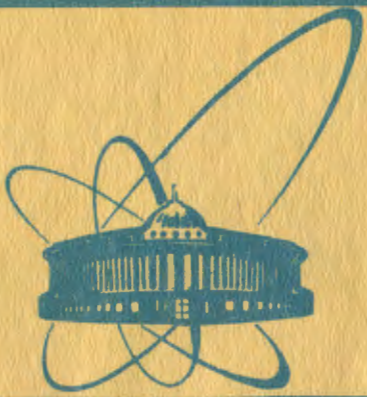


сообщения
объединенного
института
ядерных
исследований
дубна

P11-80-804

Ли Рен Хи, А.А.Хошенко

ДВУМЕРНОЕ ИЗОБРАЖЕНИЕ ПРАВИЛ CDL - ОСНОВА ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ CDL

1980

О нетехнологичности записи программ на языках программирования или в виде традиционных блок-схем упоминает И.В.Вельбицкий в монографии "Технологический комплекс производства программ на ЕС ЭВМ и БЭСМ-6" /I/. В нашей работе предлагается интерпретация изображения R -программ И.В.Вельбицкого применительно к правилам языка CDL-1 . Изображение R -программ основано на использовании ориентированных нагруженных графов. Логические структуры данных в CDL отображаются правилами, поэтому проблема изображения структур данных в CDL снимается.

Рассмотрим пример из программы CDL-1 , смысловое значение которого для данной работы неважно:

```
warning + text + info - oldpos - val:
  warning out flag,

    (was tag + info, inform + info,
      position + old  pos;
     was cons + info, get + info + val,
      outint1 + val, position + oldpos;
    outint1 + info, position + oldpos);.
```

В этом примере предикаты подчеркнуты.

Для понимания логической структуры этого правила CDL-1 дадим его алгольную запись:

```
procedure warning (text,info);
integer text,info,oldpos,val;
begin if warning out flag
      then if was tag (info)
           then begin inform (info);
                      position (oldpos)
                end
           else if was cons (info)
                then begin get (info,val);
                           outint1 (val);
                           position (oldpos)
                     end
                else begin outint1 (info);
                           position (oldpos)
                     end
      else
end;
```

Теперь дадим диаграмму приведенного правила, которую мы назовем CDL- правилом по той причине, что эта схема является однозначным соответствием рукописному тексту этого правила:

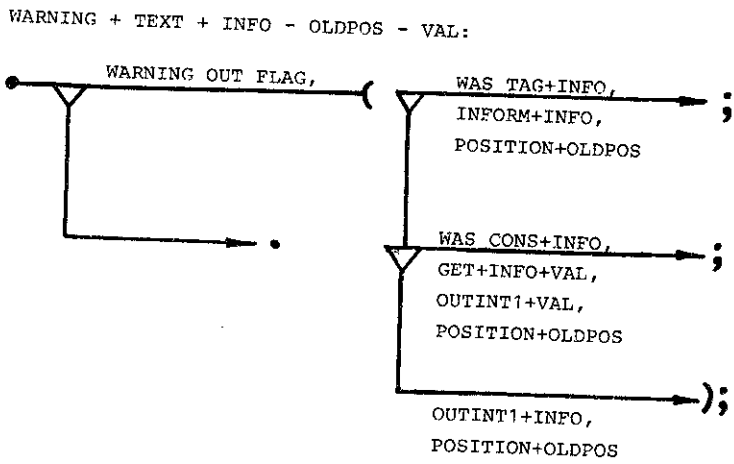


Рис. I

На схеме треугольники обозначают узлы, ребра-альтернативы правила. Над ребром мы записываем предикат, под ребром - действия, которые мы выполняем, если логическое значение этого предиката есть "истина". Ребра, идущие вниз от узла, соответствуют передаче управления из узла, если его предикат принял логическое значение "ложь".

Дадим "скелет" правила, тогда,если идти из вершины правила по пунктирной линии,одновременно и последовательно нумеруя узлы графа, то мы получим алгоритм соответствия ориентированного графа данного правила текстуальной записи правила на языке CDL:

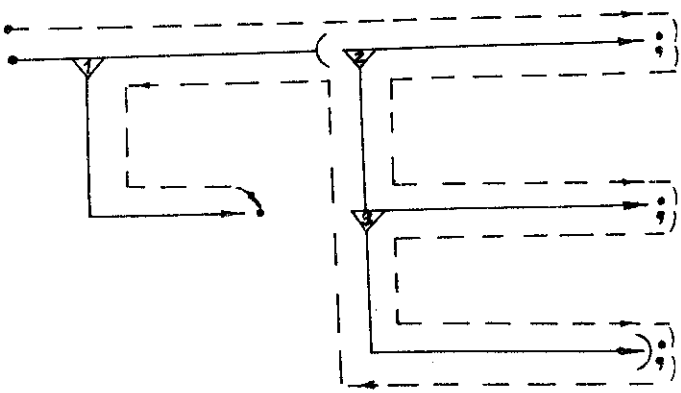


Рис. 2

Таким образом, при использовании этого метода изображения правил CDL в силу его наглядности можно использовать малоквалифицированных специалистов при подготовке больших программ. С нашей точки зрения,этот тип схем позволяет хорошо документировать отчеты по программному обеспечению и одновременно, используя метод главного программиста, сокращать сроки реализации больших систем.

Двумерное изображение правил CDL можно обобщить на все их синтаксические конструкции. Для иллюстрации этого обобщения приведем пример текста, который содержит метки, команды безусловного выхода из правила с логическим значением "ложь" со знаком "-".

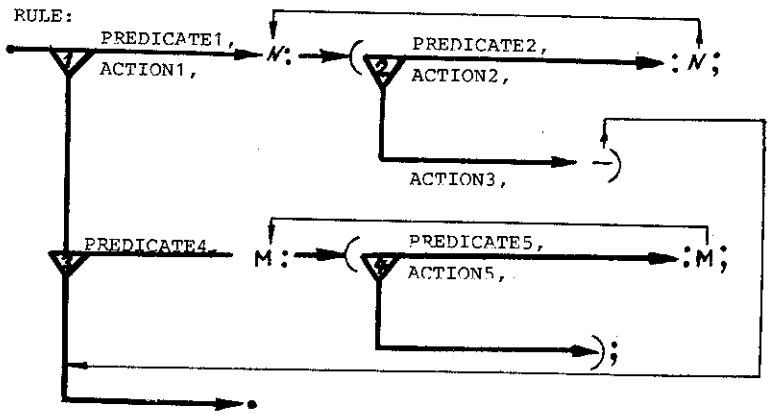


Рис. 3

Тогда, если следовать описанному алгоритму восстановления текста правила с поправкой: обход производить только по толстым линиям, не рассматривая тонкие линии передачи управления, то мы получим следующий текст:

```
rule:
     predicate1, action1,
 n: (predicate2, action2,:n;
     action3,-);
     predicate4,
 m: (predicate5, action5,:m;);.
```

В практике программирования зачастую встречаются ситуации, когда по тексту программы нужно восстановить эквивалентную ей блок-схему (для разбора программы либо для ее документирования). Самое тривиальное решение этой задачи - это встроить в транслятор CDL генератор скелета правил на АЦПУ. Более сложное решение -

выдавать полное двумерное изображение как правил, так и всей программы на АЦПУ либо на другое, фиксирующее графическое изображение устройство. Например, скелет самого сложного и текстуально самого большого правила в трансляторе CDL на БЭСМ-6 изображается полностью на АЦПУ без типичных разрывов для сложных блок-схем, когда мы пытаемся их изобразить на листах бумаги. Однако большой потребности при интенсивной работе пользователя в выдаче на АЦПУ скелета правила нет, так как скелет очень длинного правила можно построить вручную всего за несколько минут.

В дальнейшей части данной работы будет показано, как можно использовать двумерное изображение правил при поиске ошибок, более того, станет очевидным, что некоторые трудноуловимые в тексте ошибки практически невозможны при графическом изображении правил.

Одной из типичных ошибок пользователя CDL является неправильная структуризация скобок при перечислении длинного ряда альтернатив, внутри которых тоже присутствуют скобочные структуры. Вообще в языках, использующих операционные скобки / в CDL - это "(" , ")", а в Алголе - begin, end /, это типичные ошибки. При первой же попытке использовать нагруженные графы для изображения правил CDL была обнаружена старая ошибка в генераторе кодов для макросов пользователя в трансляторе CDL-1 на БЭСМ-6 и за несколько часов удалось вообще избавиться от всех подобных ошибок и неточностей в этом трансляторе.

Рассмотрим нарисованную выше скелетную схему, изображающую корректное правило. Теперь дадим аналогичную схему правила с неправильной расстановкой скобок:

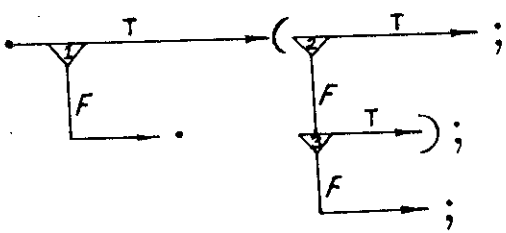


Рис. 4

В CDL -правиле закрывающая скобка не может стоять на ребре графа, исходящем из узла графа по горизонтали. Наличие этой ошибки трудно обнаружить визуально по тексту правила.

Перечисленные эмпирические находки и их простота в использовании намного облегчают теперь возможность строгой детализации транслятором структурных ошибок пользователя в его правилах и частичного их исправления. Заодно отметим, что именно CDL в силу строгой структурности правил допускает простое решение этой задачи, которая является камнем преткновения для известных нам неструктурных языков.

Допустим, что

k есть порядковый номер узла правила, получаемый вышеописанным алгоритмом (см. стр.5);

ℓ – текущее число открывающих скобок в правиле;

m – текущее число закрывающих скобок в правиле;

n – true ребро, false ребро } – указатель типа текущего ребра;

p есть счетчик узлов правила на ребре TRUE, которым не предшествует открывающая скобка.

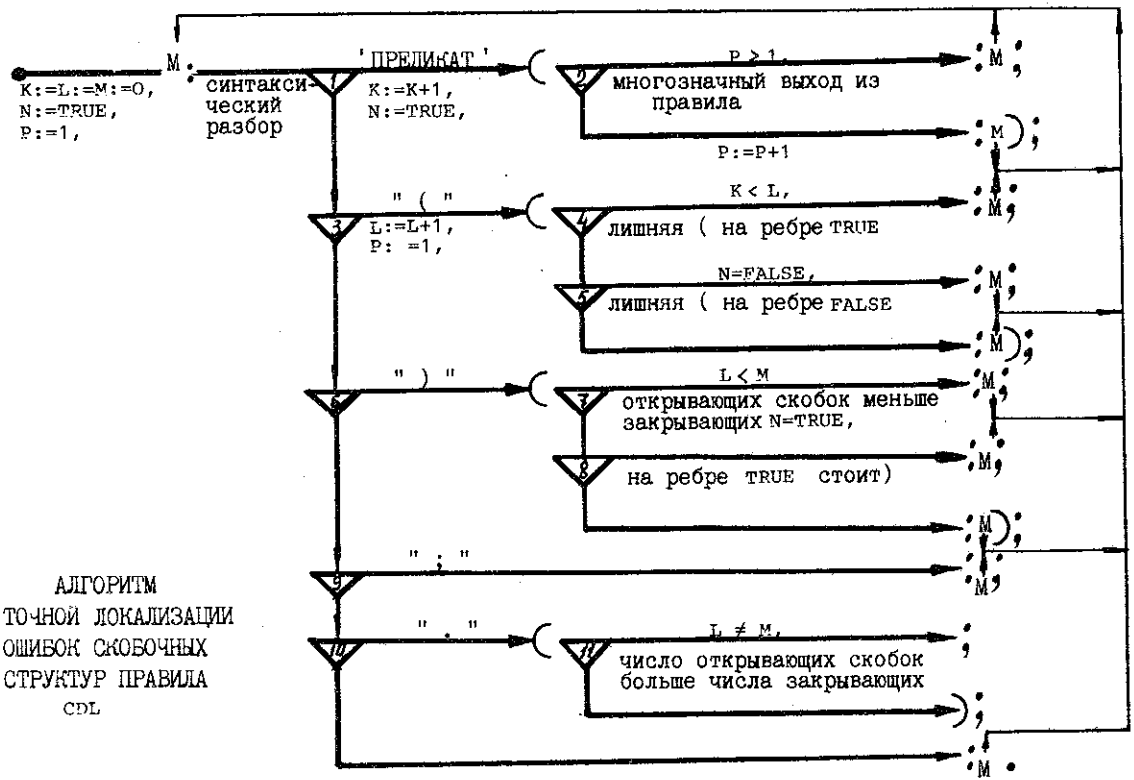
Тогда можно построить алгоритм строгой локализации ошибок при скобочной структуризации правил CDL .

Этот алгоритм изображен на рисунке 5.

Заключение

Описанная методика работы с двумерным изображением правил CDL была проверена на начинающем лаборанте. В течение первых четырех часов этот сотрудник сумел полностью разобраться в постановке задачи и методе ее решения, построил графы для 12 правил средней трудности и нашел все ошибки. В последующие 4 часа был составлен граф для самого длинного правила в трансляторе CDL и с использованием только что нарисованного графа был безошибочно восстановлен текст правила.

Психология программиста такова, что он при разработке алгоритма стремится разбить сложный алгоритм на частные алгоритмы. Трудности построения скобочных структур способствует этому разделению. В итоге результирующая программа состоит из большого



АЛГОРИТМ ТОЧНОЙ ЛОКАЛИЗАЦИИ ОШИБОК СКОБОЧНЫХ СТРУКТУР ПРАВИЛА CDL

Рис.5

числа правил, значительная часть из которых вызывается только в одном месте программы. Изображение графами правил CDL способствует написанию не разделенного на отдельные правила алгоритма. Неразделенные правила в CDL работают быстрее и занимают меньше места в памяти. При использовании транслятора CDL-2 необходимости в такой ручной оптимизации нет, так как этот транслятор автоматически устраняет правила, которые вызываются только в одном месте[6]. Авторы выражают благодарность А.Корнейчуку за ряд ценных советов, которые были учтены при подготовке публикации.

Литература

1. Вельбицкий И.В., Ходаковский В.Н., Шолмов Л.И. Технологический комплекс производства программ на машинах ЕС ЭВМ БЭСМ-6, М., "Статистика", 1980.
2. C.H.A.KOSTER, A COMPILER COMPILER. MR 127/71 NOV., STICHING MATEMATISH CENTRUM, AMSTERDAM.
3. C.H.A.KOSTER, USING THE CDL COMPILER COMPILER, COMPILER CONSTRUCTION-AN ADVANCED COURSE T.U.BERLIN.
4. Макаренкова А.Д., Назаров Ю.А., Хошенко А.А. CDL, инструкция для пользователей на БЭСМ-6 и CDC-6500 . ОИЯИ Б1-11-12214, Дубна, 1978.
5. Макаренкова А.Д., Назаров Ю.А., Хошенко А.А. Внедрение компилятора компиляторов CDL на ЭВМ БЭСМ-6. ОИЯИ Р11-12340, Дубна, 1978. "Программирование", № 3, М., "Наука", 1980.
6. Hans-Michael Stahl. CDL towards IBM/370-Assembler under OS/370. User's Guide.Verson 7.3., K.U. Nijmegen, The Netherlands.

Рукопись поступила в издательский отдел
12 декабря 1980 года.