УДК 65.011.56:621.9

ФОРМАЛИЗАЦИЯ ОБЪЕКТНО-ОРЕНТИРОВАННЫХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Д.т.н. И.Ш. Невлюдов[1], д.т.н. А.А. Андрусевич[1], к.т.н. В.В.Евсеев[1], к.т.н. С.С. Милютина[1],Я.О.Замирец[2]

1. Харьковский национальный университет радиоэлектроники

2. Государственное предприятие Научно-исследовательский технологический институт приборостроения, г. Харьков

В статье приведены основные определения для формализации объектно-ориентированных языков программирования для разработки автоматизированной системы проектирования технического задания.

У статті наведено основні визначення для формалізації об'єктно-орієнтованих мов програмування для розробки автоматизованої системи проектування технічного завдання.

Main determinations for object-oriented programming languages formalization for technical task CAD system are described.

Ключевые слова: *язык программирования, алфавит, формальное описание, команда, грамматика.*

Введение

Бурное развитие программирования приводит к необходимости создания автоматизированных систем для проектирования технического задания. Такие системы позволяют на ранней стадии разработки программного продукта учесть пожелания заказчика как по интерфейсу пользователя, так и по функциональным возможностям разрабатываемого программного продута.

Проведенные исследования [1-4] показали необходимость разработки таких систем, которые позволят рассчитать трудоёмкость, сроки проектирования, а также стоимость программного продукта на этапе разработки технического задания.

В ходе анализа выявлено, что систем, которые выполняют все перечисленные функции, не существует. Есть только локальные модули, обеспечивающие выполнение лишь отдельной задачи.

Одной из ключевых задач для систем является автоматическая генерация программного кода на базе разработанного интерфейса пользователя и функций программного обеспечения, которые указаны в техническом задании. В связи с этим необходимо формализовать языки высокого уровня программирования.

Формализация основных определений объектно-ориентируемых языков на базе теорий формальных языков

На базе проведенных исследований [1-4] определим следующие понятия в соответствии с правилами написания программного кода для объектно-орентированных языков [5].

Определение 1. Будем называть натуральными числами неотрицательные целые числа. Множество всех натуральных чисел $\{0,1,2,...\}$ обозначается N.

Определение 2. Алфавитом называется конечное непустое множество. Его элементы называются символами (буквами).

На базе определения 2 внесем следующие алфавиты, необходимые и достаточные для описания основных команд для объектно-орентированных языков.

$\Sigma_{\text{en}} = \{a,b,c,...,z\}$ - алфавит из 26 прописных английских букв, на котором пишутся команды, процедуры объектно-орентированных языков;

$\Sigma_{EN} = \{A,B,C,...,Z\}$ - алфавит 26 заглавных букв. Объектно-орентированные языки имеют разную чувствительность к регистру Object Pascal не имеет, а C++ имеет;

$\Sigma_{1} = \{0,1,2,...,9\}$ - алфавит ммножества всех натуральных чисел;

$\Sigma_{\text{ru}} = \{а,б,в,...,я\}$ - алфавит из 33 прописных русских букв, в основном используются для прописания комментариев и пояснению в программном коде.

$\Sigma_{\text{symb}} = \{.,,,/,//,=,+,\#,",',(,),*,\&,\wedge,<,>,=>,<=,\$,\%,\backslash,|,?\}$ - алфавит символов, который необходим для записи математических выражений, препинаний.

Определение 3. Словом (цепочкой, строкой) (string) в алфавите Σ называется конечная последовательность элементов Σ.

Пример: Рассмотрим алфавит $\Sigma_{\text{en}} = \{a,\mathrm{b},\mathrm{c},...,\mathrm{z}\}$ следовательно определению 3 оператор for является словом в алфавите Σ_{en}.

Определение 4. Слово, не содержащее ни одного символа (то есть последовательность длины 0), называется пустым словом и обозначается ε.

Пример: Расмотрим простой цикл for i=5 then .. в следствии чего знак пробела является пустым словом (ε) и используется как разделитель между командами в объектно-орентированных языках и степень его применения зависит от общепринятого стандарта оформления программного кода.

Определение 5. Длина слова W, обозначаемая $|w|$, есть число символов в W, причём каждый символ считается столько раз, сколько раз он встречается в W.

Пример: Очевидно, $|\mathrm{println} = 7|$ и $|\varepsilon| = 0$.

Определение 6. Если x и y - слова в алфавите Σ, то слово xy (результат приписывания слова y в конец слова x) называется конкатенацией слов x и y. Иногда конкатенацию слов x и y обозначают $x \cdot y$.

Определение 7. Если x - слово и $n \in N$, то через x^n обозначается слово $\underbrace{x \cdot x \cdot x \cdot ... \cdot x}_{n-раз}$. По определению $\mathrm{x}^0 \leftrightarrow \varepsilon$ читается“ равно по определению”).

Пример: В соотвествий определению 7 запись, $a++ = a+^2 \leftrightarrow a(+)^2$.

Определение 8. Множество всех слов в алфавите Σ_{en} обозначается Σ^*_{en}.

Определение 9. Пусть слово x - префикс (начало) слова y (обозначение $x[y$), если $y = xu$ для некоторого слова u.

Пример Очевидно из определения 9, $\varepsilon[\#include,$, $\#[\#include$, $\#in[\#include$, $\#include[\#include$.

Определение 10. Пусть слово x - суффикс (конец) слова y (обозначение $x]y$), если $y = ux$ для некоторого слова u. Пример аналогичен опридению 9.

Определение 11. Пусть слово x - подслово (substring) слова y, если $y = uxv$ для некоторых слов u и v.

Определение 12. Пусть запись $|\omega|_a$ обозначает количество вхождений символа a в слово ω.

Пример: допустим $\Sigma_{en} = \{a,b,c,...,z\}$, то команда $|assert|_a = 1$, $|assert|_s = 2$ и $|assert|_z = 0$.

Определение 13. Допустим $L \subseteq \Sigma^*$, где L называется формальным языком над алфавитом Σ. Поскольку каждый язык является множеством, можно рассматривать операции объединения, пересечения и разности языков, заданных над одним и тем же алфавитом (обозначения $L_1 \cup L_2, L \cap L_2, L_1 - L_2$).

Пример: Множество $\{a,in,...,if\}$ является языком L над алфавитом $\Sigma_{en} = \{a,b,c,...,z\}$.

Определение 14. Пусть $L \subseteq \Sigma^*$, отсюда язык $\Sigma^* - L$, будем называть дополнением (complement) языка L относительно алфавита Σ. следовательно, из контекста ясно, о каком алфавите идёт речь, говорят просто, что язык $\Sigma^* - L$ является дополнением языка L.

Определение 15. Допустим $L_1, L_2 \subseteq \Sigma^*$. Тогда $L_1 \cdot L_2 \leftrightarrow \{xy \mid x \in L_1, y \in L_2\}$. Запись $L_1 \cdot L_2$ называется конкатенацией языков L_1 и L_2.

Пример: Если $L_2 \in \Sigma^*_{en}$ и $L_1 \in \Sigma^*_{sym}$ следовательно $L_1 = \{\#\}$ и $L_2 = \{include\}$, то конкатенацией языков $L_1 \cdot L_2 = \{\#include,\}$.

Определение 15. Пусть $L \subseteq \Sigma^*$. Тогда можно записать $L^0 \leftrightarrow \{\varepsilon\}$ они равнозначны из следствия определения 4, можно записать $L_n \leftrightarrow \underbrace{L \cdot L \cdot ... \cdot L}_{n-раз}$.

Определение 16. Итерацией (Kleene closure) языка L (обозначение L^*) называется язык $\bigcup_{n \in N} L_n$. Эта операция называется также звёздочкой Клини (Kleene star, star operation).

Пример: Определим $\Sigma = \{a,b\}$ и $L = \{aa, ab, ba, bb\}$, то следовательно из определения 16 $L^* = \{\omega \in \Sigma^* \mid\mid \omega \mid \text{делиться на } 2\}$.

Определение 17. Обращением или зеркальным образом (reversal) слова ω (обозначается ω^R) называется слово, составленное из символов слова ω в обратном порядке.

Пример: Если $\omega = baaca$, то $\omega^R = acaab$.

Определение 18. Пусть $L \subseteq \Sigma^*$. Тогда $L^R \leftrightarrow \{\omega^R \mid \omega \in L\}$ равнозначность записи следует из определения 17.

Определение 19. Пусть Σ_1 и Σ_2 - алфавиты. Если отображение $h : \Sigma_1^* \to \Sigma_2^*$ удовлетворяет условию $h:(x \cdot y) = h(x) \cdot h(y)$ для всех слов $x \in \Sigma_1^*$ и $y \in \Sigma_1^*$, то отображение h называется гомоморфизмом (морфизмом).

Докозательство Пусть h - гомоморфизм, то $h(\varepsilon) = \varepsilon$.

Замечение: Каждый гомоморфизм однозначно определяется своими значениями на однобуквенных словах.

Определение 20. Если $h : \Sigma_1^* \to \Sigma_2^*$ - гомоморфизм и $L \subseteq \Sigma_1^*$, то через $h(L)$ обозначается язык $\{h(w) \mid w \in L\}$.

Пример: Пусть $\Sigma = \{a,b\}$ и гомоморфизм $h : \Sigma^* \to \Sigma^*$ задан равенствами $h(a) = abba$ и $h(b) = \varepsilon$. Следовательно, тогда $h(\{baa, bb\}) = \{abbaabba, \varepsilon\}$.

Определение 21 . Если $h : \Sigma_1^* \to \Sigma_2^*$ - гомоморфизм и $L \subseteq \Sigma_2^*$, то через $h^{-1}(L)$ обозначается язык $\{\omega \in \Sigma_1^* \mid h(\omega) \in L\}$.

Пример: Рассмотрим алфавит $\Sigma = \{a,b\}$. Пусть гомоморфизм $h : \Sigma^* \to \Sigma^*$ задан равенствами $h(a) = ab$ и $h(b) = abb$. Тогда $h^{-1}(\{\varepsilon, abbb, abbab, ababab\}) = \{\varepsilon, ba, aaa\}$.

Определение 22. Порождающей грамматикой (грамматикой типа 0) (generative grammar, rewrite grammar) называется

$$G \leftrightarrow \langle N, \Sigma, P, S \rangle,$$

где N и Σ - конечные алфавиты, $N \cap \Sigma = \varnothing$, $P \subset (N \cup \Sigma)^+ \times (N \cup \Sigma)^*$, P - конечно и $S \in N$,

Σ - основной алфавит (терминальный алфавит), его элементы называются терминальными символами или терминалами (terminal),

N - вспомогательный алфавит (нетерминальный алфавит), его элементы называются нетерминальными символами, нетерминалами или переменными (nonterminal, variable),

S - начальный символ (аксиома) (start symbol).

$(\alpha,\beta)\in P$ - называются правилами подстановки, просто правилами или продукциями (rewriting rule, production) и записываются в виде $\alpha\to\beta$.

Определение 23. Пусть дана грамматика G. Запись $\varphi\underset{\mathrm{G}}{\Rightarrow}\psi$,если $\varphi=\eta\alpha\theta$, $\psi=\eta\beta\theta$ и $(\alpha\to\beta)\in P$ для некоторых слов α,β,η,θ в алфавите $N\cup\Sigma$ Тогда из контекста ясно, о какой грамматике идёт речь, вместо $\underset{\mathrm{G}}{\Rightarrow}$ можно упростить до записи $\Rightarrow$.

Пример: Пусть $\mathrm{G}=\langle\{\mathrm{S}\},\{\mathrm{a},\mathrm{b},\mathrm{c}\},\{\mathrm{S}\to acSbcS,cS\to\varepsilon\},\mathrm{S}\rangle$.Тогда $cSacS\underset{G}{\Rightarrow}cSa$

Определение 24. Если $\omega_0\underset{G}{\Rightarrow}\omega_1\underset{G}{\Rightarrow}\ldots\underset{G}{\Rightarrow}\omega_n$, где $n\geq 0$, то можем записать $\omega_0\overset{*}{\underset{\mathrm{G}}{\Rightarrow}}\omega_n$ (так как бинарное отношение $\overset{*}{\underset{\mathrm{G}}{\Rightarrow}}$ является рефлексивным, транзитивным замыканием бинарного отношение $\overset{*}{\underset{\mathrm{G}}{\Rightarrow}}$ определенного на множестве $(\mathrm{N}\cup\Sigma)^*$. При этом последовательность слов $\omega_0,\omega_1,\ldots\omega_n$ называется выводом (derivation) слова ω_n из слова ω_0 в грамматике G. Число n называется длиной (количеством шагов) этого вывода. В часности, для всякого слова $\omega\in(\mathrm{N}\cup\Sigma)^*$ имеет место $\omega\overset{*}{\underset{G}{\Rightarrow}}\omega$ так как возможен вывод длины 0.

Определение 25. Язык, порождаемый грамматикой G - это множество $L(G)\leftrightarrow\{\omega\in\Sigma^*\mid S\overset{*}{\underset{G}{\Rightarrow}}\omega\}$. В данной работе обозначем, что грамматика G порождаёт (generates) язык $L(G)$. Существенно, что в определение порождающей грамматики включены два алфавита - Σ и N. Это позволяет нам в определении 24 “отсеять” часть слов, получаемых из начального символа. А именно, отбрасывается каждое слово, содержащее хотя бы один символ, не принадлежащий алфавиту Σ.

Определение 26. Две грамматики эквивалентны, если они порождают один и тот же язык.

Пример: Грамматика $\mathrm{S}\to\mathrm{abS},\mathrm{S}\to\mathrm{a}$ и грамматика $T\to aU,\ U\to baU,\ U\to\varepsilon$ эквивалентны.

Определение 27. Контекстной грамматикой (контекстно-зависимой грамматикой, грамматикой непосредственно составляющих, НС-грамматикой, грамматикой типа 1) (context-sensitive grammar, phrase-structure grammar) называется порождающая грамматика, каждое правило которой имеет вид $\eta A\theta\to\eta\alpha\theta$, где $\mathrm{A}\in N$, $\eta\in(N\cup\Sigma)^*$, $\theta\in(N\cup\Sigma)^*$, $\alpha\in(N\cup\Sigma)^+$.

Определение 28. Контекстно-свободной грамматикой (КС-грамматикой, бесконтекстной грамматикой, грамматикой типа 2) (context-free grammar) называется порождающая грамматика, каждое правило которой имеет вид $\mathrm{A}\to\alpha$, где $A\in N$, $\alpha\in(N\cup\Sigma)^*$.

Пример: Грамматика $\mathrm{S}\to\mathrm{ASTA}$, $S\to AbA$, $A\to\alpha$, $bT\to bb$, $AT\to UT$, $UV\to TV$, $TV\to TA$ является контекстной, но не контекстно-свободной (последние пять правил не имеют требуемого вида).

Определение 29. Линейной грамматикой (linear grammar) называется порождающая грамматика, каждое правило которой имеет вид $\mathrm{A}\to\mathrm{u}$ или $A\to uB\nu$, где $\mathrm{A}\in N$, $u\in\Sigma^*$, $\nu\in\Sigma^*$ $\upsilon\in\Sigma^*$, $B\in\Sigma^*$.

Определение 30. Праволинейной грамматикой (рациональной грамматикой, грамматикой типа 3) (right-linear grammar) называется порождающая грамматика, каждое правило которой имеет вид $\mathrm{A}\to\mathrm{u}$ или $A\to\mathrm{uB}$, где $A\in N$, $u\in\Sigma^*$, $B\in N$.

Пример: Грамматика $S\to\varepsilon$, $S\to aaaS$, $S\to abbS$, $S\to babS$, $S\to aabT$, $T\to abaT$, $T\to baaT$, $T\to bbbT$, $T\to bbaS$ праволинейная. Обобщённый вариант языка, порождаемого этой грамматикой, используется в доказательстве разрешимости арифметики Пресбургера [19, с. 207–208].

Определение 31. Правила вида $\alpha\to\varepsilon$ называются ε -правилами.

Лемма 1 для определения 31. Каждая праволинейная грамматика является линейной. Каждая линейная грамматика является контекстно-свободной. Каждая контекстно-свободная грамматика без ε -правил является контекстной грамматикой.

Определение 32. Классы грамматик типа 0, 1, 2 и 3 о3зуют иерархию Хомского (Chomsky hierarchy).

Определение 33. Язык называется контекстным языком (контекстно-свободным языком, линейным языком, праволинейным языком), если он порождается некоторой контекстной грамматикой (соответственно контекстно-свободной грамматикой, линейной грамматикой, праволинейной грамматикой). Контекстно-свободные языки называются также алге3ическими языками.

Пример: Пусть дан произвольный алфавит $\Sigma=\{\mathrm{a}_1,\ldots,\mathrm{a}_\mathrm{n}\}$. Тогда язык Σ^* является праволинейным, так как он порождается грамматикой $S\to\varepsilon$, $S\to a_1S,\ldots,a_nS$.

Определение 32. Конечный автомат (finite automaton, finitestate machine) можно представить в виде:

$$\mathrm{M}=\langle\mathrm{Q},\Sigma,\Delta,I,F\rangle,$$

где: Σ - конечный алфавит,

Q и Δ - конечные множества, $\Delta\subseteq Q\times\Sigma^*\times Q$ с условием $I\subseteq Q$, $F\subseteq Q$.

Q - называются состояниями (state),

I - начальными (initial) состояниями,

F - заключительными или допускающими (final, accepting) состояниями. Пусть в данном исследований

запись $\langle \mathrm{p},\xi,\mathrm{q}\rangle \in \Delta$, то то выражение $\langle \mathrm{p},\xi,\mathrm{q}\rangle$ будем называть переходом (transition) из p в q, а слово ξ - меткой (label) этого перехода.

Определение 33. Введем понятие ϑ путь (path) конечного автомата - это кортеж $q_0, e_1, q_1, ..., q_n$, где $n \geq 0$ и $e_i = \langle q_{i-1}, \xi_i, q_i \rangle \in \Delta$ для каждого i. При этом q_0 - начало пути, q_n - конец пути, n - длина пути, $\xi_1, ..., \xi_n$ - метка пути.

Определение 34. Состояние F достижимо (reachable) из состояния I, если существует путь ϑ, началом которого является I ,а концом - F.

Лемма 2 для определения 34. Каждый автоматный язык распознаётся некоторым конечным автоматом, в котором каждое состояние достижимо из некоторого начального состояния и из каждого состояния достижимо хотя бы одно заключительное состояние. Естественным обобщением конечного автомата является конечный преоЗзователь (finite-state transducer), позволяющий на каждом такте отправить несколько символов в дополнительный "выходной" поток [4, 3.3],[2, 3.1.3],[15, 0.1, 2.1–2.6]).

Определение 35. Путь называется успешным, если его начало принадлежит I, а конец принадлежит F.

Определение 36. Слово ξ допускается (is accepted, is recognized) конечным автоматом M, если оно является меткой некоторого успешного пути.

Определение 37. Язык, распознаваемый конечным автоматом M, - это язык $L(M)$, состоящий из меток всех успешных путей (то есть из всех допускаемых данным автоматом слов). Будем также говорить, что автомат M распознаёт (accepts) язык $L(M)$.

Определение 38. Два конечных автомата эквивалентны, если они распознают один и тот же я зык.

Определение 39. Язык L называется автоматным (finite-state language), если существует конечный автомат, распознающий этот язык.

Определение 40. Конфигурацией конечного автомата называется любая упорядоченная пара $\langle \mathrm{q}, \mathrm{w} \rangle$, где $q \in Q$ и $w \in \Sigma^*$.

Содержательно конфигурация представляет собой "мгновенное описание" конечного автомата. Если представить, что исходное слово, принадлежность которого рассматриваемому языку надо проверить, дано в некотором "входном потоке", то в конфигурации $\langle \mathrm{q}, \mathrm{w} \rangle$ слово w есть та часть исходного слова, которая пока осталась во входном потоке (это некоторый суффикс исходного слова), а q - текущее состояние "управляющего устройства".

Определение 41. Определим на множестве всех конфигураций конечного автомата M бинарное отношение $\succ$ (такт работы (step)) следующим образом. Если $\langle p,\xi,q\rangle \in \Delta$ и $w \in \Sigma^*$,то $\langle p,\xi w\rangle \succ \langle q,w\rangle$. Введем запись для понятия конфигураций конечного автомата вместо $\succ$ запишем $\underset{\Delta}{\succ}$.

Определение 42. Бинарное отношение $\overset{*}{\succ}\square$ определяется как рефлексивное, транзитивное замыкание отношения.

Лемма 3 для определения 42. Пусть дан конечный автомат $\mathrm{M} = \langle Q,\Sigma,\Delta,I,F\rangle$. Слово $w \in \Sigma^*$ принадлежит языку $L(M)$ тогда и только тогда, когда для некоторых $p \in I$ и $q \in F$ верно $\langle p,w\rangle \overset{*}{\succ} \langle q,\varepsilon\rangle$.

Лемма 4 для определения 42. Если $\langle q_1,x\rangle \overset{*}{\succ} \langle q_2,\varepsilon\rangle$ и $\langle q_2,y\rangle \overset{*}{\succ} \langle q_3,\varepsilon\rangle$, то $\langle q_1,xy\rangle \overset{*}{\succ} \langle q_3,\varepsilon\rangle$.

Лемма 5 . Каждый автоматный язык распознаётся некоторым конечным автоматом, не содержащим переходов с метками длины больше единицы и имеющим ровно одноначальное состояние и ровно одно заключительное состояние.

Лемма 6. Каждый автоматный язык распознаётся некоторым конечным автоматом, содержащим только переходы с метками длины единица и имеющим ровно одно начальное состояние.

Доказательство. Согласно лемме 5 можно предположить, что исходный язык задан конечным автоматом $\langle \mathrm{Q},\Sigma,\Delta,\mathrm{I},\mathrm{F}\rangle$,не содержащим переходов с метками длины больше единицы, причём $|I| = 1$. Построим искомый конечный автомат $\langle \mathrm{Q}',\Sigma',\Delta',\mathrm{I}',\mathrm{F}'\rangle$, положив $Q = Q'$, $I = I'$,

$\Delta' = \{\langle p,a,r\rangle \mid a \in \Sigma$ и найдеться такое $\mathrm{q} \in Q$, что $\langle \mathrm{q},\mathrm{a},\mathrm{r}\rangle \in \Delta$ и существует ϑ(путь) из p в q с меткой $\varepsilon\}$,

$\mathrm{F}' = \{\mathrm{p} \in Q \mid$ найдется такое $\mathrm{q} \in F$, что существует ϑ(путь) из p в q с меткой $\varepsilon\}$.

Теорема 1. Каждый автоматный язык является праволинейным.

Доказательство. Без ограничения общности можно предположить, что исходный язык задан конечным автоматом $\langle \mathrm{Q},\Sigma,\Delta,\mathrm{I},\mathrm{F}\rangle$, где $Q \cap \Sigma = \varnothing$ и $I = \{q_0\}$. Положим $N = Q$, $S = q_0$ и $P = \{p \to xq \mid \langle p,\xi,q\rangle \in \Delta\} \cup \{p \to \varepsilon \mid p \in F\}$.

Теорема 2. Каждый праволинейный язык является автоматным. (докозательство от противного теоремы 1)

Доказательство. Без ограничения общности можно предположить, что исходный язык задан праволинейной грамматикой, не содержащей правил вида $\mathrm{A} \to \mathrm{u}$, где $u \in \Sigma^+$. Положим $Q = N$, $I = \{S\}$, $F = \{A \in N \mid (A \to \varepsilon) \in P\}$ и $\Delta = \{\langle A,u,B\rangle \mid (A \to uB) \in P\}$.

Определение 43. Праволинейная грамматика в нормальной форме (автоматная грамматика, регулярная

грамматика) (finite-state grammar) -это праволинейная грамматика, в которой каждое правило имеет вид $A \to \varepsilon$, $A \to a$ или $A \to aB$, где $A \in N$, $B \in N$, $a \in \Sigma$ а.

Теорема 3. Каждая праволинейная грамматика эквивалентна некоторой праволинейной грамматике в нормальной форме.

Теорема 4. Если праволинейный язык не содержит пустого слова, то он порождается некоторой праволинейной грамматикой в нормальной форме без ε-правил.

Определение 44. Конечный автомат $\langle Q, \Sigma, \Delta, I, F \rangle$ называется детерминированным (deterministic), если выполняются следущие условия:

а) множество I содержит ровно один элемент;

б) для каждого перехода $\langle p, \xi, q \rangle \in \Delta$ выполняется равенство $|\xi| = 1$;

в) для любого символа $a \in \Sigma$ и для любого состояния $p \in Q$ существует не более одного состояния $q \in Q$ со свойством $\langle p, a, q \rangle \in \Delta$.

Определение 45. Детерминированный конечный автомат $\langle Q, \Sigma, \Delta, I, F \rangle$ называется полным (complete), если для каждого состояния $p \in Q$ и для каждого символа $a \in \Sigma$ найдётся такое состояние $q \in Q$, что $\langle p, a, q \rangle \in \Delta$.

Теорема 5. Каждый автоматный язык распознаётся некоторым полным детерминированным конечным автоматом.

Доказательство. Без ограничения общности можно предположить, что исходный язык задан конечным автоматом $\langle Q, \Sigma, \Delta, I, F \rangle$, содержащим только переходы с метками длины единица. Для любых $a \in \Sigma$ и $H \subseteq Q$ обозначим $\Delta_a(H) \leftrightarrow \{q \in Q \mid \langle p, a, q \rangle \in \Delta$ для некоторого $p \in H\}$. Обозначим через $\Im(Q)$ множество всех подмножеств множества Q. Построим искомый полный детерминированный конечный автомат $\langle Q', \Sigma', \Delta', I', F' \rangle$, положив $Q' = P(Q)$, $\Delta' = \{\langle H, a, \Delta_a, (H) \rangle \mid H \subseteq Q, a \in \Sigma\}$, $I' = \{I\}$ и $F' = \{H \subseteq Q \mid H \cap F \neq 0\}$.

Определение 46. Для любого состояния p полного детерминированного конечного автомата $\langle Q, \Sigma, \Delta, I, F \rangle$ и любого слова w обозначим через $\Delta_w(p)$ такое состояние q, что существует путь из p в q с меткой q (в силу полноты и детерминированности такое состояние существует и единственно).

Теорема замкнутости 6. Класс автоматных языков замкнут относительно итерации, конкатенации и объединения.

Доказательство. Без ограничения общности можно предположить, что каждый из исходных языков задан конечным автоматом с одним начальным и одним заключительным состоянием. Тогда во всех трёх случая результирующий автомат получается из исходных путём добавления нескольких ε-переходов и состояний и назначения новых начальных и заключительных состояний.

Теорема 7 пересечения и дополнения. Класс автоматных языков замкнут относительно дополнения и пересечения.

Доказательство. Если язык L распознаётся полным детерминированным конечным автоматом $\langle Q, \Sigma, \Delta, I, F \rangle$, то я зык $\Sigma^* - L$ распознаётся конечным автоматом $\langle Q, \Sigma, \Delta, I, Q - F \rangle$. Пересечение выражается через объединение и дополнение (закон де Моргана) и теоремы 5.

Лемма 7 о разрастании автоматных языков. Пусть L- автоматный язык над алфавитом Σ. Тогда найдётся такое положительное целое число p, что для любого слова $w \in L$ длины не меньше p найдутся слова $x, y, z \in \Sigma^*$, для которых верно $xyz = w$, $y \neq \varepsilon$, $|xy| \leq p$ и $xy^i z \in L$ для всех $i \geq 0$.

Доказательство. Пусть язык L распознаётся автоматом $\langle Q, \Sigma, \Delta, I, F \rangle$, содержащим только переходы с метками длины единица. Положим $p = |Q|$. Пусть слово w является меткой успешного пути $\langle q_0, e_1, q_1, e_2, ..., q_n \rangle$ и $|w| = n \geq p$. Согласно принципу Дирихле найдутся такие индексы j и k, что $0 \leq j < k \leq p$ и $q_j = q_k$. Выберем слова x, y и z так, что $|x| = j$, $|y| = k - j$ и $xyz = w$.

Теорема 8 наложение свойств гемоморфизма на автоматные языки. Для любого гомоморфизма (определение 19, 20, 21) $h : \Sigma_1^* \to \Sigma_2^*$ и автоматного языка $L \subseteq \Sigma_1^*$ язык $h(L)$ является автоматным.

Доказательство. Пусть исходный язык L задан конечным автоматом $M = \langle Q, \Sigma_1, \Delta, I, F \rangle$. Положим $\Delta' = \{\langle p, a, q \rangle \mid \langle p, x, q \rangle \in \Delta\}$. Тогда язык $h(L)$ распознаётся конечным автоматом $\langle Q, \Sigma_2, \Delta', I, F \rangle$.

Теорема 9. Для любого гомоморфизма $h : \Sigma_1^* \to \Sigma_2^*$ и автоматного языка $L \subseteq \Sigma_2^*$ язык $h^{-1}(L)$ является автоматным.

Доказательство. Без ограничения общности можно предположить, что исходный язык (L) задан конечным автоматом $M = \langle Q, \Sigma_2, \Delta, I, F \rangle$, где Δ не содержит переходов с метками длины больше единицы. Положим $\Delta' = \{\langle p, a, q \rangle \mid a \in \Sigma_1$ и существует путь из p в q с меткой $h(a)\}$.Тогда язык $h^{-1}(L)$ распознаётся конечным автоматом $M' = \langle Q, \Sigma_1, \Delta', I, F \rangle$.

Определение 47 длины слов в автоматных языках**.** Пусть $\Re \subseteq N$, $m \in N$ и $m > 0$. Множество $\Re$ называется заключительно периодическим (ultimately periodic) с периодом m, если выполнено условие $(\exists n_0 \in N)(\forall_n \geq n_0)(n \in \Re \leftrightarrow n + m \in \Re)$

Лемма 8. Пусть $\Re \subseteq N$. Тогда равносильны следующие утверждения:

а) множество $\Re$ является заключительно периодическим;

б) найдутся положительное целое число m и конечные множества $M \subseteq N$ и $K \subseteq \{0,1,...,m-1\}$, такие что $\Re = \{k \in N \mid (k \bmod \mathrm{m}) \in K\} - M$;

в) множество $\Re$ является объединением конечного семейства арифметических прогрессий.

Теорема 10. Язык L над однобуквенным алфавитом $\{a\}$ является автоматным тогда и только тогда, когда множество $\{k \in N \mid a^k \in L\}$ является заключительно периодическим.

Доказательство. Для доказательства необходимости достаточно рассмотреть детерминированный конечный автомат, распознающий язык L.

Теорема 11. Если язык L является автоматным, то множество $\{|w| \mid w \in L\}$ является заключительно периодическим.

Доказательство. Рассмотрим конечный автомат, распознающий язык L. Заменим все символы в метках переходов на символ a. применяя теорему 10 к полученному автоматному языку над однобуквенным алфавитом $\{a\}$.

Определение 48. Регулярное выражение над алфавитом Σ определяется рекурсивно следующим образом: 0 является регулярным выражением; 1 является регулярным выражением; если $\mathrm{a} \in \Sigma$, то a является регулярным выражением; если e и f являются регулярными выражениями, то $(e+f),(e \cdot f)$ и e^* тоже являются регулярными выражениями. Для упрощение математической записи скобок в данном исследований будем считать, что умножение связывает теснее, чем сложение. Вместо $(\mathrm{e} \cdot \mathrm{f})$ запишем просто ef.

Определение 49. Каждое регулярное выражение e над алфавитом Σ *задаёт* (denotes, represents) некоторый язык над алфавитом Σ (обозначение $L(e)$), определяемое рекурсивно следующим образом:

$L(a) \leftrightarrow \{a\}$, если $a \in \Sigma$,

$\mathrm{L}(0) \leftrightarrow \varnothing$,

$L(1) \leftrightarrow \{\varepsilon\}$,

$L(e+f) \leftrightarrow L(e) \cup L(f)$,

$L(e \cdot f) \leftrightarrow L(e) \cdot L(f)$,

$L(e^*) \leftrightarrow L(e)^*$.

Определение 50. Язык L называется регулярным, если он задаётся некоторым регулярным выражением.

Определение 51. Пусть e - регулярное выражение. Тогда $e^+ \leftrightarrow e^* e$.

Лемма 9 о свойствах регулярных выражений. Регулярные выражения образуют ассоциативное полукольцо с операциями $(0,+,1,\cdot)$, то есть для любых регулярных выражений e,f и g выполняются следующие тождества:

а) $\mathrm{e}+\mathrm{f}=\mathrm{f}+\mathrm{e}$,

б) $e+0=e$,

в) $(e+f)+g=e+(f+g)$,

г) $e \cdot 1=e$,

д) $1 \cdot e=e$,

е) $(e \cdot f) \cdot g=e \cdot (f \cdot g)$,

ж) $e \cdot (f+g)=e \cdot f+e \cdot g$,

з) $(f+g) \cdot e=f \cdot e+g \cdot e$,

и) $e \cdot 0=0$,

к) $0 \cdot e=0$.

С условием что равенство понимается как равенство языков, задаваемых регулярными выражениями

Лемма 10. Для любых регулярных выражений e и f выполняются следующие тождества:

а) $\mathrm{e}+\mathrm{e}=\mathrm{e}$,

б) $(1+e+ee+...+e^{n-1})(e^n)^*=e^*$ для любого $n \geq 1$,

в) $(\mathrm{e}^* \mathrm{f})^* e^*=(e+f)^*$,

г) $1+\mathrm{e}(fe)^* f=(ef)^*$.

Лемма 11. Для любых регулярных выражений e, f и g, если $e=ef+g$ и $\varepsilon \notin L(f)\varepsilon$, то $e=gf^*$.

Теорема 12 (теорема Клини). Язык L является регулярным тогда и только тогда, когда он является автоматным.

Доказательство. Пусть e - регулярное выражение. Индукцией по построению e легко показать, что задаваемый им язык является автоматным (из теоремы замкнутости 6). Обратно, пусть язык L распознаётся некоторым (недетерминированным) конечным автоматом с одним начальным состоянием и одним заключительным состоянием. Существует эквивалентный ему обобщённый конечный автомат $\langle \mathrm{Q},\Sigma,\Delta,\{\mathrm{q}_1\},\{\mathrm{q}_2\}\rangle$, где $q_1 \neq q_2$. Если есть несколько переходов с общим началом и общим концом (такие переходы называются *параллельными*), заменим их на один переход, используя операцию $+$.

Устраним по очереди все состояния, кроме q_1 и q_2. При устранении состояния q надо для каждого перехода вида $\langle p_1, f_1, q\rangle$, где $p_1 \neq q$, и для каждого перехода вида $\langle q, f_2, p_2\rangle$, где $p_2 \neq q$, добавить переход $\langle p_1, f g^* f_2, p_2\rangle$, где регулярное выражение g -метка перехода из q в q (если такого перехода нет, то считаем, что g = 0), и снова всюду заменить параллельные переходы на один переход, используя операцию $+$.

После устранения всех состояний, кроме q_1 и q_2, получится обобщённый конечный автомат $\langle \{q_1,q_2\},\Sigma,\Delta',\{q_1\}\{q_2\}\rangle$, где

$\Delta' = \{\langle q_1,e_{11},q_1\rangle,\langle q_1,e_{12},q_2\rangle,\langle q_2,e_{21},q_1\rangle,\langle q_2,e_{22},q_2\rangle\}$.

Очевидно, $L = L(e_{11}^* e_{12}(e_{22}+e_{21}e_{11}^* e_{12})^*)$.

Определение 52 деревья вывода**.** Выводам в контекстно-свободной грамматике соответствуют так называемые *деревья вывода* (или *деревья разбора*) (derivation tree, parse tree) - некоторые упорядоченные деревья, вершины которых помечены символами алфавита $N \cup \Sigma$.

Корень дерева отвечает начальному символу. Каждому символу слова w_1, на которую заменяется начальный символ на первом шаге вывода, ставится в соответствие вершина дерева, и к ней проводится дуга из корня. Полученные таким образом непосредственные потомки корня упорядочены согласно порядку их меток в слове w_1. Для тех из полученных вершин, которые помечены символами из множества N, делается анлогичное построение и т. д.

Кроной дерева вывода называется слово, записанное в вершинах, помеченных словами из алфавита Σ.

Выводы

В статье приведены основные определения, леммы, а также теоремы с доказательствами, которые показывают возможность применения теории формальных языков для описания языков высокого уровня программирования. На базе предложенных математических выражений планируется разработать методологию расчёта стоимости программного обеспечения на ранней стадии составления технического задания на базе прототипа интерфейса пользователя.

СПИСОК ЛИТЕРАТУРЫ:

1. Невлюдов И.Ш. Анализ применимости математических моделей СОСОМО при разработке современных корпоративно - информационных систем технологической подготовки производства / И.Ш. Невлюдов, В.В. Евсеев, В.О. Бортникова // Кременчуг: КНУ им. М. Остроградского, 2011. – С. 152-153.

2. Невлюдов И.Ш. Модели жизненного цикла программного обеспечения при разработке корпоративных информационных систем технологичекой подготовки производств / И.Ш. Невлюдов, В.В. Евсеев, В.О. Бортникова // Вестник Националь-ного технического уни-верситета «ХПИ». - Харьков.-2011.- Вып. № 2.-С.94-101.

3. Евсеев В.В. Применение программных метрик кода на раннем этапе жизненного цикла программного обеспечения / В.В. Евсеев // Восточно-европейский журнал передовых технологий. - Харьков.-2011.- Вып. № 1/2 (49).-С.19-21.

4. Невлюдов И.Ш. Анализ жизненного цикла разработки программного обеспечения для корпоративных информационных систем/ И.Ш. Невлюдов, А.А. Андрусевич, В.В. Евсеев // Восточно-европейский журнал передовых технологий. - Харьков.-2010.- Вып. 6/8(48).-С.25-27.

5. Пентус А.Е. Теория формальных языков / А.Е. Пентус, М.Р. Пентус. – М.: Издательство Центра прикладных исследований при механико-математическом факультете МГУ, 2004 – 80 с.

УДК 681.7.068.4

ЗАСТОСУВАННЯ АВТОКОНВОЛЮЦІЙНОГО МЕТОДУ АНАЛІЗУ ЗОБРАЖЕНЬ ФОТОННО-КРИСТАЛІЧНИХ ВОЛОКОН В АВТОМАТИЗОВАНІЙ СИСТЕМІ КЕРУВАННЯ ПРОЦЕСОМ ЇХ З'ЄДНАННЯ

Д.т.н. О.І.Филипенко, О.В. Сичова, Харківський національній університет радіоелектроніки

Наведено функціональну схему та алгоритм системи керування апаратом зварки оптичних волокон. Запропоновано класифікацію та склад технічного контролю, який виконується в автоматизованій системі керування технологічним процесом зварювання з метою підвищення якості з'єднання оптичних волокон. Розглянуто методи позиціонування стандартних оптичних волокон, виявлено, що найбільш придатним у випадку з'єднання фотонно-кристалічних волокон є метод фокусування.

Приведено функциональная схема и алгоритм системы управления аппаратом сварки оптических волокон. Предложено классификация и состав технического контроля, выполняющегося в автоматизированной системе управления технологическим процессом сварки с целью повышения качества соединения оптических волокон. Рассмотрено методы позиционирования стандартных оптических волокон, выявлено, что наиболее подходящим в случае соединения фотонно-кристаллических волокон является метод фокусировки.

The article presents functional scheme and algorithm the control system device welding optical fibers. Classification and structure of technical control in automated control system of welding process to improve the connection quality of optical fibers proposes. The methods of positioning standard optical fibers reviewed. The most suitable for the connection of photonic crystal fibers is focusing method.

Ключові слова: *фотонно-кристалічне волокно, позиціонування, зварювання, втрати сигналу, контроль якості з'єднання*

Загальна характеристика проблеми

Під час експлуатації елементів функціональної електроніки, побудованих на основі фотонно-кристалічних волокнах (ФКВ), виникає необхідність їх з'єднання між собою або зі стандартними оптичними волокнами (ОВ). З'єднання бувають роз'ємними або нероз'ємними. Нероз'ємні з'єднання можуть отримуватися за допомогою зварювальних пристроїв. Автоматизація процесу зварювання виконується шляхом введення у зварювальний апарат системи автоматичного керування [1], узагальнена функціональна схема якої представлена на рис. 1.