Подход к реализации атрибутов в системе SynGT

Федорченко Людмила Николаевна,

старший научный сотрудник

Санкт-Петербургского института информатики и автоматизации РАН (СПИИРАН)

Россия, Санкт-Петербург

E-mail: LNF@iias.spb.su

В статье рассматривается атрибутный подход к реализации контекстных условий при построении синтаксического анализатора языка с использованием инструментального средства SynGT (Syntax Graph Transformations).

Ключевые слова: КСР грамматика, синтаксическая граф-схема, атрибуты, трансляции.

Введение

Современные системы построения трансляторов для корректного отображения транслируемого языка во внутреннее машинное представление используют информацию о языке, определяя его синтаксис и статическую семантику в виде специальной грамматики, которая кроме основной своей функции порождения предложений языка позволяет задавать трансляции, необходимые разработчикам.

В теоретических работах под синтаксисом понимают множество правил грамматики, а все остальное в определении языка относят к семантике. В этом случае семантику делят на две части: те свойства языка, которые можно проверить в процессе трансляции (компиляции), называют статической семантикой, а остальные свойства языка – динамической семантикой.

В данной работе будем полагать, что грамматика, определяющая тексты языка, называется синтаксисом, некоторые правила, относящиеся к правильному использованию слов языка, называются статической семантикой, тогда как под динамической семантикой понимается то, как может быть установлено I/O отображение, создаваемое программой в процессе исполнения.

Полагаем, что программа состоит из слов. В этом случае терминальные символы КС-грамматики будут словами языка.

В сравнении с естественными языками большинство языков программирования отличаются тем, что некоторые слова не имеют заранее определенного значения, или их значение определено частично. Необходимые атрибуты таких слов фиксируются только в программе. Например, идентификатор может обозначать либо тип константы или тип переменной, либо метку или операцию. В языке BASIC буква, за которой следует открывающая скобка, может обозначать либо одномерный, либо двумерный массив и т.д. Поэтому большинство языков программирования имеет встроенные механизмы проверки синтактико-семантических свойств языка для всех допустимых слов в программе.

В реальных компиляторах все синтаксические проверки осуществляются, как правило, с помощью таблиц, где регистрируются текущие атрибуты проанализированных слов (конструкций языка) при анализе слева направо по тексту программы.

В теории формальных языков (как части Computer Science) к настоящему времени создано достаточно много абстрактных формализмов для определения языков программирования. Работа

по созданию таких формализмов велась с 50-х годов прошлого столетия в 3-х направлениях:

- заменой формализма КС-грамматики на более мощную грамматику;
- встраиванием атрибутов и предикатов (проверок) в КС-грамматику;
- введением и пошаговой модификацией состояний преобразователя, построенного по исходной грамматике.

В *первом* направлении были сделаны несколько безуспешных попыток до того момента, как была придумана W-грамматика А. ван Вейнгаардена, которая использовалась в определении языка Algol 68 [11], [12], [13].

Во *втором* направлении важной работой стала статья Д. Кнута [7], в которой описана идея о том, что каждая синтаксическая конструкция имеет наследуемые и синтезированные атрибуты. Затем появилась аффиксная грамматика Костера [8,9,10], в которой идея с атрибутами Кнута была более развита, и которая стала теоретическим базисом для языка CDL(Compiler Definition Language) [10,14]. Функциональные грамматики тоже относится к этому направлению. Помимо Кнута похожий метод был предложен Гриффитсом (M.Griffits) [17].

Третье направление, которое основывается на работе Ледгарда (H.F.Ledgard)[15], было практически применено в работах Вильямса (M.H.Williams)[16] и др. В этом случае имеется таблица (или что-то подобное ей), содержащая текущие имена с текущими атрибутами, и есть множество функций, которые связаны с синтаксическими конструкциями для модификации состояния таблицы. В таком методе терминальные символы КС-грамматики имеют функции перехода из состояния в состояние. Атрибуты для синтаксических единиц более высокого уровня вводятся так же, как в грамматике А. ван Вейнгаардена. Такой метод дает более ясное разделение между синтаксисом и статической семантикой.

1. Атрибутный подход. Атрибутная техника была введена Д. Кнутом [7] с целью описания семантики языков, порождаемых классическими контекстно-свободными грамматиками Хомского [1]. Атрибутный подход (по Кнуту) предполагает построение построение дерева разбора входного предложения языка в данной КС-грамматике, а затем вычисление значений атрибутов в каждой вершине этого дерева по правилам, связанным с правилами грамматики, участвующими в разборе. На практике данный подход был реализован в ряде инструментальных систем построения компиляторов, таких как, например, YACC [6], Eli [4], CUP[18].

Начиная с 70-х годов, для определения синтаксиса КС-языков применяется их регулярная форма, – КСР-грамматики [3], RBNF-грамматики [21], EBNF-грамматики и их графическое представление,– синтаксические граф-схемы (аналог диаграмм Н.Вирта [2]). Каждому нетерминалу соответствует одна компонента диаграммы или одно правило КСР-грамматики, правая часть которого является обобщённым регулярным выражением относительно символов объединенного алфавита грамматики [3], [19], [20]. В объединённый алфавит грамматики входят алфавиты терминалов, нетерминалов, семантик и предикатов. Семантики и предикаты присутствуют в КСР-правилах наравне с терминальными символами, а в синтаксической граф-схеме (СГС) располагаются на дугах, соединяющих вершины, помеченные терминалами и нетерминалами грамматики.

Технологии, использующие КС-грамматики с атрибутами, обычно преобразуют данную грамматику в эквивалентную однозначную КС-грамматику. . За основу была взята методика построения синтаксических таблиц для языкового процессора, реализованная в инструментальной системе SynGT [3], использующая синтаксические граф-схемы и трансляционные КСР-грамматики

для определения трансляций. Далее кратко опишем метод спецификации трансляций на трансляционных КСР-грамматиках и его использование при синтаксически-управляемой обработке данных в инструментальной системе.

2. Применяемая методика. *Трансляция* из языка $L_1 \subset \Sigma_1^*$ в язык $L_2 \subset \Sigma_2^*$ определяется как отношение $\tau \subset L_1 \times L_2$[1]. На практике рассматривается обобщение этого определения, при котором трансляция определяет изменение состояния программы в зависимости от входного предложения.

В соответствии с требованиями [10], которым должно удовлетворять описание языка, способ спецификации трансляции должен давать следующее:

- возможность автоматического синтеза транслятора по спецификации;
- возможность анализа свойств входного языка;
- наглядность в описании синтаксиса и семантики входного языка.

Привлекательность КС-грамматик для анализа языка, их простота (правила имеют вид $A \to \alpha$, где - αцепочка символов из объединенного алфавита грамматики), позволяющая создавать эффективные анализаторы, породили огромное количество классов и подклассов грамматик, с различной степенью ограничений на языки. Наиболее сильные ограничения на языки (LL(k)), LR(k), LALR, SLR) [1] позволяют строить анализаторы, имеющие линейную сложность разбора в зависимости от длины входного предложения.

Способ спецификации трансляций в системе SynGT позволяет определять контекстно-зависимые языки. Это достигается введением предикатов, ограничивающих выбор альтернатив при анализе входного предложения в зависимости от контекста. Состояние контекста изменяется семантиками, вводимыми непосредственно в текст грамматики или графа-схемы и обрабатываемыми наравне с терминальными символами. Предварительная обработка грамматики помимо выполнения различных эквивалентных преобразований (регуляризация) транслирует предикаты и семантики в предикатные функции и семантические процедуры без параметров, создавая вычислительную надстройку. В этом случае атрибутные значения используются для параметризации семантических процедур и предикатных функций.

Следует отметить, что у Кнута семантика входного предложения определяется по его контекстно-свободной синтаксической структуре [7]. В случае параметризации семантик в SynGT атрибуты используются, как классические атрибуты Кнута. Но в методе Кнута правила вычисления атрибутов применяются после того, как дерево разбора входной цепочки в КС-грамматике уже построено. В системе SynGT семантика входного предложения определяется его контекстно-зависимой синтаксической структурой, что означает принципиальную невозможность отделить во времени процесс анализа от вычисления семантики. Так или иначе, в обоих случаях атрибуты используются для задания потоков данных на уровне спецификации синтаксиса языка.

При использовании атрибутов для параметризации предикатов принят метод параметризации нетерминалов аффиксами в системе CDL3 К. Костера[14]. Таким образом, атрибутная КСР-грамматика становится двухуровневой [10]. Известными представителями этого класса грамматик, помимо аффиксных грамматик К. Костера, являются также грамматики А. ван Вейнгаардена [13].

Как правило, используемые на практике искусственные языки часто не являются контекстно-свободными. Применение двухуровневых грамматик, мощность которых достаточна для описания рекурсивных множеств, позволяет специфицировать синтаксис такого языка в рамках одного формализма.

В общем случае в SynGT технологии используется синтаксический анализ типа top-down.

Поэтому основным ограничением на класс грамматик является отсутствие леворекурсивных правил. Алгоритм удаления левой рекурсии в KCP-грамматиках подробно дан в ряде работ [19], [20].

Введение атрибутов в такой механизм анализа нуждается в разработке способа передачи контекстной информации только между процедурами анализа отдельных конструкций в процессе просмотра входного предложения.

Процесс анализа идет по нескольким параллельными маршрутами в синтаксической граф-схеме, представляющей KCP-грамматику. Для того, чтобы сохранить детерминированный характер этого процесса, на KCP-грамматику накладываются ограничения, которые учитываются при введении атрибутов.

Неформально можно провести аналогию используемого механизма анализа с анализом методом рекурсивного спуска. Каждое правило KCP- грамматики, определяющее нетерминал A, отображается в описание процедуры A. Правая часть этого правила отображается в тело процедуры A, которое реализует проверку условий (бесконтекстных или контекстных), в зависимости от которых планируется соответствующая ветвь разбора. Бесконтекстное условие состоит в проверке того, что текущий входной символ равен соответствующему терминальному символу в правой части правила для A. Если это условие выполняется, то планируется продвижение по входному тексту и переход к следующей позиции в правой части правила для A. Если в текущей позиции правила для нетерминала A находится нетерминал B, то планируется вызов процедуры B, являющейся образом правила для нетерминала B.

Контекстные условия реализуются соответствующими предикатными функциями, вызовы которых явно обозначены в рассматриваемом правиле. Вызовы семантических процедур в правой части правила также должны быть представлены явно.

Метод рекурсивного спуска в последовательной среде в общем случае сложен при реализации “отката”(back-tracking), то есть отмены семантических действий из-за нарушения бесконтекстных или контекстных условий на текущей ветви разбора. Поэтому применяемый в SynGT-технологии подход равносилен отображению нескольких правил в одну процедуру, что обеспечивает одновременное построение нескольких параллельных вариантов вывода с отбрасыванием недопустимых вариантов по ходу анализа. (см определения состояния вершины в СГС и переходного состояния распознавателя).

Интерпретация метода рекурсивного спуска в терминах множества конечных автоматов [3], взаимодействующих в общей операционной среде, дает ключ к пониманию подхода к реализации атрибутов в SynGT. Подобный подход успешно применяется в системе CDL с аффиксами.

Продолжив аналогию с процедурами, можно каждому правилу сопоставить набор входных и выходных формальных параметров и локальных переменных. Элементы этого набора назовем формальными атрибутами. Для передачи информации в процедуры, соответствующие правилам грамматики или в процедуры, реализующие семантики и предикаты, можно применять наборы фактических параметров. Элементы таких наборов назовем фактическими атрибутами.

Таким образом, атрибуты на уровне грамматики описывают лишь порядок передачи контекстной информации между элементами конкретного алгоритма трансляции. Основной их задачей является доставка данных при реализации семантик и предикатов через параметры.

Поскольку возможные варианты продолжения анализа входной цепочки “тянутся” параллельно и фиксируются в списках магазинов, то ограничения, накладываемые на KCP-грамматики в SynGT, допускают синтаксически неоднозначные грамматики при соблюдении локальной семантической однозначности с учетом детерминизма анализа. С другой стороны, исполнение семантических действий и, таким образом, изменение контекста и выполнение

трансляции происходит одновременно с анализом входного предложения. Возможна ситуация, когда выполнение некоторого семантического действия зависит от информации, целиком доступной только к моменту окончания просмотра всего предложения языка. В этом случае такое действие может быть также отложено до стадии просмотра параллельных магазинов и вызова семантики в результате дополнительного анализа. На этом этапе происходит прохождение по состояниям анализатора в обратном порядке, вызов отложенных семантик и более точное определение синтаксической структуры входной цепочки с помощью предикатов.

При выполнении отложенных действий важно восстановить контекст, в котором эти действия были отложены. Иными словами, необходимо обеспечить передачу контекстной информации в таких особых случаях.

3. Неформальное определение атрибутной трансляции. Со всяким вхождением нетерминала в левые части правил связывается набор формальных атрибутов, а со всяким вхождением нетерминала, семантики или предиката в правые части правил набор фактических атрибутов. Формальные атрибуты делятся на наследуемые, синтезируемые и локальные атрибуты. Фактические атрибуты – на наследуемые и синтезируемые.

Предлагаемое определение атрибутной трансляции близко к алгоритму Эрли [5], применяемому для анализа произвольных КС-грамматик. Метод рекурсивного спуска и преобразование правил грамматики в процедуры с параметрами аналогичен аффиксному подходу К. Костера. Основными отличиями являются учет контекстных условий, задаваемых предикатами, исполнение семантических действий, задание правил грамматики с помощью обобщённых регулярных выражений и применение атрибутов для передачи информации.

На основе алгоритма работы процессора и определения атрибутной трансляции выявляются и формулируются ограничения на атрибутную спецификацию трансляции, которые сводятся в основном к требованию отсутствия неопределённых значений (свойство детерминизма процессора).

Это требование вытекает из общих ограничений на класс KCP-грамматик, применяемым в SynGT для получения детерминированных процессоров линейной сложности.

Атрибуты (как формальные, так и фактические) подразделяются на атрибуты прямого просмотра и отложенные, а некоторые локальные атрибуты могут определяться как двусторонние. Значения двусторонних атрибутов вычисляются при просмотре входного предложения и могут быть использованы в случае отложенности вычислений (когда просматриваются стеки в обратном порядке).

В ходе исполнения алгоритма трансляции процессор на отдельных участках входной цепочки ведет себя, как детерминированный конечный автомат (ДКА). В некоторых случаях работа такого ДКА приостанавливается, и управление передается другому ДКА для анализа некоторой подконструкции. После окончания обработки подконструкции управление возвращается исходному автомату. Каждому такому ДКА соответствует одно или несколько правил грамматики. Значения формальных атрибутов, связанных с нетерминалами, стоящими в левых частях правил, определяют локальный контекст автомата. Значения двусторонних атрибутов считаются вычисленным к моменту достижения ДКА конечного состояния. В этот момент эти значения должны быть записаны с тем, чтобы быть восстановленными в случае отложенных проверок. Запись происходит вместе с номером состояния, в которое осуществляется возврат управления, так называемого, возвратного состояния (второго элемента в списках пар состояний) (см. пример в [3]).

Передачи управления между конечными автоматами на отложенном просмотре согласованы и происходят при принятии возвратного состояния (второе состояние в паре (переходное, возвратное)). В этот момент формируется новый локальный контекст из значений отложенных

атрибутов, значения двусторонних атрибутов восстанавливаются из соответствующей записи.

4. Модификация метода Кнута. При построении процессора языка его управляющая таблица дополняются таблицей контекстов, которые используется только на этапе синтеза процессора, и таблицей цепочек фактических атрибутов. Таблица контекстов связывает с каждым нетерминалом цепочку формальных атрибутов. Элементом такой цепочки является пара, состоящая из имени и класса атрибута. Таблица цепочек фактических атрибутов связывает с элементами правых частей правил наборы фактических атрибутов. Элементами такого набора являются следующие пары:

1) индекс формального атрибута, соответствующего нетерминалу, стоящему в левой части рассматриваемого правила;

2) класс фактического атрибута.

При проведении сравнения атрибутного подхода с использованием технологии в SynGT без атрибутов выявляются следующие основные преимущества атрибутных спецификаций:

а) потоки данных задаются с использованием атрибутов независимо для каждого KCP-правила.

б) программисту-разработчику спецификации трансляции не приходится разрабатывать свой механизм для передачи данных между реализациями семантик и предикатов; такой механизм предоставляется самим атрибутным алгоритмом трансляции;

в) обмен данных между семантиками и предикатами происходит через фактические параметры;

г) предлагаемый способ передачи информации в атрибутном подходе осуществляется через параметры процедур, связанных с синтаксической структурой входной цепочки.

Представленные в работе [3] ограничения на класс грамматик задают алгоритм проверки таких условий на уровне синтаксической граф-схемы.

Использование атрибутов во вспомогательных понятиях: вспомогательные понятия (новые нетерминалы, которые появляются в результате эквивалентных преобразований в SynGT и не являющиеся первоначальными грамматическими конструкциями), см. [3] в спецификациях на языке обобщённых регулярных выражений применяются для макроподстановок на этапе построения синтаксической граф-схемы. Использование атрибутов в спецификациях, таким образом, должно отразиться и на применении вспомогательных понятий. Эта проблема решается на этапе применения макроподстановки с помощью переименования атрибутов, т.к. в процессе регуляризации KCP-грамматики в результате подстановок «теряются» вхождения нетерминалов.

Контроль типов значений атрибутов, может быть реализован на этапе построения управляющих таблиц. Кроме того, возможно осуществлять дополнительный контроль во время исполнения алгоритма трансляции, применяя предикаты. Важно отметить, что контроль времени исполнения невозможен в случае классического подхода [10].

Описанный в [3] алгоритм оптимизации процессоров (минимизация числа состояний управляющей таблицы) базируется на вычислении отношения эквивалентности для различных элементов управления процессора. Это отношение может быть модифицировано с учетом введения атрибутов. Сами алгоритмы оптимизации при этом не изменяются.

Заключение

Впервые атрибутная техника применяется непосредственно для контекстно-свободных грамматик в регулярной форме при построении средств синтаксически управляемой обработки данных в контексте следующих особенностей алгоритма трансляции:

- атрибуты вычисляются в процессе построения дерева вывода;
- дерево вывода строится с учетом контекста, определяемого значениями атрибутов;
- атрибуты применяются для контекстно-зависимого разрешения семантической неоднозначности трансляции.

Полученные результаты развивают технологию синтаксически-управляемой обработки данных с помощью инструментального комплекса SynGT, улучшая условия разработки и надежность информационных систем, создаваемых с ее использованием. Прототип системы с атрибутами реализован на платформе .NET. В качестве основы для данной реализации был взят код инструментального комплекса SynGT, разработанного в СПИИРАН.

Использование разработанного прототипа предполагает следующее:

- применение подхода с атрибутами улучшает условия разработки семантической части спецификации трансляции, избавляя разработчика от необходимости реализации своего метода передачи контекстной информации, согласованного с синтаксическим механизмом;
- использование семантик и предикатов с параметрами делает структуру семантической части спецификации более прозрачной и модульной.

Литература

1. Ахо А., Сети Р., Ульман Дж. Компиляторы: принципы, технологии и инструменты. Пер. с англ. М., 2001.

2. Вирт Н. Алгоритмы+структуры данных=программы. Пер. с англ. М., 1984.

3. Федорченко Л. Н. Книга – Регуляризация контекстно-свободных грамматик. Saarbrucken, Germany: LAP LAMBERT Academic Publishing GmbH & Co. KG Dudweiler Landstr. 99, 66123, 2011. 180 с.

4. Gray R.W., Heuring V.P., Levi S.P. et al. Eli: a complete, flexible compiler construction system // Communications of ACM, Vol. 35, №. 2, 1992. P. 121-130.

5. Early J. An Efficient Context-Free Parsing Algorithm // Communications of the ACM, Vol. 13(2), 1970. P. 94-102.

6. Johnson S.C. Yacc yet another compiler compiler. Computer Science Technical Report 32. AT&T Bell Laboratories, Murray Hill, N.J., 1975.

7. Knuth D.E. Semantics of context-free languages // Mathematical Systems Theory 2:2. 1968, pp. 127-145.

8. Koster C.H.A. On the Construction of ALGOL-Procedures for Generating, Analyzing and Translating Sentences in Natural Languages. Report MR72, Mathematisch Centrum, Amsterdam, 1965.

9. Koster C.H.A. Affix Grammars // In: Proceedings of IFIP Conference on ALGOL 68 Implementation. 1970. Munich, pp. 95-109.

10. Koster C.H.A. Affix Grammars for Programming Languages // In: Attribute Grammars, Applications and Systems. International Summer School SAGA. 1991. Prague.

11. A. van Wijngaarden A. Orthogonal Design and Description of a Formal Language. Report MR76, Mathematisch Centrum, Amsterdam, 1965.

12. A. Van Wijngaarden et al.:"Report on the algorithmic language ALGOL 68" Numerische Mathematik 14, pp. 79–218 (1969) Springer-Verlag Berlin.

13. Revised report on the algorithmic language Algol 68 // ACTA Informatica 5 pp.1–236. 1974.

14. Koster C.H.A., et all. CDL3 manual \\http://www.cs.ru.nl/~kees/vbcourse/ ivbstuff/cdl3.pdf .

15. Ledgard H.F. “Production system or can we do better than BNF?” CACM 1974/2. P.94–102.

16. Williams V.H. “Static semantics features of Algol 60, and BASIC”. The Computer Journal. Vol. 21. No. 3. P. 234–242.

17. Griffiths M. “Relationship between definition and implementation of language”. //Advanced courses on software engineering. Lecture Notes in Economics and Math Syst. Springer-Verlag 1973.

18. Федорченко Л.Н, Извлечение крайней рекурсии из КСР-грамматики в системе SynGT, Труды СПИИРАН, вып.1,т. 1.– СПб: СПИИРАН, 2002, с.350–359.

19. Федорченко Л.Н. О регуляризации контекстно-свободных грамматик. // Изв. вузов. Приборостроение, 2006. Т.49, №11, с.50-54.

20. Лукичёв А.С. Использование атрибутов в технологии SYNTAX // Вест. Петерб. ун-та. Серия 1. Вып.2. СПб. 2005