ФОРМАЛЬНЫЙ ЯЗЫК КАК ИНТЕРФЕЙС ВЗАИМОДЕЙСТВИЯ

Д. И. Черемисинов

Объединенный институт проблем информатики НАН Беларуси
Минск, Беларусь
e-mail: cher@newman.bas-net.by

Рассматривается модель взаимодействия элементов программной системы, отличающаяся от модели контрактного программирования тем, что она основана на коммуникации репрезентативными сообщениями, принадлежащими формальному языку. Предлагается модель взаимодействия программных компонент, использующих в качестве платформы коммуникации формальную операцию подстановки языков. Подстановка языков формализуется как регулярное выражение с побочным эффектом. Описывается схема трансляции регулярных выражений в программы на языке С.

Ключевые слова: модель взаимодействия элементов программной системы; интеграция САПР СБИС; конвертация формальных языков.

FORMAL LANGUAGE AS INTERACTION INTERFACE

D. I. Cheremisinov

United Institute of Informatics Problems
National Academy of Sciences of Belarus
Minsk, Belarus

The model of interaction of elements of a software, which differs from the model of contract programming in that it is based on a representative messaging communication is considered. The model of interaction between software components is the formal operation of languages substitution. In turn, language substitution formalized as a regular expression with a side effect. The principle of regular expressions compilation into the program in C is described.

Keywords: interaction model of software components; EDA tools integration; formal language conversion.

ВВЕДЕНИЕ

Область компьютерных наук с момента своего образования столкнулась с проблемами, связанными со сложностью программных систем. Сложность программных систем зависит от многих факторов. Популярным методом проектирования *программного обеспечения* сложной архитектуры является контрактное программирование (design by contract (DbC), contract-based programming). Основная идея этого метода проектирования состоит в том, что проектировщик должен определить *формальные*, точные и верифицируемые спецификации интерфейсов для компо-

нентов системы [1]. Предполагается, что компонентами системы являются переменные абстрактного типа. Абстрактный тип данных предоставляет для работы с элементами этого типа определенный набор функций, а также возможность создавать элементы этого типа при помощи специальных функций. В контрактном программировании используется сервисная модель взаимодействия элементов программной системы. На интерфейс связи клиентских компонент, вызывающих операции в серверных объектах, накладываются предварительные условия, указанных в требованиях для этой операции. Смысл взаимодействия состоит в воздействии агента на поведение сервера.

Взаимодействие имеет отношение к способу, которым компоненты программной системы обменивают информацией. Формализация понятия взаимодействия отталкивается от работы Шеннона по теории связи: *взаимодействие – это способ передать сообщение от одного отправителя нескольким получателям через среду передачи.* В широком понимании обмен информацией – это коммуникация, обмен сообщениями, т. е. – общение. Часто коммуникацией считается исключительно вербальная разновидность (устная или письменная), так как она является для человека основной, универсальной формой общения, кодирования и передачи информации. Если сообщение не имеет смысла влияния на поведение партнера по коммуникации, а просто несет информацию, то такие сообщения называются репрезентативами [2]. Коммуникация репрезентативами широко используется при взаимодействии между программами.

В этой работе рассматривается модель взаимодействия элементов программной системы, отличающаяся от модели контрактного программирования тем, что она основана на коммуникации репрезентативными сообщениями, принадлежащими формальному языку.

ПЛАТФОРМА КОММУНИКАЦИИ САПР СБИС

В настоящее время фабрики, выпускающие СБИС, предоставляют возможность использовать их производственные линии для выпуска заказных микросхем. Однако проектирование заказных СБИС невозможно без доступа к инструментам САПР. Имеется ряд коммерческих компаний, разрабатывающих САПР СБИС, и электронные проектные компании имеют возможность выбора инструментов САПР от различных поставщиков. Теперь задачей проектировщика САПР стала интеграция инструментов САПР от различных поставщиков в однородную систему.

В этой системе компоненты (инструменты САПР) взаимодействуют, обмениваясь сообщениями, представляющими собой описание проектируемого устройства на языке проектирования аппаратуры или в специальном обменном формате. Интерфейс такого взаимодействия задается конвертаций одного формального языка в другой. Задачи языковых преобразований похожи на задачи разработки компиляторов [3]. Однако, в отличие от языков программирования, форматы данных, используемые для системной интеграции программ САПР СБИС, – это языки регулярного типа. Программы конверторов, построенные с помощью инструментов построения компиляторов, неоправданно сложны, так как содержат поддержку рекурсии, необходимую для разбора контекстно-свободных языков. Предлагается модель взаимодействия программных компонент, использующих в качестве платформы коммуникации формальную операцию подстановки языков.

ПОДСТАНОВКА ЯЗЫКОВ

Формально языковую подстановку можно рассматривать как упорядоченную тройку (L_1, L_2, ϕ), состоящую из двух языков L_1, L_2 и отображения $\phi: L_1 \to L_2$ [4]. Опе-

рация применения языковой подстановки состоит в построении слова W из слова V. Если существует $x \in L_1$ и -вхождение в V, то W получается заменой в V вхождения X словом $y = \phi(x) \in L_2$. Если $X \in L_1$ не содержится в V, то подстановка (L_1, L_2, ϕ) к V не применима.

Регулярным выражением [5] называется формула алгебры Клини К. Язык регулярных выражений составляет множество формул алгебры К. Элементами К являются строки символов фиксированного алфавита Σ, сигнатуру операции составляют константы 0 и 1 и операции объединения (∨), конкатенации (.) и звезда Клини (*). В стандартной интерпретации регулярное выражение s задает язык $L(s)$ алфавита Σ; 0 интерпретируется как пустая строка, 1 – как язык, состоящий из пустой строки, операция $a \vee b$ – как язык $L(a \vee b) = L(a) \cup L(b)$, операция $a.b$ – как язык $L(a.b) = \{xy \mid x \in L(a), y \in L(b)\}$ и звезда Клини $a*$ – язык $L(a*)$ всех строк, полученных конкатенацией нуля или более строк из $L(a)$.

Состоянием разбора слова v называется упорядоченная пара (v, i), где $v \in \Sigma^*$, а i – целое число; $0 \leq i < |v|$,. $|v|$ – длина слова v. Состояние разбора (v, i) задает представление слова v в виде конкатенации $br = v, |b| = i$.

Пусть f_i и p_i – символы некоторых функций и предикатов на состояниях разбора. Частичная функция преобразования состояний разбора строки f – образец [6], если для любых состояний $\alpha = (w, j), \beta = (v, i)$ $\alpha = f(\beta)$ $и$ $w = v, j \geq i$. Пусть $p(f, \alpha)$ – предикат, обозначающий утверждение, что образец f определен на состоянии $\boldsymbol{\alpha}$. Если образцы $f_1 = f_2$, то и $p(f_1) = p(f_2)$.

Будем использовать инфиксную форму записи операции # применения образца α к некоторому состоянию β строки: $f\#\beta = \alpha$ и условные выражения Мак-Карти [7]. Введем следующие операции над образцами.

Катенацией fg образцов f и g называется операция, определяемая условным выражением $fg\#\alpha = p(f, \alpha) \rightarrow g\#(f\#\alpha)$. Альтернация $f \vee g$ образцов f и g определяется условным выражением $f \vee g\#\alpha = p(f, \alpha) \rightarrow (f\#\alpha); p(g, \alpha) \rightarrow (g\#\alpha)$. Итерация образца f^* определяется через степень следующим выражением с бесконечным числом членов

$f* \#\alpha = \neg p(f_1, \alpha) \rightarrow \alpha;\ \neg P(f_2, \alpha) \rightarrow f_1\#\alpha; \ldots; \neg P(f_i, \alpha) \rightarrow f_{i-1}\#\alpha; \ldots$. Степень определяется рекурсивно выражениями $f^1\#\alpha = f\#\alpha$ и $f^n\#\alpha = f\#(f^{n-1}\#\alpha)$.

Веденные операции образуют алгебру образцов, элементами которой в отличие от алгебры Клини являются состояния разбора – пары (v, i), где $v \in \Sigma^*$, а i – целое число. Нетривиальными константанами этой алгебры являются примитивные образцы, распознающие вхождение слов, состоящих из единственного символа алфавита Σ. Константа 1 этой алгебры – тождественный образец – обладает следующими свойствами: если f образец, то $1 *= 1, 1f = f, f1 = f, 1 \vee f = 1, f \vee 1 = g$ такой *(всюду определенный)*, что *на области определения* f. $g = f$, *в остальной области* $g = 1$.

Пара состояний разбора α и $\beta = f(\alpha)$ задают представление слова v в виде axc, т. е. образец f распознает слова x *как часть слова* v. Множество всех слов, распознаваемых образцом f, называется языком $L[f]$, распознаваемым образцом, т. е. образец решает задачу анализа текста.

В отличие от традиционно рассматриваемой в теории формальных языков задачи распознавания принадлежности языку заданного текста, задачей анализатора языка является построение структуры анализируемого текста, выполняя его разбор. Алгоритм распознавания принадлежности имеет форму грамматики языка. Для построения анализатора в грамматику нужно добавить действия по формированию структуры

данных, представляющей результат анализа. В рассматриваемом случае грамматика задана в виде регулярного выражения. В алгоритме анализатора действия по распознаванию участков текста и действия построения результатов разбора чередуются.

Сопоставим с образцом f, распознающим язык $L[f]$, *однозначную функцию* $\phi: L[f] \to L_2$.Такой образец называется образцом с побочным эффектом. В информатике функция или выражение программы имеет побочный эффект (side effect), если вызов конструкции изменяет состояние вызывающей программы [8]. Преобразование $\beta = f(\alpha)$состояния разбора составляет основной эффект образца f.

Побочным эффектом образца может служить другой образец с побочным эффектом. Образцы с побочным эффектом являются классом рекурсивных функций. Так как рекурсивные функции являются одной из алгоритмических моделей, то вследствие тезиса Черча [9] образцы с побочным эффектом – это алгоритмически полная система. Следствием алгоритмической полноты языка регулярных выражений с побочным эффектом является возможность распознавания языка любого типа в классификации Хомского, а не только класса регулярных языков.

Алгебра образцов с побочным эффектом может быть построена модификацией интерпретации тождественного образца – константы 1 алгебры. Достаточно допустить, чтобы только тождественные образцы имели побочный эффект, т. е. вести нужное количество тождественных образцов, различающихся побочным эффектом.

Можно считать, что образцы с побочным эффектом задают некоторую операцию, применимую к анализируемому слову только в том случае, если в этом слове имеется состояние разбора, в котором образец применим. Таким образом, образец с побочным эффектом задает алгоритм построения слова y на основе информации о системе составляющих анализируемого слова *x*. Образец с побочным эффектом представляет собой нормальный алгоритм Маркова [10]. Конструирование этого алгоритма заключается во «встраивании» операций построения слова y в процедуру анализа x. Для этого в образец, задающий процедуру анализа, включаются тождественные образцы с побочным эффектом. Операции, представляющие побочный эффект этих образцов, являются базисом алгоритмического разложения операции вычисления $\varphi(x) = y$.

Регулярное выражение с семантикой алгебры образцов можно интерпретировать как текст программы, реализующей алгоритм применения этого образца, и выраженной на особом языке программирования. Базовый набор операций этого языка составляют операции применения примитивных образцов, а функциональные формы служат средством задания последовательности применения этих базовых операций, так как условные выражения, являющиеся определениями функциональных форм, суть не что иное, как указания о порядке применения операндов.

Платформу коммуникации составляют язык регулярных выражений и транслятор – это система программирования образцов. Транслятор этой системы представляет собой препроцессор компилятора C, преобразующий обозначения образцов в программы на языке C. В программе, построенной этим препроцессором, для представления состояния строк используются указатель $char * A$. Шаблон примитивного образа задается выражением $int\ B = *(A++)\hat{}n$; где n – распознаваемый символ. Катенация задается шаблоном h if(! B){ g }. Альтернация задается шаблоном

$$P.push\_back(A);\ h\ if(B)\{A = P.pop\_back();\ g\}\ else\ \{P.pop\_back();\ B = 1;\}.$$

Итерация задается шаблоном $do\{P.push\_back(A);\ if(!*A)\{B = 0; break;\}\ h\ B = (!B);\ \}while(!B);\ if(!B)B = -1; else\{A = P.pop\_back(); B = 0;\}.$

ЗАКЛЮЧЕНИЕ

Образцы алгебры образцов одновременно являются функциональной моделью операции анализа системы составляющих и задают последовательность выполнения примитивных образцов в процессе анализа распознаваемого слова. Применение побочного эффекта в образцах делает регулярные выражения таким же универсальным формализмом, как и грамматики. Реализация операции анализа, основанная на применении образцов предлагаемого типа, потенциально более эффективна, чем алгоритм грамматического разбора, так как в этом случае имеется возможность управлять последовательностью применения подстановок. Представление подстановки образцами более естественно, особенно в простых случаях, по сравнению с атрибутными грамматиками, так как, указывая атрибуты для вычисления $\varphi(x) = y$, приходится учитывать особенности алгоритма грамматического разбора.

Система программирования образцов успешно использовалась для построения ряда конверторов форматов [11].

БИБЛИОГРАФИЧЕСКИЕ ССЫЛКИ

1. Mitchell R., McKim J. Design by Contract: by example – Addison-Wesley. Boston : MA, 2002.
2. Остин Дж. Слово как действие // Новое в зарубежной лингвистике. М., 1986. Вып. 17.
3. Ахо А. В., Хопкрофт Д. Э., Ульман Д. Д. Построение и анализ вычислительных алгоритмов М. : Мир, 1979.
4. Черемисинов Д. И. Программирование подстановок языков // Программирование. 1981. № 5. С. 30–37.
5. Фридл, Дж. Регулярные выражения. СПб. : Питер, 2001.
6. Aho A. Algorithms for finding patterns in strings // Handbook for theoretical computer science, MIT Press. V. A., 1990. P. 257–300.
7. McCarthy J. Recursive function of symbolic expressions and their computation by machine, part 1 // Comm. ACM 1960, Vol. 3, № 4. P. 184–195.
8. Себеста Р. У. Основные концепции языков программирования = Concepts of programming languages. 5-е изд. М. : Вильямс, 2001. С. 282–284.
9. Мальцев А. И. Алгоритмы и рекурсивные функции. М. : Наука, 1986.
10. Марков А. А. Теория алгорифмов // Тр. МИАН СССР. Т. 42. М.; Л. : Изд-во АН СССР, 1954. С. 3–375.
11. Черемисинов Д. И. Перепроектирование ПЛИС на основе трансформации моделей // Проблемы разработки перспективных микро- и наноэлектронных систем – 2014. Сб. тр. / под общ. ред. акад. РАН А. Л. Стемпковского. М. : ИППМ РАН, 2014. Ч. 1. С. 25–30.