Проверка контекстных условий

Гладков Артемий Николаевич

23.10.2024

Задача проверки контекстных условий

В программе перед запуском исполнения требуется проверить разные контекстные условия.
Примеры контекстных условий:

- Каждый используемый идентификатор (переменная, функция, класс, тип и т.д) должен быть определён и не более одного раза.
- Число параметров, переданных в функцию, должно соответствовать количеству параметров в объявлении функции.
- Ограничения на типы операндов.

Процесс проверки этих условий называют семантическим анализом.

Задача проверки контекстных условий

Проверку некоторых контекстных условия можно совместить с синтаксическим анализом.

Задача проверки контекстных условий

Проверку некоторых контекстных условия можно совместить с синтаксическим анализом.
Как только синтаксический анализатор распознает конструкцию, на компоненты которой наложены некоторые ограничения, будем проверять их выполнение.

Задача проверки контекстных условий

Проверку некоторых контекстных условия можно совместить с синтаксическим анализом.
Как только синтаксический анализатор распознает конструкцию, на компоненты которой наложены некоторые ограничения, будем проверять их выполнение.
Таким образом, необходимо встроить в процесс синтаксического анализа дополнительные «синтаксические» действия. Такие действия как раз удобно встраивать в метод рекурсивного спуска.

Добавление семантических действий

Расширим понятие КС-грамматики, добавив в ее правила вывода символы-действия. Пусть в грамматике есть правило вывода:
$A \to a < D_1 > B < D_1; D_2 > | \, bC < D_3 >$

```
void A() {
  if (c=='a') {
    c = fgetc(fp);
    D1();
    B();
    D1(); D2();
  }
  else if (c == 'b') {
    c = fgetc(fp);
    C();
    D3();
  }
  else
    ERROR();
}
```

Контекстные условия М-языка

Контекстные условия, выполнение которых нам надо контролировать в программах на М-языке, таковы.

- Любое имя, используемое в программе, должно быть описано и только один раз.
- В операторе присваивания типы переменной и выражения должны совпадать.
- В условном операторе и в операторе цикла в качестве условия возможно только логическое выражение.
- Операнды операции отношения должны быть целочисленными.
- Тип выражения и совместимость типов операндов в выражении определяются по обычным правилам (как в Паскале).

Обработка описаний

На этапе лексического анализатора уже будут известны все идентификаторы, внесём их в таблицу.

```
struct record {
    string name; // идентификатор
    bool is_declared = false;// Объявлен ли?
    string type; // тип переменной
};
vector<record> TID;
```

Поле type заполним на этапе синтаксического анализа.
i-й элемент таблицы соответствует идентификатору под номером i.

Обработка описаний

void decid(int i, Type type) – в i-той строке таблицы TID контролирует и заполняет поле declare и, если лексема (4,i) впервые встретилась в разделе описаний, заполняет поле type:

```
void decid(int i, string type){
    if (TID[i].is_declared)
        ERROR(); // повторное описание
    else {
        TID[i].type = type;
        TID[i].is_declared = true;
    }
}
```

Обработка описаний

Раздел описаний имеет вид $D \to I\{,I\} : [\textbf{int} \mid \textbf{bool}]$ Преобразованный будет иметь вид:

$$D \to I < ipush(curr\_lex.Value) > \{,I < ipush(curr\_lex.Value) >\} : [int < dec(”int”) > | bool < dec(”bool”) >]$$

```
void ipush(int i); // целое в стек
int ipop(); // целое из стека
bool iempty();// пуст ли стек целых чисел
void dec (string type)
{
  int i;
  while (!iempty())
    decid(ipop(), type);
}
```

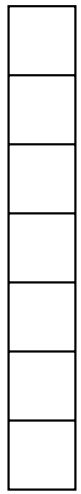
Проверка контекстных условий выражений

Пусть есть функция `string gettype(string op, string type1, string type2)`, которая проверяет корректность применения операции op к типам операндов `type1` и `type2`, и возвращает получаемый тип или "no если операция некорректна.
Определим вспомогательные функции:
Встречая лексему, представляющую собой числовую или логическую константу, добавляем её в стек с помощью `spush("int")` или `spush("bool")`.

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```



Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.
Рассмотрим выражение: $a := a/2 + b * 54$

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

Рассмотрим выражение: $a := a/2 + b * 54$

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.
Рассмотрим выражение: $a := a/2 + b * 54$

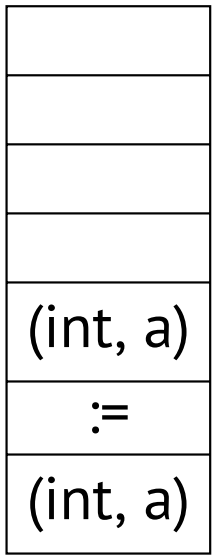
Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

Рассмотрим выражение: $a := a/2 + b * 54$



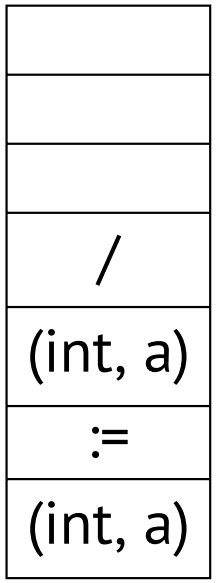
Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

Рассмотрим выражение: $a := a/2 + b * 54$



Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

Рассмотрим выражение: $a := a/2 + b * 54$

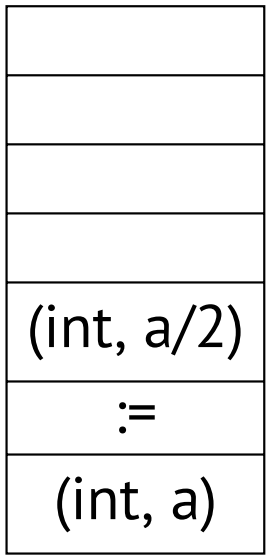
(int, 2)
/
(int, a)
:=
(int, a)

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.
Рассмотрим выражение: $a := a/2 + b * 54$



Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.
Рассмотрим выражение: $a := a/2 + b * 54$

+
(int, a/2)
:=
(int, a)

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

Рассмотрим выражение: $a := a/2 + b * 54$

(int, b)
+
(int, a/2)
:=
(int, a)

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

Рассмотрим выражение: $a := a/2 + b * 54$

*
(int, b)
+
(int, a/2)
:=
(int, a)

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

Рассмотрим выражение: $a := a/2 + b * 54$

(int, 54)
*
(int, b)
+
(int, a/2)
:=
(int, a)

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

Рассмотрим выражение: $a := a/2 + b * 54$

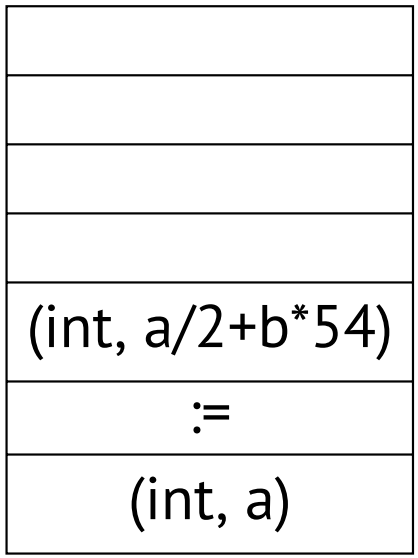
(int, b*54)
+
(int, a/2)
:=
(int, a)

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

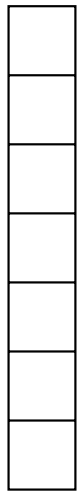
Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.
Рассмотрим выражение: $a := a/2 + b * 54$



Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

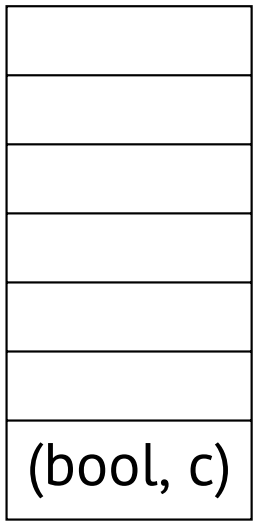


Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.
Рассмотрим выражение: $a := a/2 + b * 54$

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

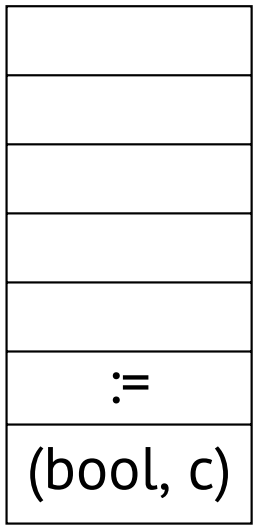


Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.
Рассмотрим выражение: $c := c\ and\ (a = b)$

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```



Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.
Рассмотрим выражение: $c := c\ and\ (a = b)$

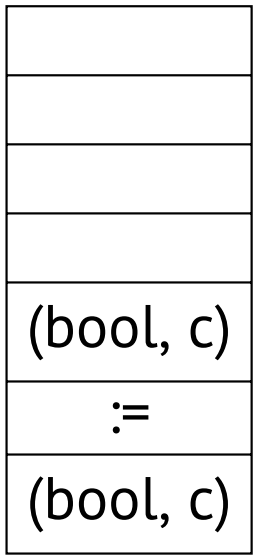
Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

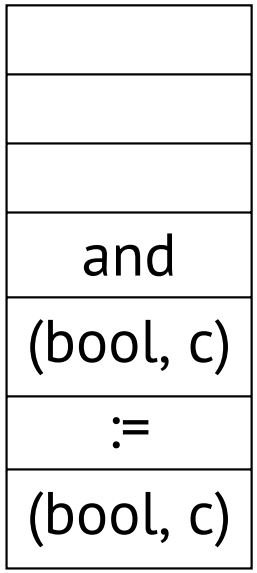
Рассмотрим выражение: $c := c\ and\ (a = b)$



Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```



Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.
Рассмотрим выражение: $c := c\ and\ (a = b)$

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

Рассмотрим выражение: $c := c\ and\ (a = b)$

(int, a)
and
(bool, c)
:=
(bool, c)

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

Рассмотрим выражение: $c := c\ and\ (a = b)$

=
(int, a)
and
(bool, c)
:=
(bool, c)

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

Рассмотрим выражение: $c := c\ and\ (a = b)$

(int, b)
=
(int, a)
and
(bool, c)
:=
(bool, c)

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

Рассмотрим выражение: $c := c\ and\ (a = b)$

(bool, a=b)
and
(bool, c)
:=
(bool, c)

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```

Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.

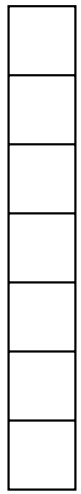
Рассмотрим выражение: $c := c\ and\ (a = b)$

(bool, c and a=b)
:=
(bool, c)

Проверка контекстных условий выражений

Пример: проверить корректность следующих выражений:

```
int a,b;
bool c;
a := a / 2 + b * 54;
c := c and (a = b);
```



Для проверки применяем стек. Типы операндов и операции сохраняем в стек и, с соблюдением приоритета операций, вычисляем результирующий тип, пока не вычислим тип всего выражения.
Рассмотрим выражение: $c := c\ and\ (a = b)$

Проверка контекстных условий выражений

Получаемая грамматика:

$E \to E_1 \mid E_1[= \mid < \mid > \mid !=] < push(oper) > E_1 < check\_op() >$

$E_1 \to T\{[+ \mid - \mid \textbf{or}] < push(oper) > T < check\_op() >\}$

$T \to F\{[* \mid / \mid \textbf{and}] < push(oper) > F < check\_op() >\}$

$F \to I < check\_declared(); push(type) > \mid N < push(int) > \mid L < push(bool) > \mid \textbf{not}\, F < check\_not() > \mid (E)$

Что дальше?

- Посмотреть видео с моей реализацией семантического анализатора: <ссылка>.

Что дальше?

- Посмотреть видео с моей реализацией семантического анализатора: <ссылка>.
- Начать делать работу №3 по семантическому анализатору.

Лабораторная работа №3

- Разбивка по вариантам и сами варианты будут выложены в курсе в moodle. И туда же нужно прикрепить решения.
- Что требуется? — Реализовать семантический анализатор.
- Дедлайн: 6 недель.
- Когда сдавать? — На следующих парах или online(назначим созвон на один из вечеров).

Будущие лекции

- Занятие 4: первая половина пары — генерация внутренних представлений.
- Занятие 5: первая половина пары — исполнение кода.
- Занятие 6 — приём Лабораторных. Или конец материала, если что-то не успеем.