НАУЧНОЕ ПРИБОРОСТРОЕНИЕ, 1995, том 5, № 3-4, с. 66 - 73

СИСТЕМЫ СБОРА ДАННЫХ И УПРАВЛЕНИЕ

УДК 519.683 :681.3.015

О ПРЕДСТАВЛЕНИИ ПРОГРАММ ЧЕЛОВЕКО-ОРИЕНТИРОВАННЫМИ ВНЕШНИМИ ЯВЛЕНИЯМИ*

© 1995г. **В.И. Курганский**

Иркутский государственный университет

Рассмотрены проблемы конструирования любых завершающихся за конечное число шагов структурированных программ для диалоговых систем на основе их представления внешними явлениями.

Введение

С момента появления первых трансляторов и специалисты, и пользователи вычислительных систем различают два уровня представления программ – машинно-ориентированное и человеко-ориентированное. Во всех случаях речь идет о последовательных текстах. Появление нового поколения диалоговых механизмов, новой мультимедиа аппаратуры привело и к новому уровню человеко-ориентированного представления программ – внешними явлениями, сопровождающими выполнение программы на примерах.

Следует отметить, что первые дивиденды от представления программ внешними явлениями уже получены и поделены. Хорошо известны системы визуального программирования, программирования на демонстрациях и другие.

Для дальнейшего развития этих разработок необходимы исследования потенциальных возможностей представления программ внешними явлениями, разработка подходящих языковых и других технологических средств.

В работе разрабатывается универсальная нормальная форма структурированных программ без рекурсивных подпрограмм. Доказывается существование функционально эквивалентной нормальной формы для любой структурированной программы (п.1). Здесь же исследуется связь введенной нормальной формы структурированных программ с их наивной теоретико-множественной спецификацией.

Результаты п.1, а также методические соображения Д.Гриса о приемах конструирования программ [1] позволили синтезировать методику конструирования любых завершающихся за конечное число шагов структурированных программ для диалоговых систем на основе их представления внешними явлениями.

1. Нормальная форма структурированных программ

Нормальная форма структурированных программ вводится и исследуется с целью выделения некоторых методических основ для представления программ внешними человеко-ориентированными явлениями. Возьмем за основу нотацию для записи алгоритмов и программ, предложенную Э.Дейкстрой и развитую Д.Грисом [1]. Заметим, что определение языка программирования предполагает явное или неявное определение машины обработки данных. В нашем случае мы должны определить организацию операционного пространства, систему простейших команд, устройство программной памяти и основной алгоритм работы испол-

*Работа выполняется при поддержке Российского фонда фундаментальных исследований - проект №95-01-01465.

нителя по программе. Напомним, что рассматриваемый язык предназначен для кодирования программ, интерпретируемых диалоговыми системами.

Относительно операционного пространства положим, что оно содержит скалярные элементы, имеющие простые типы значений – целый, вещественный, строковый и булевский. Набор этих типов сейчас несущественен и нам необходим только булевский тип данных. Простые типы могут объединяться в составные. Будем полагать, что все возможности объединения определяются используемыми для доступа к операционному пространству диалоговыми механизмами. В настоящее время в диалоговых системах используются аналоги записей (комплекты разнотипных значений), одномерные и многомерные массивы, файлы и др.

Будем различать два способа адресации элементов операционного пространства – прямой и косвенный. При прямой адресации диалоговая система должна обеспечивать средства непосредственного именования обрабатываемых данных. Так, в табличных процессорах одной или двумя буквами обозначаются графы, целыми без знака – строки, а конкатенацией имен – графы и строки, клетки электронного бланка. При косвенной адресации для доступа к текущему или какому-либо другому элементу операционного пространства (например, клетке в табличном окне, символу в текстовом окне и т.д.) используют специальные функции.

Система простейших команд определяется возможностями программно-управляемой диалоговой системы. Несмотря на кажущееся разнообразие, набор этих команд сводится к следующей паре

- присваивание,
- обращение к подпрограмме.

Рассмотрим их подробнее. Команду присваивания ниже будем кодировать так [1]:

$$X_1, \ldots, X_n := E_1, \ldots, E_n.$$

$X_1, \ldots, X_n$ символизируют программные переменные, а $E_1, \ldots, E_n$ – вычисляемые выражения. Выполнение оператора сводится к вычислению $E_1, \ldots, E_n$. Предполагается, что при их вычислении не возникает никаких побочных эффектов (например, изменения аргументов в E_i при вычислении $E_j, i \neq j$). В отличие от распространенной в процедурном программировании команды присваивания введенную команду будем называть командой кратного или параллельного присваивания. Заметим, что практически все команды в диалоговых механизмах сводятся к параллельному присваиванию. Команда обращения к подпрограмме имеет вид:

$$< \text{Имя подпрограммы} > (< \text{список значений параметров} >).$$

Имя подпрограммы идентифицирует программный код в каком-либо их хранилище (библиотеке) или в составе рассматриваемой программы. Список значений параметров – это вычисляемые выражения и их частные случаи – имена программных переменных. По каждому параметру при обращении к подпрограмме или при ее описании указывают способ его передачи. Различают четыре таких способа [1]

- передача наименования,
- передача значения-аргумента,
- передача значения-результата,
- передача значения и аргумента, и результата.

Семантика составных операторов процедурного программирования задается алгоритмом интерпретации текста программы исполнителем рассматриваемой машины обработки данных. Во всех случаях в качестве элементов в составных операторах можно использовать введенные выше простые команды и составные операторы. Ниже для конструирования составных операторов будем использовать следующие средства:

- последовательную композицию операторов,
- оператор выбора,
- оператор цикла,
- подпрограммы.

Рассмотрим их. Если S_1 и S_2 операторы, то $S_1; S_2$ тоже оператор. Его выполнение заключается в последовательном выполнении вначале оператора S_1, а потом – S_2.

Оператор выбора имеет вид

$$If \ldots \square B_i \rightarrow S_i \ldots Fi,$$

где $i = 1, \ldots, k$, B_i – булевые выражения, а S_i – операторы. Семантика оператора $If \ldots Fi$ очевидна. Его выполнение начинается с вычисления условий B_i. Порядок их вычисления несущественен. Если $BB = B_1 \vee \ldots \vee B_n = 0$, то выполнение программы завершается аварийно. В противном слу-

чае выполняется оператор S_i для найденного первым истинного B_i. Потребуем, чтобы $BB = 1, B_i \wedge B_j = 0$ при $i \neq j$, и отсутствия побочных эффектов при вычислении B_i. Оператор цикла синтаксически похож на оператор выбора:

$$Do \ldots \square\ B_i \rightarrow S_i \ldots Od.$$

Его компоненты B_i и S_i имеют тот же смысл, что и в операторе выбора. На каждом шаге цикла выполняется только одна из его ветвей, а именно та, для которой в первую очередь было обнаружено истинное условие B_i. Цикл выполняется до тех пор, пока $BB = B_1 \vee \ldots \vee B_n = 1$.

Потребуем, как и выше, чтобы программа была детерминированной, т.е. $B_i \wedge B_j = 0$ при $i \neq j$, и отсутствия побочных эффектов при вычислении B_i.

Аппарат подпрограмм используется в традиционном для процедурного программирования ключе. Потребуем дополнительно, чтобы используемые подпрограммы были нерекурсивными, т.е. не обращались сами к себе. Потребуем также, чтобы наши программы при исполнении не модифицировали сами себя.

Таким образом, исследуется широкий класс программ, обеспечивающий многие приложения.

Введем и исследуем специальные формы представления структурированных программ для их представления внешними явлениями.

Определение. Структурированная программа вида

$$NP_0 \quad \text{или} \quad NP_0; Do \ldots \square B_i \rightarrow NP_i \ldots Od\ , \tag{1}$$

где NP_0, NP_i – операторы кратного присваивания, называется *нормальной*.

Теорема 1. *Для всякой структурированной программы существует функционально эквивалентная ей нормальная форма.*

Доказательство. Для получения нормальной формы будем использовать следующие эквивалентные преобразования.

1) $If \ldots \square B_i \rightarrow S_i \ldots Fi$ эквивалентно
$Ind := 1; Do \ldots \square(Ind \wedge B_i) \rightarrow S_i; Ind := 0 \ldots Od$, где S_i – произвольные операторы.
2) $Do \ldots \square B_i \rightarrow S_i \ldots Od; S$ эквивалентно
$Ind := 1; Do \ldots \square(B_i \wedge Ind) \rightarrow S_i \ldots \square(\neg BB \wedge Ind) \rightarrow S; Ind := 0; \ldots Od$, где $BB = B_1 \vee \ldots \vee B_n$, а S, S_i – произвольные операторы.
3) S эквивалентно
$Ind := 1; Do\ \square Ind \rightarrow S; Ind := 0; \ldots Od$, где S – произвольный оператор.
4) $Do \ldots \square B_k \rightarrow Ind_1, \ldots, Ind_m := 1, \ldots, 1; Do \ldots \square B_i \rightarrow S \ldots Od \ldots Od$ эквивалентно
$Ind := 1; Do \ldots \square(Ind \wedge B_k) \rightarrow Ind_1, \ldots, Ind_m, Ind := 1, \ldots, 1, 0$
$\square(\neg Ind \wedge B_k \wedge B_i) \rightarrow S\ \ \square(\neg Ind \wedge \neg B_k \wedge \neg B_i) \rightarrow Ind := 1 \ldots Od$,
где S – произвольный оператор, $Ind_1, \ldots, Ind_m$ – булевые индикаторы, введенные при осуществлении преобразований (1) - (4).
5) $X_1 :=< Exp_1 >; \ldots; X_n :=< Exp_n >$ эквивалентно
$X_1, \ldots, X_n :=< Exp_{1m} >, \ldots, < Exp_{nm} >$, где $< Exp_{im} >$ – модификация $< Exp_i >$, которая заключается в замене всех аргументов $X_j (j < i)$ на $< Exp_j >$. Если в подставляемом выражении $< Exp_j >$ имеют место вхождения X_k $(k < j)$, то они тоже должны быть рекурсивно заменены на $< Exp_k >$.

Очевидно, что если применять эти преобразования к программе без подпрограмм, то в конце концов получается функционально эквивалентная программа вида (1). Избавиться в произвольной программе от обращений к подпрограммам можно заменой всех обращений к ним их кодами. Подстановки выражений фактических параметров вместо соответствующих им формальных параметров достаточно очевидны и здесь не приводятся.

Теорема о нормальной форме структурированных подпрограмм коррелирует с известными результатами теории алгоритмов и теоретического программирования: нормальными алгорифмами Маркова [2], нормальным представлением Клини частично рекурсивных функций [2], конструктивными логиками схем программ [3].

Рассмотрим связь нормальной формы структурированной программы с ее наивными теоретико-множественными спецификациями. Из технических соображений будем полагать, что последние задаются булевыми

выражениями Q и R такими, что $\{Q\}S\{R\}$ [1]. Запись $\{Q\}S\{R\}$ означает, что всякое состояние, удовлетворяющее Q, после выполнения S преобразуется в состояние, удовлетворяющее R. Предикат Q и соответствующее ему подмножество пространства состояний называют предусловием S, а предикат R – постусловием S. Среди всех возможных предусловий оператора S выделяют его слабейшее предусловие относительно постусловия R. Обозначают слабейшее предусловие $WP(S,R)$. Слабейшее предусловие – это множество **всех** состояний, применение к которым оператора S дает состояние, удовлетворяющее R. Кроме постусловия и слабейшего предусловия программы S, приведенной к нормальной форме, интересны постусловия и слабейшие предусловия операторов NP_i из нормальной формы (1) как таковых, этих же операторов при переходе от одной ветви цикла к другой, а также при завершении цикла. Обозначим B, E – подмножества начальных и конечных состояний программы S соответственно. Пусть R – постусловие, а $Q = WP(S,R)$ – слабейшее предусловие. Обозначим $Q = (q_1 \dots q_n)$ и $R = (r_1 \dots r_m)$ элементарные высказывания, из которых строятся слабейшее предусловие и постусловие. Потребуем, чтобы эти высказывания строились из отношений сравнения ("равно" и "не равно"), отношений строгого и нестрогого порядка, а также всюду определенных предикатов вычислимости заданных выражений. Речь идет о выражениях, фигурирующих в отношениях сравнения и порядка, а также о выражениях в операторах присваивания, выражениях-аргументах подпрограмм и булевых выражениях-охранах операторов выбора и цикла. Частным случаем отношения вычислимости заданного выражения является отношение принадлежности скалярного значения заданному типу значений. Величины q_i, r_j могут принимать значения из множества $\{0,1,2\}$, где 0 – символизирует ложь, 1 – истину, а 2 – неопределенное значение. Третье значение переменной-высказывания необходимо для отражения частичной определенности некоторых выражений в пространстве состояний программы и, в частности, наличия программных переменных с неопределенными значениями. Избавиться от этого третьего значения можно заменой всех элементарных отношений из q и r квазиконъюнкциями вида $d \oplus q_i$ и $d \oplus r_j$, где d – конъюнкция предикатов о вычислимости выражений из q_i или r_j. Квазиконъюнкция $\oplus$ отличается от обычной конъюнкции тем, что она не коммутативна, вычисляется слева направо, вычисления прекращаются как только обнаруживается ложный сомножитель.

Программа S реализует отображение $S : B \to E$. Рассмотрим отображения $Q : B \to Q$, $R : E \to R$, $S : B \to E$ и $WP(S,R) : R \to Q$. Потребуем, чтобы

$$WP(S,R)(R(S(B))) = Q(b) \tag{2}$$

Пусть

$$\begin{gathered} Q = WP(NP_0, (P_0 \vee \dots \vee P_n)), \\ \text{а} \\ R = (P_0 \vee \dots \vee P_n) \wedge \neg(\\ B_{1m} \vee \dots \vee B_{nm}), \end{gathered} \tag{3}$$

где P_i – конъюнкции отношений сравнения, построенные из оператора кратного присваивания NP_i, B_{im} – модифицированные охраны из оператора цикла нормальной формы. Правила построения P_i и B_{im}

- неитеративные присваивания вида $X := < Exp >$ преобразуются в отношения сравнения $X =< Exp >$,
- итеративные присваивания вида $X := < Exp(X) >$ преобразуются в отношения сравнения вида $X_c :=< Exp(X_a) >$ (X_c и X_a символизируют значения величины X в последующем и предыдущем состояниях программы),
- B_{im} строятся умножением слева (квазиконъюнкцией $\oplus$) предикатов, специфицирующих вычислимость выражений из B_i и NP_i, на условия B_i из (1).

Теорема 2. *Для всякой структурированной программы S, завершающейся за конечное число шагов, и ее булевой спецификации вида (3) справедливо* (2).

Доказательство. Рассмотрим произвольную точку b пространства начальных состояний B такую, что $Q(b) = 1$ и $e = S(b)$. Покажем, что $R(e) = 1$. По построению Q $P(NP_0(b)) = 1$. Напомним, что по теореме об инварианте цикла [1] $R(x) = P(x) \wedge \neg BB(x)$. Если $BB(NP_0(b)) = 0$, то $R(b) = P(b) \wedge \neg BB(b) = 1$, что и означает справедливость теоремы. Если срабатывает какая-либо ветвь цикла, то все изменения значений величин, имевшие место при

этом, по построению инварианта сохраняют истинность хотя бы для одного из его слагаемых P_i. В силу предположения о конечности программы S и в этом случае $R(e) = P(e) \wedge \neg BB(e) = 1$.

Доказанная теорема конструктивно гарантирует построение из текста программы ее постусловия и слабейшего предусловия. Программа должна заканчиваться за конечное число шагов. Нетрудно видеть, что постусловия и слабейшие предусловия операторов NP_i из нормальной формы (1) как таковых и при переходе от одной ветви цикла к другой, а также при завершении цикла присутствуют в (3). Это формы P_i, $P_i \wedge B_i$ и $P_i \wedge B_j$.

2. Требования к инструментальным средствам программирования

Рассмотрим систему программирования, базирующуюся на наивном теоретико-множественном представлении программы. Предполагается, что переход от наивного теоретико-множественного представления программы к ее представлению внешними человеко-ориентированными явлениями наиболее прост. Назначение системы программирования – подготовка и эксплуатация программ, представляемых внешними явлениями.

Существенным усложнением принципа программного управления в диалоговых системах по сравнению с этим принципом для машины фон Неймана является часто практикуемая возможность приостановки программно-управляемого процесса с последующими диалоговой обработкой данных и возобновлением выполнения программы. Эта проблема сравнительно просто решается за счет разбиения конструируемой программы на фрагменты, приостановка выполнения которых невозможна, и применения средств событийного управления для запуска таких программ. Конструктивно события можно задавать с помощью булевых и квазибулевых в смысле п.1 выражений. Дополнительные примитивы в этих выражениях по сравнению с традиционными средствами кодирования условий в процедурном программировании являются предикатами, задающими разбиение возможных входных воздействий на классы эквивалентности. Примеры классов – нажата клавиша Enter, нажата алфавитно-цифровая клавиша, указатель мыши переместился в определенную область экрана и т.п. Еще одна специфическая особенность программного управления в диалоговых системах заключается в необходимости управления отображением изменений их состояния в процессе выполнения программ. Применение средств событийного управления позволяет решить эту проблему, автоматически отображая состояние диалоговой системы после завершения обработки всякого события. Поскольку внешними явлениями в общем случае естественно представляются только последовательные программы, то необходимо некоторое промежуточное человеко-ориентированное представление программы в целом. В качестве такого представления предлагается использовать конструктивные описания дискретных упорядоченных множеств и их примеры.

Нормальная форма структурированных программ вида (1) позволяет строить программу как совокупность упорядоченных множеств. Множество S называется упорядоченным, если заданы

- условия принадлежности этому множеству любого значения в виде вычисляемого булевого (или квазибулевого в смысле п.1) выражения B_i,
- выражения, параллельное вычисление которых позволяет получить первый обрабатываемый элемент S,
- выражения, вычисление которых позволяет получить очередной обрабатываемый элемент S.

Процедурными средствами упорядоченное множество задается

- оператором кратного присваивания NPS_0 для вычисления первого элемента множества,
- оператором выбора вида $If \ldots \square BS_i \rightarrow NPS_i \ldots Fi$, где BS_i – булевые выражения, дизъюнкция которых задает принадлежность заданного значения множеству, а NPS_i – операторы присваивания, выполнение которых обеспечивает переход от заданного элемента множества к следующему.

Частным случаем задания упорядоченного множества, непосредственно вытекающим из синтаксической структуры нормальной формы структурированных программ, является тройка NPS_0, NPB_1, NPS_1. Эта тройка соответствует простейшему оператору выбора с одной ветвью.

Изучение ряда программ позволяет выделить в элементах NP_0, B_i и NP_i нормальной формы программы компоненты упорядоченного множества – BS_i, NPS_i – и так называемый конструктивный компонент. Синтаксически конструктивный компонент аналогичен упорядоченному множеству. Например, в программе сортировки конструктивный компонент обычно представляет собой условие перестановки двух из обрабатываемых элементов и собственно их перестановку. Начальная установка этого конструктивного компонента совпадает с начальной установкой соответствующего упорядоченного множества.

Таким образом, предлагаемый способ конструирования программ заключается в разработке системы событий и конструировании программы обработки события для каждого их класса. Конструирование программ обработки событий сводится к трем следующим технологическим элементам

- формированию подходящей совокупности упорядоченных множеств,
- формированию подходящей совокупности конструктивных компонентов,
- надлежащему соединению упорядоченных множеств и конструктивных компонентов.

Поскольку речь идет о конструировании программ для диалоговых систем, каждая из которых предполагает определенные множества входных воздействий и способы организации данных (например, реляционные базы данных, таблицы, текстовые и квазитекстовые файлы и пр.), целесообразно разрабатывать и вести библиотеки, элементами которых являются элементарные абстрактные события, абстрактные упорядоченные множества и абстрактные конструктивные компоненты. Переход от абстрактных элементов этих библиотек к конкретным осуществляется подстановкой подходящих аргументов выбранных абстрактных выражений. Фактическими аргументами могут быть

- значение текущего элемента программируемого диалогового механизма,
- абсолютный адрес текущего элемента программируемого диалогового механизма,
- относительный адрес текущего элемента программируемого диалогового механизма.

Результаты известной теории решения изобретательских задач [4] позволяют предположить, что мощность этих библиотек относительно невелика.

Чрезвычайно важной представляется возможность человеко-ориентированного представления на примерах, настроенных на конкретные условия применения, элементов библиотек событий, упорядоченных множеств и конструктивных компонентов.

Рассмотрим вначале упорядоченные множества. Пример такого упорядоченного множества размещается в операционном пространстве программируемой диалоговой системы, обозревается и редактируется средствами этой диалоговой системы. Факторами, повышающими сложность нашей задачи, являются

- размещение элементов упорядоченного множества как в различных элементах операционного пространства программируемой диалоговой системы, так и в различных экземплярах одного и того же элемента операционного пространства, соответствующих различным элементам пространства состояний конструируемой программы;
- сложность элементов упорядоченного множества в том смысле, что примером элемента может быть как скалярное значение, так и упорядоченное множество.

Представление на примере упорядоченного множества предполагает:

- обзор некоторого множества значений, часть из которых принадлежит рассматриваемому множеству, а часть ему не принадлежит;
- выделение элементов, принадлежащих множеству;
- человеко-ориентированное воспроизведение на примере операторов присваивания, реализующих установку на первый элемент упорядоченного множества, а также переход от текущего элемента множества к следующему.

Человеко-ориентированное представление на примере настроенных на конкретные условия применения конструктивных элементов является частным случаем человеко-ориентированного представления упорядоченного множества.

Рассмотрим теперь человеко-ориентированное представление событий. Библиотека событий содержит абстрактные элементарные события, как то: функции, индицирующие принадлежность входного воздействия классу, попытку смещения за конец файла и

т.п. Обрабатываемое событие характеризуется булевым или квазибулевым в смысле п.1 выражением, составленным из базовых отношений о свойствах входного воздействия, состоянии диалоговой системы и значений элементов ее операционного пространства. Это выражение должно вычисляться перед обработкой всякого входного воздействия.

Пример события строится на основе примера для подходящей пары, составленной из упорядоченного множества и конструктивного компонента. Условия выделения упорядоченного множества и конструктивного компонента усиливаются подходящим отношением о принадлежности последнего входного воздействия определенному их классу. Вопрос о человеко-ориентированном представлении такого отношения сводится к заблаговременному созданию для каждой программируемой диалоговой системы подходящего набора пиктограмм о возможных входных воздействиях и их свойствах.

Собственно конструирование программы для обработки отдельных событий на основе библиотек упорядоченных множеств и конструктивных элементов сводится к следующим двум технологическим элементам

- выбору и настройке подходящих элементов библиотек,
- надлежащему их соединению с целью получения программы в нормальной или квазинормальной формах.

Например, программа поиска максимального значения среди сумм элементов строк прямоугольной матрицы строится из множества строк, множества элементов строки, конструктивного компонента выделения максимального значения и конструктивного компонента вычисления суммы элементов множества.

При пополнении библиотек упорядоченных множеств и конструктивных элементов наиболее сложной является задача формирования человеко-ориентированного представления выражений, фигурирующих в вычисляемых условиях, операторах начальной установки и перехода от текущего элемента к следующему. По-видимому, в общем случае эта задача неразрешима. Тем не менее, следует отметить, что задача представления человеко-ориентированными внешними явлениями частного случая оператора выбора, реализующего упорядоченное множество или конструктивный компонент, существенно проще этой задачи для программы произвольной синтаксической структуры.

Заключение

Предмет исследования настоящей работы — структурированные программы без рекурсивных подпрограмм. Эти результаты достаточно просто могут быть распространены на рекурсивные программы. Теоретической базой здесь является теорема Клини о нормальной форме частично рекурсивных функций [2].

Перспективной также является задача представления человеко-ориентированными внешними явлениями программ более высокого уровня, например, реляционных запросов. Известная технология Query by example может быть развита за счет разработки библиотеки конструктивных компонентов для программирования скалярных вычислений.

Разработка программ представлением на примерах событий, упорядоченных множеств и конструктивных компонентов позволяет одновременно испытывать их на достаточно полных тестах. Тест строится разбиением пространства состояния программы с помощью булевых или квазибулевых выражений, фигурирующих в ее исходном тексте. Тест, построенный на основе внешних спецификаций программы [5], является его подмножеством.

Литература

1. *Грис Д.* Наука программирования. М.: Мир, 1984. 416 с.

2. *Мальцев А.И.* Алгоритмы и рекурсивные функции. М.: Наука, 1986. 308 с.

3. *Непейвода Н.Н.* Некоторые семантические конструкции конструктивных логик схем программ // Теория алгоритмов и ее приложения (Вычислительные системы). Новосибирск, 1989. Вып.129. С.49 - 66.

4. *Половинкин А.И.* Основы инженерного творчества. М.: Машиностроение, 1988. 368 с.

5. *Корольков Ю.Д., Курганская Г.С., Курганский В.И.* Моделирование внешних спецификаций программ формулами алгебры логики для построения тестов // Интеллектуализация программных средств. Новосибирск: Наука, 1990. С.76 - 80.

ON THE PROGRAM REPRESENTATION BY MAN-ORIENTED EXTERNAL EVENTS

V.I. Kurgansky

Periodic signals generators (PSC) are intended to form contiguous differential signals to simulate voltage which is applied to the input of measurement channels of chemical mass spectrometers, liquid chromatographs, etc., to create scan control signals. The design of PSC is based on the method of restoring a discrete time chain of code variables by spline approximator of the second order.