Обход AST-дерева с помощью визиторов

Разработка оптимизирующих компиляторов
ФИИТ, магистратура, 2 семестр
мехмат ЮФУ

AST-дерево – что дальше?

- После получения AST-дерева его можно обойти и выполнить различные задачи:
  - генерация трёхадресного кода (для последующей оптимизации)
  - непосредственная интерпретация
  - генерация отформатированного кода на том же языке программирования (Pretty Printer)
  - генерация кода на другом языке программирования
  - простейший анализ кода
- Как обойти дерево с различными типами узлов?

Другие способы обхода дерева

- Можно обходить дерево рекурсивно. Например, в постфиксном порядке: поддеревья слева направо, а потом корень
- При обходе каждого узла можно возвращать некоторое значение (например, список сгенерированных трехадресных команд или имя временной переменной, в которой будет храниться подвыражение)
- При рекурсивном обходе всё равно придётся делать switch по типам (скажем, при обходе statement необходимо проверять, какой подтип statement находится в узле и в зависимости от этого выполнять разные действия)

Базовый класс визитора

```
public abstract class Visitor
{
    public virtual void VisitIdNode(IdNode id) { }
    public virtual void VisitIntNumNode(IntNumNode num) { }
    public virtual void VisitBinOpNode(BinOpNode binop) { }
    public virtual void VisitAssignNode(AssignNode a) { }
    public virtual void VisitCycleNode(CycleNode c) { }
    public virtual void VisitBlockNode(BlockNode bl) { }
    public virtual void VisitWriteNode(WriteNode w) { }
}
```

Изменения в классах узлов AST-дерева

```
public abstract class Node
{
  public abstract void Visit(Visitor v);
}

public class AssignNode: Node
{
  ...
  public override void Visit(Visitor v)
  {
    v.VisitAssignNode(this);
  }
}
```

Автовизитор с логикой обхода по умолчанию

```
class AutoVisitor : Visitor {
    public override void VisitBinOpNode(BinOpNode binop) {
        binop.Left.Visit(this);
        binop.Right.Visit(this);
    }
    public override void VisitAssignNode(AssignNode a) {
        a.Id.Visit(this);
        a.Expr.Visit(this);
    }
    public override void VisitCycleNode(CycleNode c) {
        c.Expr.Visit(this);
        c.Stat.Visit(this);
    }
    public override void VisitBlockNode(BlockNode bl) {
        foreach (var st in bl.StList)
            st?.Visit(this);
    }
    public override void VisitWriteNode(WriteNode w) {
        w.Expr.Visit(this);
    }
}
```

Пример визитора – подсчёт присваиваний

```
class AssignCountVisitor : AutoVisitor
{
    public int Count = 0;
    public override void VisitAssignNode(AssignNode a)
    {
        Count += 1;
    }
    public override void VisitWriteNode(WriteNode w) { }
    public override void VisitVarDefNode(VarDefNode w) { }
}

static void Main(string[] args)
{
    var avis = new AssignCountVisitor();
    parser.root.Visit(avis);
    WriteLine($"Количество присваиваний = {avis.Count}");
}
```

Зачем присутствует пустой метод VisitWriteNode

Последовательность вызовов

```
parser.root.Visit(avis);
```

```
avis.VisitBlockNode(parser.root);
```

```
foreach (var st in bl.StList)
   st.Visit(avis);
```

```
v.VisitAssignNode(st);
```

```
avis.Count += 1;
```

Пример визитора – Pretty Printer

Сервисные методы

```
class PrettyPrintVisitor: Visitor
{
    public string Text = "";
    private int Indent = 0;

    private string IndentStr()
    {
        return new string(' ', Indent);
    }
    private void IndentPlus()
    {
        Indent += 2;
    }
    private void IndentMinus()
    {
        Indent -= 2;
    }
}
```

Пример визитора – Pretty Printer (2)

Методы Visit

```
public override void VisitIdNode(IdNode id) {
    Text += id.Name;
}
public override void VisitIntNumNode(IntNumNode num) {
    Text += num.Num.ToString();
}
public override void VisitBinOpNode(BinOpNode binop) {
    Text += "(";
    binop.Left.Visit(this);
    Text += " " + binop.Op + " ";
    binop.Right.Visit(this);
    Text += ")";
}
public override void VisitAssignNode(AssignNode a) {
    Text += IndentStr();
    a.Id.Visit(this);
    Text += " := ";
    a.Expr.Visit(this);
}
```

Пример визитора – Pretty Printer (3)

```
public override void VisitCycleNode(CycleNode c) {
    Text += IndentStr() + "cycle ";
    c.Expr.Visit(this);
    Text += Environment.NewLine;
    c.Stat.Visit(this);
}

public override void VisitBlockNode(BlockNode bl) {
    Text += IndentStr() + "begin" + Environment.NewLine;
    IndentPlus();

    var Count = bl.StList.Count;

    if (Count>0)
        bl.StList[0].Visit(this);
    for (var i = 1; i < Count; i++) {
        Text += ';';
        if (!(bl.StList[i] is EmptyNode))
            Text += Environment.NewLine;
        bl.StList[i].Visit(this);
    }
    IndentMinus();
    Text += Environment.NewLine + IndentStr() + "end";
}
```

Генерация трёхадресного кода для выражения

```
override void VisitAssignNode(AssignNode a)
{
    string tmp = gen(a.Expr);
    genCommand(a.Id + "=" + tmp);
}

string gen(ExprNode ex)
{
    if (ex.GetType() == typeof(BinExpr))
    {
        bin = (BinExpr)ex;
        string tmp1 = gen(bin.Left);
        string tmp2 = gen(bin.Right);
        string tmp = genTmpName();
        genCommand(tmp + "=" + tmp1 + bin.op + tmp2);
        return tmp;
    }
  ...
}
```

Генерация трёхадресного кода для условного оператора

Код:

```
if (усл)
  оп1;
else оп2;
```

Трёхадресный код:

```
t = усл
if t goto L1
оп2
goto L2
L1: оп1
L2:
```

```
override void VisitIfNode(IfNode n)
{
    string tmp = gen(n.Cond);
    string L1 = genTmpLabel();
    string L2 = genTmpLabel();
    genCommand("if " + tmp + " goto " + L1);
    n.operator2.Visit(this);
    genCommand("L1: nop");
    n.operator1.Visit(this);
    genCommand("L2: nop");
}
```

Визиторы – способ обхода дерева

- Визитор является одним из паттернов проектирования (GOF – Гамма и другие)
- Визитор позволяет отделить логику обхода от действий, выполняемых в каждом узле
- Один визитор – одна группа действий над узлами дерева
- Недостатки базового паттерна "визитор" – при обходе не запоминается контекст (где мы были до этого). Если требуется, контекст надо запоминать самостоятельно
- Визиторы можно сочетать с другими способами обхода дерева

Визиторы и оптимизация

- Визиторы позволяют выполнять простейшую оптимизацию по дереву
- Для этого необходим сервис поиска поддеревьев по заданному паттерну и замены поддеревьев
- Технически следует поддерживать в каждом узле ссылку Parent

Визитор с заполнением Parent

```
class FillParentVisitor : AutoVisitor {
    Stack<Node> st = new Stack<Node>(); // можно заменить на List
    public override void VisitBinOpNode(BinOpNode binop)
    {
        binop.Parent = st.Top();
        st.Push(binop);
        base.VisitBinOpNode(binop);
        st.Pop();
    }
    public override void VisitAssignNode(AssignNode a)
    {
        a.Parent = st.Top();
        st.Push(a);
        base.VisitAssignNode(a);
        st.Pop();
    }
    ...
}
```

Визитор с заменой поддеревьев. ReplaceExpr

```
class ChangeVisitor : AutoVisitor {
    public void ReplaceExpr(ExprNode from, ExprNode to) {
       var p = from.Parent;
       to.Parent = p;
       if (p is AssignNode assn)
       {
         assn.Expr = to;
       }
       else if (p is BinOpNode binopn)
       {
         if (binopn.Left == from) // Поиск подузла в Parent
           binopn.Left = to;
         else if (binopn.Right == from)
           binopn.Right = to;
       }
       else if (p is StatListNode)
       {
         throw new Exception("Родительский узел не содержит выражений");
       }
    }
    ...
}
```

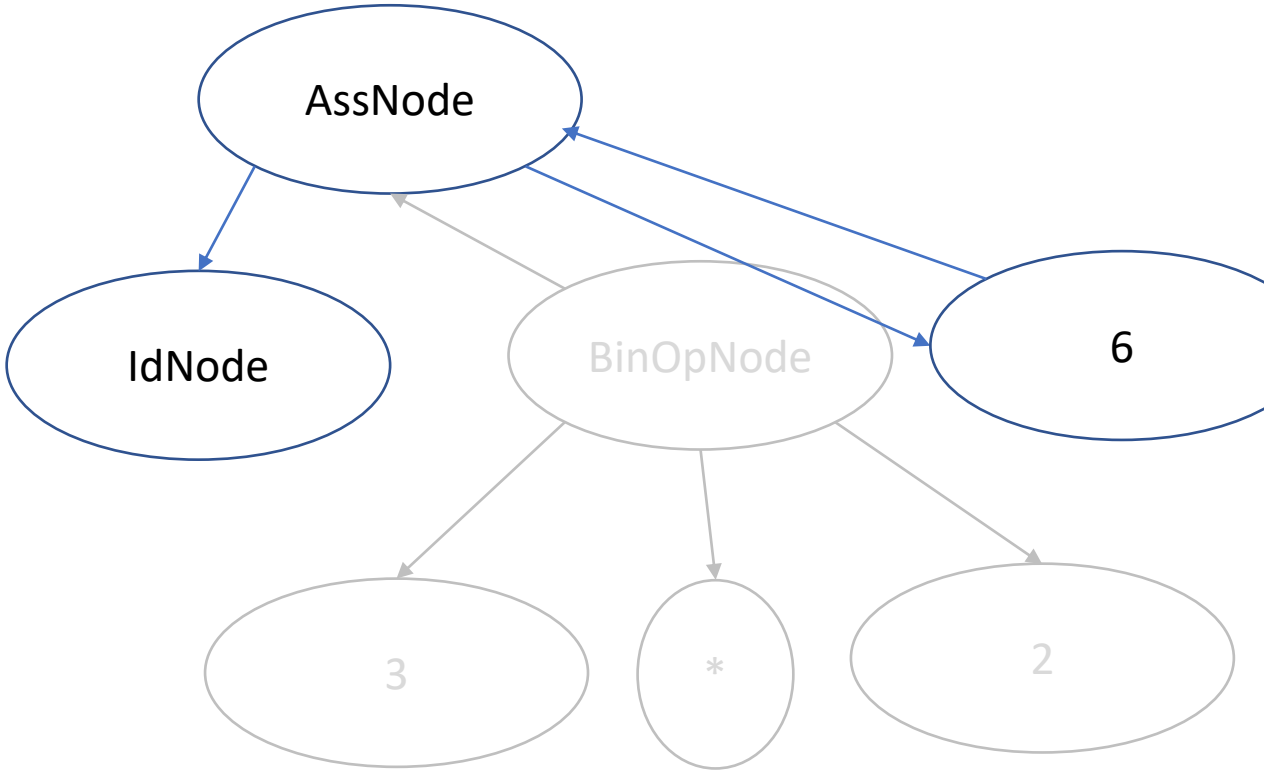
Использование Parent для замены в дереве по шаблону

```
class Opt1Visitor : ChangeVisitor {
    public override void VisitBinOpNode(BinOpNode binop) {
        if (binop.Left is IntNode && (binop.Left as IntNode).Value == 1 &&
          binop.op == '*')
        {
          binop.Right.Visit(this); // Вначале сделать то же в правом поддереве
          ReplaceExpr(binop,binop.Right); // Заменить себя на своё правое поддерево
        }
        else // Если оптимизаций нет, то
        {
          base.VisitBinOpNode(binop); // Обойти потомков обычным образом
        }
    }
    ...
}
```

a = 3 * 2

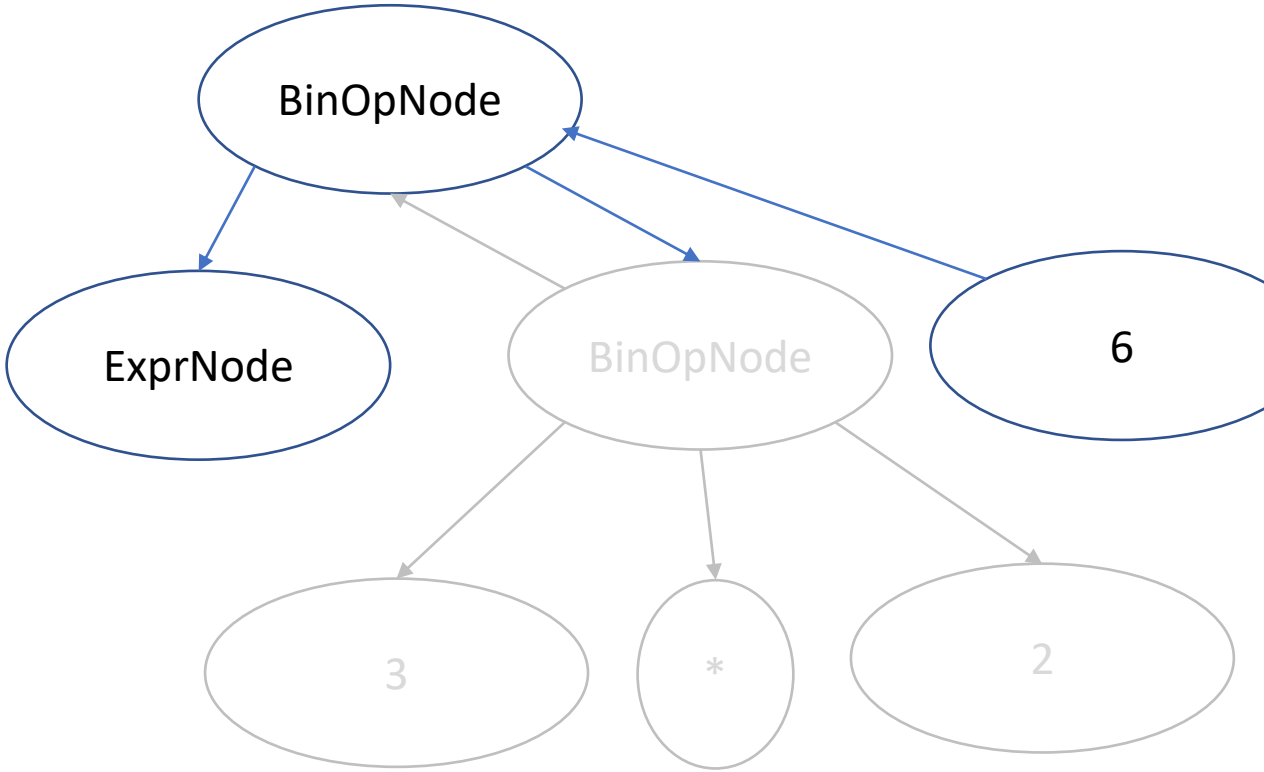
a = 2;

a = b * 1;



a = 3 * 2

a = 2;



Визитор с заменой поддеревьев. ReplaceStat

```
class ChangeVisitor : AutoVisitor {
    public void ReplaceStat(StatNode from, StatNode to) {
       var p = from.Parent;
       if (p is AssignNode || p is ExprNode …)
       {
         throw new Exception("Родительский узел не содержит операторов");
       }
       to.Parent = p;
       if (p is BlockNode bln) // Можно переложить этот код на узлы!
       {
         for (var i=0; i<bln.lst.Count; i++)
           if (bln.lst[i] == from)
             bln.lst[i] = to;
       }
       else if (p is IfNode ifn) {
         if (ifn.ThenStat == from) // Поиск подузла в Parent
           ifn.ThenStat = to;
         else if (ifn.ElseStat == from)
           ifn.ElseStat = to;
       }
    }
    ...
}
```

Оптимизирующая замена оператора (x = x; if (true)…; if (false)…)

```
class Opt2Visitor : ChangeVisitor {
    public override void VisitAssignNode(AssignNode ass) {
        if (ass.Expr is IdNode idn && ass.Id.Name == idn.Name) {
          ReplaceStat(ass,null); // Заменить на null.
                                 // Потом этот null надо специально проверять!!!
        }
        // Не обходить потомков
    }
    public override void VisitIfNode(IfNode ifn) {
        if (ifn.Expr is BoolNumNode bnn && bnn.Value == "false") {
          if (ifn.ElseStat != null)
            ifn.ElseStat.Visit(this); // Вначале обойти потомка
          ReplaceStat(ifn, ifn.ElseStat);
        }
        else if (ifn.Expr is BoolNumNode bnn && bnn.Value == "true") {
          if (ifn.ThenStat != null)
            ifn.ThenStat.Visit(this); // Вначале обойти потомка
          ReplaceStat(ifn, ifn.ThenStat);
        }
        else {
          base.VisitIfNode(ifn);
        }
    }
}
```

Дальнейшая оптимизация – исключение null - операторов

```
class Opt3Visitor : ChangeVisitor
{
    public override void VisitBlockNode(BlockNode bln)
    {
       bln.lst = bln.lst.Where(x=>x!=null).ToList();
    }
    public override void VisitIfNode(IfNode ifn)
    {
      if (ifn.ElseStat == null && ifn.ThenStat == null)
      {
        ReplaceStat(ifn, null);
      }
    }
}
```

Проблема данного кода. Вначале мы можем обойти BlockNode и удалить null-узлы, а потом обойти IfNode – и null-узлы снова появятся.
Один из выходов – обходить от листов к корню. **Как?**

Проблема

```
class Opt3Visitor : ChangeVisitor
{
    public override void VisitBlockNode(BlockNode bln)
    {
       bln.lst = bln.lst.Where(x=>x!=null).ToList();
       base.VisitBlockNode(bln);
    }
    public override void VisitIfNode(IfNode ifn)
    {
      if (ifn.ElseStat == null && ifn.ThenStat == null)
      {
        ReplaceStat(ifn, null);
      }
    }
}
```

```
{
   x = 3;
   null;
   x = 4;
   {
      x = 5;
      null; // не будет обойдён
   }
}
```

Проблема 2

```
class Opt3Visitor : ChangeVisitor
{
    public override void VisitBlockNode(BlockNode bln)
    {
       bln.lst = bln.lst.Where(x=>x!=null).ToList();
       if (bln.lst.Count == 0)
           ReplaceStat(bln,null);
       else base.VisitBlockNode(bln);
    }
    public override void VisitIfNode(IfNode ifn)
    {
      if (ifn.ElseStat == null && ifn.ThenStat == null)
      {
        ReplaceStat(ifn, null);
      }
```

```
{
   if (a == 0)
      null;
   else
   {
      null;
   }
}
```

PreVisit и PostVisit

```
class AutoVisitor : Visitor {
    public virtual void PreVisit(Node n) {}  // переопределить в потомках
    public virtual void PostVisit(Node n) {} // переопределить в потомках
    public override void VisitBinOpNode(BinOpNode binop) {
        PreVisit(binop);
        binop.Left.Visit(this);
        binop.Right.Visit(this);
        PostVisit(binop);
    }
    public override void VisitAssignNode(AssignNode a) {
        PreVisit(binop);
        a.Id.Visit(this);
        a.Expr.Visit(this);
        PostVisit(binop);
    }
    public override void VisitCycleNode(CycleNode c) {
        PreVisit(binop);
        c.Expr.Visit(this);
        c.Stat.Visit(this);
        PostVisit(binop);
    }
    ...
}
```

Решение с помощью PreVisit и PostVisit

```
class Opt3Visitor : ChangeVisitor
{
    public override void PostVisit(Node n) // первыми обходятся самые внутренние
    {
        if (n is IfNode ifn)
            if (ifn.ElseStat == null && ifn.ThenStat == null)
            {
                ReplaceStat(ifn, null);
            }
        else (n is BlockNode bln)
        {
            bln.lst = bln.lst.Where(x=>x!=null).ToList();
        }
    }
}
```

Старое решение для сравнения

```
class Opt3Visitor : ChangeVisitor
{
    public override void VisitBlockNode(BlockNode bln)
    {
       bln.lst = bln.lst.Where(x=>x!=null).ToList();
    }
    public override void VisitIfNode(IfNode ifn)
    {
      if (ifn.ElseStat == null && ifn.ThenStat == null)
      {
        ReplaceStat(ifn, null);
      }
    }
}
```

Задания на оптимизации по дереву

До	После
`1 * ex, ex * 1, ex / 1`	`ex`
`0 * expr, expr * 0`	`0`
`2 * 3`	`6`
`0 + expr`	`expr`
`a - a`	`0`
`2 < 3`	`true`
`2 == 4`	`false`
`a == a, a >= a`	`true`
`a > a, a != a`	`false`
`x = x`	`null`
`if (true) st1; else st2;`	`st1`
`if (false) st1; else st2;`	`st2`
`if (ex) null; else null;`	`null`
`while (false) st;`	`null`

Проблемы

Проблемы	
2 * a * 3	
2 * 3 * a	
6 * a	

Задания по командам

Команда	До	После
ТатароваШкуро	`1 * ex, ex * 1, ex / 1`	`ex`
РыжЕвс	`0 * expr, expr * 0`	`0`
Потапов	`2 * 3`	`6`
РыжЕвс	`0 + expr`	`expr`
Потапов	`a - a`	`0`
ЛутчПисьм	`2 < 3`	`true`
УшПац	`2 == 4`	`false`
ВолМозд	`a == a, a >= a`	`true`
ГалЧерк	`a > a, a != a`	`false`
КарКар	`x = x`	`null`
ТатаШк,ВолМозд	`if (true) st1; else st2;`	`st1`
ГалЧерк ЛутчПисьм	`if (false) st1; else st2;`	`st2`
КарКар	`if (ex) null; else null;`	`null`
УшПац	`while (false) st;`	`null`

Задания по командам 2

Команда	До	После
Манукян	`1 * ex, ex * 1, ex / 1`	`ex`
ГарькРудн	`0 * expr, expr * 0`	`0`
	`2 * 3`	`6`
Манукян	`0 + expr`	`expr`
ГарькРудн	`a - a`	`0`
ОстапГурт	`2 < 3`	`true`
АгафЧух	`2 == 4`	`false`
ПогДомб	`a == a, a >= a`	`true`
МаслОсм	`a > a, a != a`	`false`
ЧубРом	`x = x`	`null`
АгафЧух ПогДомб	`if (true) st1; else st2;`	`st1`
МаслОсм	`if (false) st1; else st2;`	`st2`
ЧубРом	`if (ex) null; else null;`	`null`
ОстГурт	`while (false) st;`	`null`

В каком порядке делать оптимизации

Op1 2 – 2 = 0
Op2 0 + a
Op3

a - ((a - a) + a)
a – (0 + a)
a – a
0

Op1
2 – 2 + a
0 + a

Opt1 + Opt2

Op2
0 + a
a

Op1-нет Op2-нет
Op3

Op1-нет Op2-нет Op3 - нет
Op4

Q & A