:x, 1:variable)));

(rule (type in x) var (x) abn (x) abn then
 (let::und w be (index in x) in
  (-2:w, -1:type, 0:x, 1:variable)));
```

Операторы. Часть 1

```
(rule current-scope++ abn then
 ((0:current-scope) ::= ((0:current-scope) + 1)));

(rule current-scope-- abn then
 ((0:current-scope) ::= ((0:current-scope) - 1)));

(rule (x := y) var (x, y) val (y) abn (x) exc (y) abn
  where (not (y::* is exception)) then
  (let::und w be (index in x) in
    (if (let::und w2 be (-2:w, -1:type, 0:x, 1:variable)
          in (y::* is w2))
     then ((-2:w, -1:value, 0:x, 1:variable) ::= y::*::q)}
     else und)));
```

Операторы. Часть 2

```
(rule (block x) seq (x) abn
  then enter-block
  (let w1 be (collect-variables in x)
    in x, (catch::u w (exit-block in w1), (throw w::q))));

(rule enter-block abn then current-scope++);

(rule (exit-block in (x)) seq (x) abn
 then (delete-variables in x), current-scope--);
```

Операторы. Часть 3

```
(rule (delete-variables in x) seq (x) abn
  then (let w be current-scope
            in (delete-variables-1 in w, x)));

(rule (delete-variables-1 in x, y, z) var (x, y) seq (z)
  abn (x, y) abn then
   ((-2:x, 0:y, 1:variable) ::=),
   ((-2:x, 1:type, 0:y, 1:variable) ::=),
   ((-2:x, -1:value, 0:y, 1:variable) ::=),
   (delete-variables-1 in x, z));

(rule (delete-variables-1 in x) var (x) abn (x) abn then).
```

Спасибо!