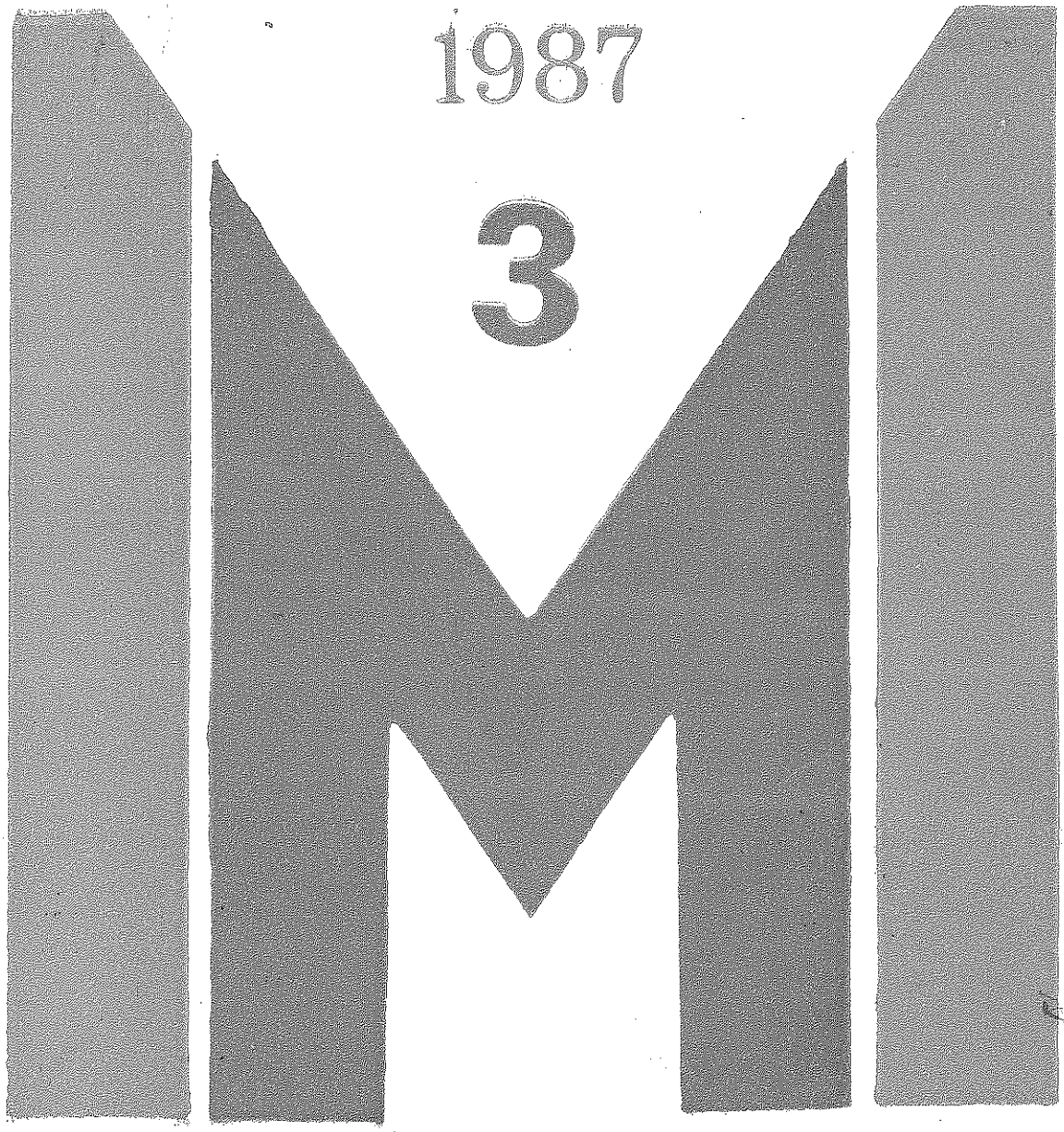
ISSN 0021—3446

ИЗВЕСТИЯ ВЫСШИХ УЧЕБНЫХ ЗАВЕДЕНИЙ

МАТЕМАТИКА

1987

3



ОТДЕЛЬНЫЙ ОТТИСК

будут пространства, допускающие транзитивную группу движений G_r порядка $r=n(n-1)/2+1$ — подгруппу группы движений n-мерного пространства Лобачевского. Метрические функции указанных пространств выражаются формулами

$$F=F(u^1,\ e^{2k_1x^1}/\bar{\gamma}_2),\ k_1\in R,\ \det\|F_{.j.k}\|\neq 0.$$

Перейдем к определению пространств $H_{n,\underline{u}}$ гиперплоскостных элементов определенной метрики второй лакунарности. Искомые пространства допускают группу G_r движений порядка $r=n(n-1)/2$, продолженную на гиперплоскостные элементы. Интегрируя уравнения инвариантности, получим

$$H=H(x^1,\ u_1,\ B^2\underline{\gamma}_2),\ \underline{\gamma}_2=\sum_{\lambda=2}^{n}u_\lambda^2,\ \det\|H^{\cdot j\cdot k}\|\neq 0. \tag{2}$$

Таким образом, приходим к теореме.

Теорема 5. *Метрические функции пространства $H_{n,\underline{u}}$ гиперплоскостных элементов знакоопределенной метрики второй лакунарности с интранзитивной группой движений G_r порядка $r=n(n-1)/2$ выражаются формулами* (2), *где H — произвольная дифференцируемая функция от указанных аргументов и*

$$k_2\in R,\ \alpha_2=\sum_{\lambda=2}^{n}x^{\lambda^2}.$$

Отметим, что если функция H не зависит от x^1, то пространство гиперплоскостных элементов $H_{n,\underline{u}}$ второй лакунарности допускает транзитивную группу движений порядка $r=n(n-1)/2+1$. Здесь существует также еще один класс однородных пространств гиперплоскостных элементов $H_{n,\underline{u}}$. Это будут пространства $H_{n,\underline{u}}$, допускающие группу движений G_r порядка $r=n(n-1)/2+1$, являющуюся подгруппой группы движений n-мерного пространства Лобачевского. Метрические функции этих пространств находятся из формул

$$H=H(u_1,\ e^{-2k_1x^1}\underline{\gamma}_2),\ \det\|H^{\cdot j\cdot k}\|\neq 0, \tag{3}$$

где k_1 — ненулевая постоянная. Из формул (3) следует, что рассматриваемые пространства $H_{n,\underline{u}}$ являются функционально полуприводимыми.

Рассмотрим теперь пространства Дейвиса векторных плотностей второй лакунарности. В уравнениях движения

$$DF^w=v^\alpha\partial_\alpha F^w+u^\alpha\partial_\alpha v^\sigma\frac{\partial F^w}{\partial u^\sigma}-w\partial_\alpha v^\alpha u^\sigma\frac{\partial F^w}{\partial u^\sigma}=0 \tag{4}$$

появляются дополнительные слагаемые (так называемые весовые части). Интегрируя уравнения (4), получим

$$F^w=\bar{\gamma}_2^{w(n-1)}\psi(x^1,\ u^1,\ \bar{\gamma}_2/B^2), \tag{5}$$

где ψ — дифференцируемая функция от указанных аргументов, причем $\det\|F^w_{.j.k}\|\neq 0$. Следовательно, справедливо следующее предложение.

Теорема 6. *Метрические функции пространств Дейвиса векторных плотностей второй лакунарности с группой движений G_r порядка $r=n(n-1)/2$ определяются формулами* (5).

Если в формуле (5) функция ψ не зависит от x^1, то G_r будет транзитивной группой движений порядка $r=n(n-1)/2+1$.

Если метрическая функция F^w с той же группой движений G_r будет не обязательно однородной второй степени относительно переменных $u^1, u^2, \ldots, u^n$, то будем иметь

$$F^{\text{н}\,w}=\varphi[x^1,\ u^1B^{w(n-1)},\ \bar{\gamma}_2B^{2[w(n-1)-1]}],$$

$$\det\|F^{\text{н}\,w}_{.j.k}\|\neq 0,\ (j,\ k=1,\ 2,\ \ldots,\ n).$$

В случае пространств Дейвиса ковекторных плотностей второй лакунарности имеем

$$H^w = \underset{12}{\gamma}^{-w(n-1)}\psi(x^1,\ u_1,\ B^2\underset{12}{\gamma}).$$

Если метрическая функция пространств Дейвиса второй лакунарности будет не обязательно однородная второй степени относительно компонент опорного объекта $u_1, u_2, \ldots, u_n$, то находим, что

$$H^{nw} = \Phi[x^1,\ B^{w(n-1)}u_1,\ B^{2[1+w(n-1)]}\underset{12}{\gamma}].$$

В последних двух формулах функции ψ, Φ являются произвольными дифференцируемыми функциями от перечисленных аргументов, причем метрические тензоры рассматриваемых пространств невырожденные.

ЛИТЕРАТУРА

1. Кручкович Г. И. К вопросу о классификации римановых пространств по группам движений.— Тр. Всесоюзн. заочн. энерг. ин-та. М., 1963, т. 24, с. 25—37.
2. Тасиро Я. Теория групп преобразований в обобщенных пространствах и ее применение к финслеровым и картановым пространствам. I. Mat. Soc. Japan, 1959, 11, № 1, 42—71.

г. Пенза

Поступили
первый вариант 12.02.1985
окончательный вариант 12.11.1985

А. И. Еникеев, К. А. Р. Хоор, А. Терюэл УДК 519.68

МОДЕЛЬ ТЕОРИИ ВЗАИМОДЕЙСТВУЮЩИХ ПОСЛЕДОВАТЕЛЬНЫХ ПРОЦЕССОВ ДЛЯ МЕНЮ-ДИАЛОГОВЫХ СИСТЕМ

Сложность и многообразие систем программного обеспечения, существенно увеличившиеся в последние годы, приводят к необходимости качественного изменения используемых технологических средств их проектирования. Большинство этих технологических средств не обеспечивают решения таких принципиально важных проблем, как достижение высокой надежности программ, создание эффективных моделей, позволяющих выполнять полное исследование свойств программных систем до этапа их непосредственной разработки, и, наконец, строгое и компактное описание схем взаимодействия процессов для проектируемой системы. Перечисленные проблемы решаются путем использования формальных методов. Однако опыт показывает, что использование формальных методов в проектировании программных систем приводит к слишком громоздким выкладкам, что является серьезным препятствием на пути практического применения. Отсюда вытекает необходимость создания концептуального аппарата, обеспечивающего применение формальных методов в практике разработки программных систем. Наиболее подходящей с этой точки зрения является теория взаимодействующих последовательных процессов CSP (Communicating Sequential Processes) [1], которая наряду с концептуализацией последовательных процессов, обеспечивает возможность спецификации и анализа различных схем взаимодействия между этими процессами (включая параллелизм). Принципы спецификации и анализа процессов, положенные в основу CSP, адекватно согласуются с методом проектирования сверху — вниз, позволяя редуцировать сложность формальных методов разработки программ путем абстрагирования от всех несущественных для данного уровня проектирования деталей. Эффективность использования подобного аппарата может быть достигнута путем разработки программного инструментария для машинного проектирования программных систем требуемого класса и разумного сочетания формальных методов с интуицией программистов. В частности, в работе [2] предлагается заменять формальное доказательство корректности программ применением так называемых аргументов корректности, которые в случае

излишней громоздкости не доказываются, а формулируются и считаются верными исходя из опыта и интуиции программиста.

Одним из важных этапов строгого подхода к разработке программ (под строгим подходом в соответствии с работой [2] имеется в виду использование формальных методов) является создание модели проектируемой программной системы на основе подходящего концептуального аппарата. Данная статья посвящена построению модели меню-диалоговой системы на основе CSP. Предлагаются спецификации основных управляющих процессов меню-диалога. Рассматривается методика анализа процессов в CSP. Предлагается реализация указанных процессов на ЛИСП-подобном языке.

1. Модель меню-диалоговой системы

Одним из существенных требований, предъявляемых к диалоговым системам, является обеспечение удобного интерфейса с пользователем, который заключается в минимизации вводимых с терминала данных без потери мнемоничности обозначений, уменьшении количества ошибок при вводе информации и, самое главное, в быстрой адаптации пользователя к диалоговой системе. Такой интерфейс является важным особенно в тех случаях, когда диалоговая система эксплуатируется „случайным“ пользователем или вводимые в режиме диалога данные имеют сложную структуру. Существует много способов обеспечения указанного интерфейса, среди которых меню-диалог является одним из важных. Суть меню-диалога заключается в выдаче альтернативного списка элементов (меню), один из которых выбирается пользователем путем подвода курсора или нажатием соответствующей функциональной клавиши при диалоговом решении задачи на ЭВМ. Наиболее очевидные преимущества, достигаемые при использовании меню-диалоговых систем, следующие:

1) быстрая адаптация пользователя к диалоговой системе и практически полное исключение необходимости специального изучения системы;

2) минимизация вводимой с терминала информации и, следовательно, уменьшение вероятности появления ошибок.

Меню-диалоговое взаимодействие адекватно описывается множеством последовательностей выбираемых из меню элементов. Последнее позволяет использовать для описания меню-диалога средства CSP (краткий обзор средств CSP, используемых в данной статье, приводится в приложении). В CSP упомянутые выше последовательности называются следами, а каждый элемент последовательности определяет событие (напр., функцию, вызываемую к действию после выбора соответствующего элемента). *След* — это конечная последовательность символов в заданном алфавите. Содержательно след определяет один из возможных путей прохождения меню-диалогового взаимодействия от начала его функционирования до определенного момента времени, а каждый символ следа обозначает одно из событий. Например, $\langle a, b\rangle$ обозначает след, состоящий из событий a и b. Множество всех символов, обозначающих события, которые могут быть вызваны функционированием заданного процесса, называется *алфавитом процесса*. *Процесс* — это множество всех его следов, описывающих всевозможные пути его функционирования. Для любого процесса P с алфавитом A имеют место следующие соотношения:

P0. $P \subseteq A^*$, где A^* — множество всевозможных следов с символами из алфавита A (универсальное множество);

P1. $\langle\rangle \in P$, где $\langle\rangle$ обозначает пустой след;

P2. $st \in P \Rightarrow s \in P$ для всех s, $t \in A^*$ (st — конкатенация следа s со следом t). Пусть s — непустой след. Тогда через s_0 обозначается начальный символ следа s, а через s' — след, получившийся после удаления начального символа из следа s. Через P^0 (или initials (P)) обозначается множество начальных символов всех следов процесса P. Формально $P^0 \triangleq \{c \in A \mid \langle c\rangle \in P\}$ ($\triangleq$ — есть по определению).

Пусть $s \in P$. Тогда $P_{/s} \triangleq \{t \mid st \in P\}$ (P после s). Будем предполагать, что все рассматриваемые в статье процессы принадлежат к так называемым успешно завершаемым процессам (WTP-процессам, Well Terminating Processes) в том смысле, что они „ничего не делают“ после завершения. Формально:

$$P \in \mathrm{WTP} \Leftrightarrow (\forall (s \circ \langle\surd\rangle) \in P)(P_{/s \circ \langle\surd\rangle} = \mathrm{FAIL}),$$

где $s \circ \langle\surd\rangle$ — успешно завершаемый след процесса P, $\surd$ — обозначает признак завершения следа (процесса), FAIL $\triangleq \{\langle\rangle\}$ — пустой процесс. (Успешно завершаемым следом называется след, завершающийся символом $\surd$). Это предположение является важным для реализации процессов.

В CSP меню-диалоговое взаимодействие может быть специфицировано в виде процесса P, определяемого парой (I, F), где I — начальное меню (т. е. P^0), а F — функция, отображающая каждый элемент (событие) x из I в процесс $P_{/\langle x\rangle}$ (P после x), определяющий дальнейшее функционирование меню-диалога после выбора элемента x. В свою очередь каждый элемент меню определяет функцию, вызываемую к действию после соответствующего выбора пользователя. Среди этих функций выделим функции, предназначенные для управления меню-диалоговым процессом. Такие функции будем называть управляющими функциями.

К наиболее типичным управляющим функциям относятся:

1. функции для прерывания или завершения процесса (stop и др.);

2. функции, позволяющие возвратиться к предыдущим шагам выполнения процесса (reset — для восстановления начального меню, и back — для возврата к предыдущему меню);

3. функции-переключатели — для переключения с выполнения одного процесса на другой, в частности, для переключения на другую группу меню или ввод данных и команд, когда техника меню-выборки неприменима (off — для прямого и on — для обратного переключений). Модель меню-диалоговой системы основывается на спецификации перечисленных функций, интерпретируемых соответственно процессами stoppable (P), resetable (P), backtrackable (P) и coroutine (P, Q) (или $P_{\text{on/off}}Q$), где P, Q — процессы, управляемые упомянутыми функциями.

1.1. *Процессы управления меню-диалогом.*

1.1.1. Процесс stoppable. Пусть stop$\notin \alpha P$, где αP — алфавит процесса P. Тогда процесс stoppable(P) определяется как процесс, в котором все следы, не содержащие символа stop, принадлежат процессу P и:

1) stop $\in \alpha$ stoppable (P);

2) stop $\in$ (stoppable $(P)_{/s})^0$ для любого s такого, что $s \in$ stoppable (P) & & $s \wedge \{\surd\} = \langle\rangle$;

3) если некоторый след содержит символ stop, то stop является последним символом этого следа (т. е. выполнением события stop процесс stoppable завершает свою работу).

Пример. $\langle a, b, c,$ stop$\rangle \in$ stoppable $P \Leftrightarrow \langle a, b, c\rangle \in P$.

1.1.2. Процесс resetable. Пусть reset$\notin \alpha P$. Тогда процесс resetable (P) определяется как процесс, в котором все следы, не содержащие символа reset, принадлежат процессу P и:

1) reset $\in \alpha$ resetable (P);

2) $s \in$ resetable (P) & $s \wedge \{\surd\} = \langle\rangle \Rightarrow$
reset $\in$ (resetable $(P)_{/s})^0$ & resetable $(P)_{/s \circ \langle\text{reset}\rangle}$ = resetable (P);

3) $s \circ \langle\surd\rangle \in$ resetable (P) & $s \wedge \{\surd\} = \langle\rangle \Rightarrow$ resetable $(P)_{/s \circ \langle\surd\rangle}$ = FAIL.

Содержательно, после каждого следа, завершающегося символом reset, процесс resetable „стартует“ с самого начала.

Пример. $\langle a, b, c,$ reset, $b, c, d\rangle \in$ resetable $(P) \Leftrightarrow \langle a, b, c\rangle \in P$ & $\langle b, c, d\rangle \in P$.

1.1.3. Процесс backtrackable. Пусть back$\notin \alpha P$. Тогда процес backtrackable (P) определяется как процесс, в котором все следы, не содержащие символа back, принадлежат процессу P:

1) back $\in \alpha$ backtrackable (P);

2) $s \in$ backtrackable (P) & $s \wedge \{\surd\} = \langle\rangle \Rightarrow$ back $\in$ (backtrackable $(P)_{/s})^0$;

3) backtrackable $(P)_{/\langle\text{back}\rangle}$ = backtrackable (P);

4) $s \circ \langle a,$ back$\rangle \in$ backtrackable (P) & $s \wedge \{\surd\} = \langle\rangle$ & $a \notin \{$back, $\surd\} \Rightarrow$
backtrackable $(P)_{/s \circ \langle a,\ \text{back}\rangle}$ = backtrackable $(P)_{/s}$;

5) $s \circ \langle\surd\rangle \in$ backrtackable (P) & $s \wedge \{\surd\} = \langle\rangle \Rightarrow$ backtrackable $(P)_{/s \circ \langle\surd\rangle}$ = FAIL.

Содержательно после каждого следа, завершающегося символом back, процесс back'trackable возвращается к предыдущему состоянию.

Примеры. $\langle a, b, c,$ back, $d\rangle \in$ backtrackable $(P) \Leftrightarrow$
$\langle a, b, d\rangle \in P$ & $\langle a, b, c\rangle \in P$
$\langle a, b, c,$ back, back, $d\rangle \in$ backtrackable $(P) \Leftrightarrow$
$\langle a, d\rangle \in P$ & $\langle a, b, c\rangle \in P$

1.1.4. Coroutine. Пусть $(\alpha P \cup \alpha Q) \cap \{off, on\} = \varnothing$. Определим процесс coroutine (P, Q) следующим образом:

1. α coroutine $(P, Q) = \alpha P \cup \alpha Q \cup \{off, on\}$;

2. (coroutine $(P, Q)_{/s})^0 = (P_{/s})^0 \cup \{off\}$, для любого $s \in P$ и не содержащего символа „$\surd$“;

3. $(s \in P$ & $s \wedge \{\surd\} = \langle\rangle) \Rightarrow$ coroutine $(P, Q)_{/s}$ = coroutine $(P_{/s}, Q)$ &
coroutine $(P, Q)_{/s \circ \langle\text{off}\rangle}$ = coroutine$'$ $(Q, P_{/s})$;

4. $s \circ \langle\surd\rangle \in$ coroutine (P, Q) & $s \wedge \{\surd\} = \langle\rangle \Rightarrow$ coroutine $(P, Q)_{/s \circ \langle\surd\rangle}$ = FAIL.

Процесс coroutine$'$ (Q, P) определяется следующим образом:

1$'$. α coroutine$'$ $(Q, P) = \alpha$ coroutine (P, Q);

2$'$. (coroutine$'$ $(Q, P)_{/s})^0 = (Q_{/s})^0 \cup \{on\}$, для любого $s \in Q$ и не содержащего символа $\surd$;

3$'$. $(s \in Q$ & $s \wedge \{\surd\} = \langle\rangle) \Rightarrow$ (coroutine$'$ $(Q, P)_{/s}$ = coroutine$'$ $(Q_{/s}, P)$ &
coroutine$'$ $(Q, P)_{/s \circ \langle\text{on}\rangle}$ = coroutine $(P, Q_{/s})$;

4$'$. $s \circ \langle\surd\rangle \in$ coroutine$'$ (Q, P) & $s \wedge \{\surd\} = \langle\rangle \Rightarrow$ coroutine$'$ $(Q, P)_{/s \circ \langle\surd\rangle}$ = FAIL.

Этот процесс принадлежит так называемому классу сопрограмм и описывает такие известные процессы, как мультипрограммирование, режим разделения времени, совместную работу лексического и синтаксического анализаторов при компиляции и т. п.

Пример. $\langle a_1, a_2,$ off, $b_1, b_2, b_3,$ on, $a_3,$ off, $b_4, b_5\rangle \in$
coroutine $(P, Q) \Leftrightarrow \langle a_1, a_2, a_3\rangle \in P$ &
$\langle b_1, b_2, b_3, b_4, b_5\rangle \in Q$

1.2. *Пример меню-диалогового взаимодействия.* Для иллюстрации меню-диалогового взаимодействия рассмотрим систему машинного контроля при изучении сложных операторов (операторов, содержащих много параметров) в языках программирования. Рассматривается система контроля, предусматривающая выдачу списка альтернативных ответов, из которых обучаемый должен выбрать верный. Ниже приводится сеанс диалогового взаимодействия обучаемого с системой, который иллюстрируется на контроле изучения макрокоманды ввода-вывода DCB в ОС ЕС.

Кадр 0: СОСТАВЬТЕ ОПЕРАТОР „DCB“ ДЛЯ ПЕЧАТИ НА АЦПУ,
ИМЯ DD = SYSPRINT, ДЛИНА ЗАПИСИ = 128,
РЕЖИМ ПЕРЕСЫЛКИ.

Кадр 1: DSORG = ?

ПАРАМЕТРЫ	ФУНКЦИИ УПРАВЛЕНИЯ
(1) PS	(S) STOP
(2) IS	(R) RESET
(3) PO	(B) BACK
(4) DA	

Меню 1 = (PS, IS, PO, DA, STOP, RESET, BACK).
Выбор 1 : 1 (PS).

Кадр 2: MACRF = ?

ПАРАМЕТРЫ	ФУНКЦИИ УПРАВЛЕНИЯ
(1) GM	(S) STOP
(2) GL	(R) RESET
(3) PM	(B) BACK
(4) PL	

Меню 2 = (GM, GL, PM, PL, STOP, RESET, BACK).
Выбор 2:4 (PL) (выбор сделан неверно, т. к. должно быть PM)
Кадр 3: RECFM = ?

ПАРАМЕТРЫ	ФУНКЦИИ УПРАВЛЕНИЯ
(1) F	(S) STOP
(2) FA	(R) RESET
(3) FB	(B) BACK
(4) V	
(5) VB	
(6) U	

Меню 3 = (F, FA, FB, V, VB, U, STOP, RESET, BACK)
Выбор 3: B (BACK, для восстановления предыдущего меню; обучаемый обнаружил свою ошибку и решил вернуться к предыдущему меню).
Кадр 4: так же как кадр 2.
Выбор 4:3 (PM)
Кадр 5: так же, как кадр 3.
Выбор 5:2 (FA)
Кадр 6: EODAD = ?

ПАРАМЕТРЫ	ФУНКЦИИ УПРАВЛЕНИЯ
(1) BB — ввод EODAD?	(S) STOP
(2) IGNORE	(R) RESET
	(B) BACK

Меню 6 = (BB, IGNORE, STOP, RESET, BACK)
Выбор 6: S (STOP, обучающий не стал набирать другие параметры, напр DDNAME и т. п.).
Кадр 7: ВАШ ОТВЕТ: DCB DSORG = PS,
MACRF = PM,
RECFM = FA
ПРАВИЛЬНЫЙ ОТВЕТ: DCB DSORG = PS,
MACRE = PM,
RECFM = FA,
DDNAME = SYSPRINT,
BLKSIZE = 128

Функционирование процесса машинного контроля в данном случае опр де делено следом: ⟨PS, PL, BACK, PM, FA, STOP⟩. В приведенном примере н рассмотрены функции-переключатели, которые можно использовать в данно случае для переключения с одного сеанса контроля на другой и обратно переключения.

2. Спецификация и анализ процессов

В этом разделе рассматриваются спецификации и методика анализа упра ляющих процессов на основе средств CSP. Рассматриваются следующие сп собы определения процессов:
1) теоретико-множественный способ, при котором процесс определяет множеством его следов в заданном алфавите;
2) процедурный способ, при котором процесс P определяется парой объе тов (I, F), где
а) множество I определяет начальное состояние (т. е. $I = P^0$),
в) F — функция, которая отображает каждый элемент $c \in I$ в проце $P_{/\langle c\rangle}$ (P после c), определяющий дальнейшее функционирование процесса после выполнения события „c".
Первый способ является наиболее подходящим для доказательства свойс процессов, поскольку он позволяет применять при доказательствах достаточ

азвитый и хорошо исследованный аппарат теории множеств. Анализ свойств процессов составляет важную часть исследования модели разрабатываемой программной системы, поскольку удачно сформулированный список свойств позволяет строго обосновывать решения, принимаемые на последующих этапах разработки системы и ее модификации, существенно повышая эффективность используемого технологического аппарата. В частности, эти свойства могут служить основой доказательства корректности соответствующих программ.

Процедурный способ адекватным образом определяет способ реализации процесса. Такой способ определения позволяет реализовать процесс P на ЛИСПе в виде cons (I, F), где I — список, определяющий начальное состояние процесса P, а F — ламбда-выражение, показывающее как вычислять $P_{/\langle x\rangle}$ для каждого x из I.

Для доказательства корректности реализации процессов можно сформулировать список свойств, обосновывающих корректность процедурного определения по соответствующему теоретико-множественному определению. Разложение на список свойств позволяет уменьшить сложность формальной верификации процессов за счет расчленения верификации на доказательства отдельных свойств (подобно принципу модульности). С другой стороны, при громоздкости доказательства истинность соответствующих свойств может быть определена исходя из опыта и интуиции программиста.

Ниже приводятся формальные спецификации процессов resetable и backtrackable, иллюстрирующие указанные выше способы определения процессов. Методика анализа процессов иллюстрируется на процессе resetable.

2.1. *Спецификации.*

2.1.1. Определение resetable.

Определение 1 (теоретико-множественное).

$$\text{resetable}(P) \triangleq \{s \in (\alpha P \cup \{\text{reset}\})^* \mid (\forall s_1 \leqslant s)(\text{rest}(s_1) \in P)\},$$

где

а) $P \in \text{WTP} \,\&\, \text{reset} \notin \alpha P$

$$\text{rest}(s) = \begin{cases} s, & s \wedge \{\text{reset}\} = \langle\rangle \\ s_2, & s = s_1 \circ \langle\text{reset}\rangle \circ s_2 \,\&\, s_1 \wedge \{\surd\} = \langle\rangle \,\& \\ & s_2 \wedge \{\text{reset}\} = \langle\rangle \end{cases}$$

Определение 2 (процедурное).

$$\text{resetable}(P)_{\text{df}} = \text{start}(P,\ P),$$

$$\text{start}(P,\ Q)^0 = P^0 \cup \{\text{reset}\} \,\&$$

$$\text{start}(P,\ Q)_{/\langle a\rangle} = \begin{cases} \text{resetable}(Q), & a = \text{reset} \\ \text{FAIL}, & a = \surd \\ \text{start}(P_{/\langle a\rangle},\ Q), & a \in P^0 - \{\surd\} \end{cases}$$

Из процедурного определения непосредственно вытекает следующая реализация процесса resetable (P) на ЛИСП-подобном языке:

$$\text{resetable}(P)_{\text{df}} = \text{start}(P,\ P),$$

$$\text{start}(P,\ Q)_{\text{df}} = \lambda P,\ Q.\ \text{cons}(P^0 \cup \{\text{reset}\},$$

$$\lambda x.\ \textbf{if}\ x = \text{reset}\ \textbf{then}\ \text{resetable}(Q)\ \textbf{else}$$

$$\textbf{if}\ x = \surd\ \textbf{then}\ \text{FAIL}\ \textbf{else}\ \text{start}(P_{/\langle x\rangle},\ Q).$$

Второй аргумент функции start используется для сохранения начального состояния. Реализация функции start требует использования „задержанных вычислений", обеспечиваемых в данном случае специальной реализацией функции cons (см. [3]). Последнее относится также к реализации процесса backtrackable.

10) αP — алфавит процесса P.

11) $P^0 \triangleq \{c \mid \langle c \rangle \in P\}$ (начальное состояние процесса P).

12) $(c \to P) \triangleq \{\langle\rangle\} \cup \{(c \to s) \mid s \in P\}$.

13) $P \square Q \triangleq P \cup Q$ (альтернативная операция).

14) $P_{/s} \triangleq \{t \mid st \in P\}$ (P после s).

ЛИТЕРАТУРА

1. Hoare C. A. R. A model for Communicating Sequential Processes.— Technical monograph PRG — 22. Oxford University Computing Laboratory, PRG, 1981. 28 p.

2. Jones C. B. Software Development. A Rigorous Approach.— Prentice Hall International, 1980. 382 p.

3. Хендерсон П. Функциональное программирование. Применение и реализация. М., 1983. 349 с.

4. Лавров С. С. Синтез программ.— Кибернетика, 1982, № 6, с. 11—16.

5. Волож Б. Б., Мацкин М. Б., Минц Г. Е. и др. Система ПРИЗ и исчисление высказываний.— Кибернетика, 1982, № 6, с. 63—70.

г. Казань
Великобритания, г. Оксфорд
Венесуэла, г. Каракас

Поступила
9.09.1985