УДК 519.681

В.Г. Акуловский, А.Е. Дорошенко

ОПИСАНИЕ ПАРАЛЛЕЛИЗМА В АЛГОРИТМАХ ИНФОРМАЦИОННО-УПРАВЛЯЮЩИХ СИСТЕМ СРЕДСТВАМИ АЛГЕБРАИЧЕСКОГО АППАРАТА

Показана возможность описания в рамках алгебры алгоритмов с данными параллельных алгоритмов для класса информационно-управляющих систем. Введены средства синхронизации параллельно выполняемых ветвей алгоритма и продемонстрирована возможность построения производных средств синхронизации на основе введенных.

Введение

Широко распространенными в различных областях человеческой деятельности являются информационно-управляющие системы (ИУС). Об этих системах, исходя из многочисленных публикаций (например, [1]) и не взирая на различные классификации программных систем, можно сказать, что они сочетают функции автоматизированных систем управления (АСУ) с функциями систем автоматического управления (САУ). В силу специфики выполняемых функций, системы данного класса обычно реализуются в виде многомашинных систем, часто иерархических, подсистемы которых часто функционируют в многозадачном режиме.

Вопросы, связанные с описанием алгоритмов данного класса систем, в частности, связанные с описанием параллелизма, в рамках алгебро-алгоритмического подхода, рассмотрены недостаточно. Учитывая важность роли, которую играют ИУС в современном мире, рассмотрение данных вопросов является актуальной задачей.

Для решения данной задачи воспользуемся алгебраическим аппаратом, который построен в результате модификации известной модели ЭВМ Глушкова.

Модификация заключается в дополнении упомянутой модели внешней средой (ВС), которая состоит из программмной (ПС) и аппаратной (АС) составляющих.

Первая обеспечивает взаимодействие между моделями, функционирующими в различных ВС (многомашинная организация), и между задачами в рамках одной ВС (квазипараллельная, многозадачная организация). Вторая – представляет собой память и внешние устройства (ВУ), включающие как устройства ввода-вывода (УВВ), так и устройства связи с объектом управления (УСО), и интерпретируется как данные, формализованные следующим образом [2].

Определение 1. Данными называется пара $D=\langle N, 3 \rangle$, где N – носитель данных, З – кортеж значений, носителем которых является N. На каждом шаге вычислительного процесса носитель содержит (хранит) некоторый (текущий) кортеж значений данных, в частности, эти значения могут быть неопределенными.

С учетом определения 1, АС рассматривается как множество данных $D^{AC}=\{\{D^{\Pi}\},\{D^{BУ}\}\}$, где носитель D^{Π} – память, а носитель $D^{BУ}$ – ВУ. Эти данные, характеризующиеся в каждый момент времени некоторым множеством значений, которые изменяются в ходе вычислительного процесса.

Такая модификация модели ЭВМ позволила ввести Д-операторы вида $(D)O(D')$ со специфицированными данными D и D', которые они обрабатывают, т. е. анализируют и преобразуют, изменяя их значения. Частным случаем является Д-оператор $(D)Z(D')$, который не изменяет данные, называется тождественным и в дальнейшем часто записывается в виде Z.

Заметим, что значения специфицируемых данных в общем случае изменяются в ходе вычислительного процесса, а на некоторых его этапах могут быть не определены. Отсюда следует, что специфицируются, фактически, носители данных. Поэтому под теоретико-множественными операциями, определенными на множествах данных, понимаются операции над их носителями.

Специфицируемые данные, с учетом сделанного замечания, такие, что $D \subseteq D^{AC}$ и $D' \subseteq D^{AC}$.

По поводу специфицируемых данных в [2] введено определение.

Определение 2. Данные, специфицируемые на входе и выходе Д-оператора $(D)O(D')$, будем называть: D – входными или обрабатываемыми, D' – выходными или продуцируемыми. Для множеств специфицируемых данных допустимо как соотношение $(D \cap D') \neq \varnothing$ (в частности, $D \subseteq D'$ и $D' \subseteq D$), при котором множество данных $D \supseteq D'' = (D \cap D')$ изменяется в результате обработки, так и $(D \cap D') = \varnothing$, когда множество данных D остается неизменным. Специфицированные данные изменяются в результате и только в результате их обработки Д-операторами.

На основе модифицированной модели ЭВМ в работах [2–4] предложена система алгоритмических алгебр (САА/Д), представляющая собой трехосновную алгебраическую систему $<U, L, D, \Omega>$, основами которой являются множество Д-операторов U, множество логических условий L и множество данных D, а – её сигнатура, состоящая из Ω_1 – операций, принимающих значения на множестве U, Ω_2 – логических операций, принимающих значения на множестве L, Ω_3 – операций, принимающих значения на множестве данных D.

Из всего многообразия операций САА/Д, определенных на множестве Д-операторов, в данной работе воспользуемся следующими:

– **композиция Д-операторов** $(D_i)O_i(D_i') * (D_{i+1})O_{i+1}(D_{i+1}')$ (операция обозначается "*") означает последовательное выполнение сначала Д-оператора $(D_i)O_i(D_i')$ и затем Д-оператора $(D_{i+1})O_{i+1}(D_{i+1}')$;

– **р-итерация** $\langle (D^{\pi})P(\alpha) \rangle \{(D)O(D')\}$ состоит в вычислении предиката $(D^{\pi})P(\alpha)$, где D^{π} – вектор заданного отношения, и проверке логического условия $\alpha \in \{1, 0\}$, продуцируемого предикатом. Если α истинно, выполняется Д-оператор $(D)O(D')$ и вновь вычисляется предикат и проверяется условие α. Циклический процесс, состоящий в проверке условия α и выполнении Д-оператора, осуществляется до тех пор, пока условие $\alpha = 1$ в противном случае операция р-итерации завершается.

В частном случае данная операция, записана в виде

$$\langle 1 \rangle \{(D)O(D')\}, \qquad (1)$$

где 1 – тождественно истинное условие, реализует "бесконечный" цикл.

Представление любого исходного Д-оператора из U через образующие элементы системы $<U, L, D, \Omega>$ называется регулярной схемой данного Д-оператора (РСД). В частном случае, когда используется единственная операция – композиция, такое представление Д-оператора называется его композиционной схемой (КС).

Описанию параллелизма ИУС в рамках предложенного алгебраического аппарата и посвящена данная работа.

Описание параллелизма в алгоритмах ИУС

Прежде чем начать рассмотрение основного вопроса, определим алгоритм ИУС. Для этого сначала определим алгоритм в общем случае.

Определение 3. Алгоритмом называется Д-оператор $(D)A(D')$, на входе которого специфицированы все необходи-

мые для решения некоторой задачи глобальные данные, а на выходе – все результирующие глобальные данные.

При нисходящей стратегии проектирования [5] данные D и D' детализуются, а алгоритм декомпозируется и записывается в виде РСД или КС. В последнем случае это выглядит как

$$(D)\mathrm{A}(D') = (D_1)O_1\ (D_1') * \\ *(D_2)O_2(D_2') * ... * (D_k)O_k(D_k'), \quad (2)$$

где $D = D_1 \cup ... \cup D_k$, $D' = D_1' \cup ... \cup D_k'$, $(D_1)O_1(D_1'),...,(D_k)O_k(D_k')$ – производные Д-операторы, полученные в результате декомпозиции. На следующем шаге декомпозируются производные Д-операторы и детализуются специфицированные у них данные. Процесс продолжается до достижения требуемого уровня подробности описания алгоритма.

При этом производные Д-операторы взаимодействуют в процессе обработки данных. То есть, обработка множества данных, начатая некоторым Д-оператором $(D_i)O_i(D_i')$, может продолжаться одним или несколькими следующими за ним Д-операторами.

Учитывая это введем понятия связи между Д-операторами.

Определение 4. В РСД и КС Д-операторы $(D_i)O_i(D_i')$ и $(D_j)O_j(D_j')$ $(j>i)$ назовем информационно связанными (в дальнейшем связанными), если для специфицированных у них данных выполняется соотношение $D_i' \cap D_j \neq \varnothing$, в противном случае эти Д-операторы не связаны. Данные ${}_iD_j = (D_i' \cap D_j)$ назовем связывающими и на выходе i-го Д-оператора обозначим ${}_i\vec{D}_j$, на входе j-го ${}_i\overleftarrow{D}_j$.

Известно, что управляющие системы, в силу своей специфики, обычно функционируют неограниченное время (см., например, [6]), т. е. их работа прекращается только в случае отключения питания. Алгоритмы такого рода будем называть непрерывными и, используя (1), представим в виде РСД:

$$(D)A(D') = \\ = (D_{\text{п}})O_{\text{п}}(D_{\text{п}}')*\langle 1\rangle(D_{\text{т}})O_{\text{т}}(D_{\text{т}}'), \quad (3)$$

где 1 – тождественно истинное логическое условие, $O_{\text{п}}$ – выполняет подготовительные (стартовые) операции, $O_{\text{т}}$ – “тело” алгоритма.

С учетом (3), алгоритм ИУС определим следующим образом.

Определение 5. Алгоритм ИУС состоит из двух (или более) информационно связанных подсистем, реализующихся в виде непрерывного алгоритма и выполняются параллельно (квазипараллельно) и асинхронно. Под информационной связью понимается двусторонняя связь, т. е. некоторое подмножество данных, продуцируемых одной подсистемой, используется второй и наоборот. По крайней мере, одна из подсистем взаимодействует с УВВ, а одна с УСО. Любая из подсистем в свою очередь может состоять из параллельно (квазипараллельно) и асинхронно выполняемых подсистем (подзадач, процессов).

Дальше будем рассматривать, вариант ИУС, когда она состоит из двух подсистем, каждая из которых функционирует в собственной программно-аппаратной среде (двухмашинный случай).

Для случаев, когда подсистемы ИУС функционирует в различных программно-аппаратных средах, в сигнатуру алгебры включим операцию **асинхронная дизъюнкция Д-операторов**

$$(D_i)O_i(D_i') \dot{\vee} (D_j)O_j(D_j'),$$

состоящую в параллельном асинхронном выполнении Д-операторов.

Архитектуру системы, исходя из определения 5, используя (3) и введенную операцию, опишем в виде следующей РСД:

$$(D)ИУС(D') = ((D_{\text{п\_АСУ}})\Pi_{АСУ}(D_{\text{п\_АСУ}}')* \\ *\langle 1\rangle(D_{АСУ})АСУ(D_{АСУ}')) \dot{\vee} \\ \dot{\vee}((D_{\text{п\_САУ}})\Pi_{САУ}(D_{\text{п\_САУ}}')* \\ \langle 1\rangle(D_{САУ})САУ(D_{САУ}')),$$

где

$$D = D_{\text{п\_}АСУ} \cup D_{АСУ} \cup D_{\text{п\_}САУ} \cup D_{САУ},$$

$$D' = D'_{\text{п\_}АСУ} \cup D'_{АСУ} \cup D'_{\text{п\_}САУ} \cup D'_{САУ}.$$

Чтобы отобразить двухсторонние информационные связи между подсистемами, ограничимся подсистемами АСУ и САУ, которые, в соответствии с определением 4, запишем в виде следующего фрагмента алгоритма:

$$\langle 1\rangle({}_{САУ}\vec{D}_{АСУ}, D_{АСУ})АСУ(D'_{АСУ}, {}_{АСУ}\vec{D}_{САУ}))\dot{\vee}$$

$$\dot{\vee}\langle 1\rangle({}_{АСУ}\vec{D}_{САУ}, D_{САУ})САУ(D'_{САУ}, {}_{САУ}\vec{D}_{АСУ}).$$

Однако, полученная схема содержит противоречие, так как соотношения $D_{АСУ} \cap D'_{САУ} \neq \varnothing$ и $D'_{АСУ} \cap D_{САУ} \neq \varnothing$ не выполняются из-за того, что подсистемы функционируют в разных ВС.

Для того, чтобы устранить возникшее противоречие и решить проблему синхронизации, позаимствуем идею синхронизации в работе [7] и модернизируем её следующим образом. Полагая, что ВС предоставляет необходимые сервисы, построим механизм передачи сообщений посредством следующих элементарных Д-операторов:

– контрольная точка $(\vec{D})T(S)$, где $\vec{D}$ – передаваемые данные, S – синхронизатор, которому эти данные передаются;

– синхронизатор $(\vec{D})S(\text{a}_{\vec{D}})$ такой, что $(\vec{D})S(\text{a}_{\vec{D}}) = \langle 1\rangle\{Z\}$ пока $\vec{D} = \varnothing$ и $(\vec{D})S(\text{a}_{\vec{D}}) = Z(\text{a}_{\vec{D}})$ в противном случае, где Z – тождественный Д-оператор, $\text{a}_{\vec{D}}$ – указатель на полученные данные.

Неформально говоря, синхронизатор задерживает вычислительный процесс, до момента поступления связывающих данных и передает адрес полученных данных следующему за ним Д-оператору после их поступления.

Введенные средства синхронизации наряду с решением своей основной задачи устраняют возникшее противоречие, путем дополнения связывающих данных из определения 4 передаваемыми.

Однако, при попытке их использования

$$*\langle 1\rangle(({}_{САУ}\vec{D}_{АСУ})S_{АСУ}(\text{a}_{{}_{САУ}\vec{D}_{АСУ}})*$$

$$*(D_{АСУ})АСУ(D_{АСУ})*$$

$$*({}_{АСУ}\vec{D}_{САУ})T_{АСУ}(САУ))\dot{\vee}$$

$$\dot{\vee}\langle 1\rangle(({}_{АСУ}\vec{D}_{САУ})S_{САУ}(\text{a}_{{}_{АСУ}\vec{D}_{САУ}})*$$

$$*(D_{САУ})САУ(D_{САУ})*$$

$$*({}_{САУ}\vec{D}_{АСУ})T_{САУ}(АСУ)),$$

сталкиваемся с ограничением, вызванным тем, что подсистемы АСУ и САУ, в соответствии с определением 5, имеют двухсторонние связи. Из приведенной схемы видно, что параллельное выполнение подсистем невозможно, так как в результате взаимной синхронизации каждая из подсистем ожидает выполнения другой, что порождает “клинч”.

Ограничение снимается в результате декомпозиции алгоритмов подсистем. Поскольку вместе с декомпозицией осуществляется детализация данных, в том числе и связывающих, то при достаточной степени декомпозиции для каждой из подсистем будет получена такая детальность описания, при которой любой из входящих в это описание Д-операторов, имеет только односторонние связи. С достижением такого этапа проектирования средства синхронизации могут вводиться в описание алгоритма, что изобразим в виде следующим фрагментом схемы алгоритма:

$$...*({}_{САУ_c}\vec{D}_{АСУ_{p+1}})S_p(\text{a}_{{}_{САУ_c}\vec{D}_{АСУ_{p+1}}})*$$

$$*({}^{s}D_{АСУ_{p+1}})^{s}АСУ_{p+1}({}^{s}D'_{АСУ_{p+1}})*...$$

$$.....$$

$$...*({}^{p}D_{САУ_c})^{p}САУ_c({}^{p}D'_{САУ_c})*$$

$$*({}_{САУ_c}\vec{D}_{АСУ_{p+1}})T_{c+1}(S_p)*...,$$

где верхний левый индекс – номер шага (уровня) декомпозиции алгоритма.

Достигнутая степень параллелизма алгоритма, как правило, окажется недостаточной, так как подсистемы часто функционируют в многозадачной ПС.

Рассмотрим возможность распараллеливания для данного случая, т. е. возможности распараллеливания алгоритмов подсистем на некотором этапе их разработки. При этом будем учитывать, что для параллельно и асинхронно выполняемых Д-операторов ни продолжительность выполнения каждого из них, ни очередность начала и завершения этих Д-операторов не определены.

Рассмотрение начнем с ограничений на параллельное выполнение Д-операторов $(D_i)O_i(D_i')*(D_{i+1})O_{i+1}(D_{i+1}')$, входящих в КС (2). Заметим при этом, что определение 4, распространяется и на этот случай, а ${}_iD_{i+1}$ – связывающие данные.

Первым фактором, ограничивающим параллельное выполнение, являются информационные связи, определяющие жесткий порядок обработки данных, так как при наличии такой связи Д-оператор $(D_{i+1})O_{i+1}(D_{i+1}')$ продолжает обработку данных, начатую Д-оператором $(D_i)O_i(D_i')$, и, таким образом, параллельное их выполнение невозможно.

Для того чтобы учесть второй фактор, введем следующее определение.

Определение 6. Д-операторы $(D_i)O_i(D_i')*(D_{i+1})O_{i+1}(D_{i+1}')$ имеют между собой обратные связи, если выполняется соотношение $D_i \cap D_{i+1}' \neq \varnothing$, что приводит к изменению значений данных, специфицированных на входе $(D_i)O_i(D_i')$, соотношение $D_i' \cap D_{i+1}' \neq \varnothing$, выполнение которого приводит к изменению значений данных D_i', продуцируемых Д-оператором $(D_i)O_i(D_i')$.

Второй фактор вытекает из определения 6, так как обратные связи обусловливают влияние одного Д-оператора на результаты выполнения другого.

Для того чтобы рассмотреть третий фактор вспомним, что в качестве носителей данных могут выступать ВУ. Учитывая это вводимые и выводимые данные определим следующим образом, полагая, что в нашем распоряжении имеется некоторые множества ВУ R и W.

Определение 7. Данные, носителем которых являются некоторое ВУ $R_i \in R$, с которого хранимые значения переносятся (копируются) в память, будем обозначать D_i^R и называть вводимыми, и некоторое ВУ $W_i \in W$, в которое хранимые значения переносятся (копируются) из памяти, – D_i^W и выводимыми. Вводимые и выводимые данные специфицируются, соответственно, на входе и выходе произвольного Д-оператора. Множества D_i^R и\или D_i^W могут быть пустыми, т. е. вводимые и\или выводимые данные могут отсутствовать.

При наличии, в соответствии с определением 7, спецификаций вводимых и выводимых данных, Д-оператор, очевидно, осуществляет операции ввода-вывода. Это обусловливает наличие третьего фактора, которым является строго заданная в алгоритме последовательность операций ввода-вывода. Асинхронное выполнение Д-операторов, может привести к нарушению этой последовательности. Заметим, что в общем случае для произвольного алгоритма существует некоторое множество корректных (допустимых) последовательностей этих операций.

Будем полагать, что при параллельном выполнении Д-операторов корректность последовательности операций ввода-вывода в некоторых случаях будет сохраняться, а во многих случаях обеспечиваться средствами синхронизации параллельных процессов (средства синхронизации будут ниже рассмотрены). Кроме того будем полагать, что существует возможность проверки такой корректности, которая, в связи с отсутствием формализации, реализуется “вручную”.

Учитывая приведенные факторы, определим условия, при котором возможно параллельное выполнение Д-операторов. Для этого введем операцию асинхронного параллельного выполнения Д-операторов в многозадачном режиме в виде $\ddot{\vee}$.

Определение 8. Первым условием, необходимым для параллельного выполнения Д-операторов

$$(D_i)O_i(D_i') \ddot{\vee} (D_{i+1})O_{i+1}(D_{i+1}'),$$

является то, что в результате такого их выполнения значения всех обрабатываемых данных D_i, D_i', D_{i+1}, D_{i+1}', должны полностью совпадать со значениями тех же данных, полученных в результате их последовательной обработки композицией Д-операторов $(D_i)O_i(D_i') * (D_{i+1})O_{i+1}(D_{i+1}')$. Второе условие – допустимая последовательность операций ввода-вывода при параллельном выполнении Д-операторов.

Теперь докажем возможность параллельного выполнения Д-операторов, предварив это доказательство следующей леммой.

Лемма. В композиции Д-операторов $(D_i)O_i(D_i') * (D_{i+1})O_{i+1}(D_{i+1}')$, если они не связаны и не имеют обратных связей, то результатом её выполнения являются значения данных D_i, D_i' и D_{i+1}, D_{i+1}', полученные после выполнения соответствующего Д-оператора (O_i и O_{i+1}) и только этого Д-оператора.

Доказательство основывается на определениях 6 и 7.

Результатом выполнения композиции Д-операторов являются:

– значения данных D_i', … , в силу $D_i' \cap D_{i+1}' = \varnothing$;

– значения данных D_i', полученные после выполнения O_i и только O_i, так как эти значения не изменятся в результате выполнения O_{i+1}, в силу;

– значения данных D_{i+1}', полученные после выполнения O_{i+1} и только O_{i+1}, так как на эти значения, не зависят от результатов выполнения O_i, в силу $D_i' \cap D_{i+1} = \varnothing$;

– значения данных D_i и D_{i+1}, которые, в соответствии с определением 2, изменятся или остаются неизменными в результате выполнения Д-операторов, соответственно, O_i и O_{i+1}, но изменениям в результате выполнения, соответственно, O_{i+1} и O_i не подвергаются в силу $D_i \cap D_{i+1}' = \varnothing$ и $D_{i+1} \cap D_i' = \varnothing$.

Лемма доказана.

Теорема. В КС два любых последовательно выполняющихся Д-оператора могут выполняться параллельно $(D_i)O_i(D_i') \ddot{\vee} (D_{i+1})O_{i+1}(D_{i+1}')$ при тех условиях, что эти Д-операторы не связаны, не имеют обратных связей и последовательность операций ввода-вывода, если они имеют место, остается допустимой.

Доказательство основывается на определениях 6 и 7.

При любой последовательности запуска и завершения (последовательности выполнения) Д-операторов

$$(D_i)O_i(D_i') \ddot{\vee} (D_{i+1})O_{i+1}(D_{i+1}')$$

результатом их параллельного выполнения будут:

– значения данных D_i' и D_{i+1}', продуцируемые, соответственно, O_i и O_{i+1}, так как с одной стороны полученные значения не изменятся в результате выполнения, соответственно, O_{i+1} и O_i, поскольку $D_{i+1}' \cap D_i' = \varnothing$. С другой – значения данных D_i' и D_{i+1}' не влияют на выполнение O_{i+1} и O_i, поскольку $D_i' \cap D_{i+1} = \varnothing$ и $D_{i+1}' \cap D_i = \varnothing$;

– значения данных D_i и D_{i+1}, которые, в соответствии с определением 2, могут измениться или остаться неизменными в результате выполнения, соответственно O_i и O_{i+1}, но в результате выполнения, соответственно, O_{i+1} и O_i не изменятся, поскольку $D_i \cap D_{i+1}' = \varnothing$ и $D_{i+1} \cap D_i' = \varnothing$.

Таким образом, результаты параллельного выполнения Д-операторов $(D_i)O_i(D_i')$ и $(D_{i+1})O_{i+1}(D_{i+1}')$ совпадают с результатами их последовательного выполнения, полученными в лемме, что удовлетворяет первое условие из определения 8.

Если операции ввода-вывода отсутствуют или в результате проверки

выяснилось, что последовательность этих операций допустима, или в результате синхронизации этих операций была обеспечена такая последовательность, то, в соответствии с определением 8, выполняется второе условие, необходимое для параллельного выполнения Д-операторов.

Теорема доказана.

Следствие 1. Полученный результат может быть обобщен на случай более двух Д-операторов. Если условия теоремы распространяются на композицию из трех или более Д-операторов, то все они могут выполняться параллельно.

Следствие 2. Полученный результат может быть обобщен на случай последовательности Д-операторов

$$(D_j)O_j(D'_j)*(D_{j+1})O_{j+1}(D'_{j+1})*...$$

$$...*(D_p)O_p(D'_p),$$

поскольку в работе [8] показана возможность объединения Д-операторов. Если в результате объединения будет получен Д-оператор

$$(D)O(D') = (D_j)O_j(D'_j)*...*(D_p)O_p(D'_p),$$

который удовлетворяет условиям теоремы, то указанную последовательность Д-операторов, очевидно, можно выполнять параллельно как с Д-операторами, так и с другими последовательностями Д-операторов.

Из следствий к теореме видно, что связанные и имеющие обратные связи Д-операторы, которые не могут выполняться параллельно, могут входить в параллельно выполняемые последовательности Д-операторов. При этом число параллельно выполняемых Д-операторов и их последовательностей ограничивается только наличием связей между ними.

Очевидно, что и в рассматриваемом случае для распараллеливания алгоритма нужны средства синхронизации, которые введем как частный случай вышерассмотренных.

Синхронизатор – это элементарный Д-оператор $(\varnothing)S(\alpha)$ (где $\tilde{D} = \varnothing$) такой, что $(\varnothing)S(\alpha) = \langle\alpha\rangle\{Z\}$, где α – условие синхронизации истинное в исходном состоянии, Z – тождественный Д-оператор. С синхронизатором ассоциирована контрольная точка – элементарный Д-оператор $(\varnothing)T(S)$, где S – ассоциированный с этой контрольной точкой синхронизатор. При прохождении контрольной точки Д-оператор оповещает об этом ВС, которая передает ложное значение логического условия α синхронизатору. Синхронизатор, реализующий операцию р-итерации, получив ложное значение логического условия, прерывает циклирование. Синхронизатор может быть ассоциирован с несколькими контрольными точками $(\varnothing)T_1(S_i),...,(\varnothing)T_p(S_i)$, в этом случае он записывается в виде $(\varnothing)_{1,...,p}S_i(\alpha)$, а циклирование прекращается после прохождения всех контрольных точек.

Неформально говоря, синхронизатор задерживает вычислительный процесс, до момента прохождения вычислительным процессом контрольной точки или нескольких контрольных точек.

Отметим, что на введенные средства накладывается ограничение на взаимную синхронизацию аналогичное вышеописанному, которое аналогично и преодолевается.

Очевидно, что ограничения на использование средств синхронизации приведенным случаем не исчерпываются.

Учитывая разнообразие архитектур ИУС и специфичность решаемых ими задач продемонстрируем возможность построения производных средств синхронизации.

Композиции Д-операторов

$$(\varnothing)T_j(S_i)*(\varnothing)S_{j+1}(\alpha) = (\varnothing)TS_j(\alpha)$$

и

$$(\varnothing)S_i(\alpha)*(\varnothing)T_{i+1}(S_j) = (\varnothing)ST(S_j,\alpha)$$

порождают производные средства синхронизации (производные Д-операторы), позволяющие подтвердить факт получения информации синхронизатором S_i о прохождении контрольной точки T_j путем задержки вычислительный процесс в этой контрольной точке, посредством синхро-

низатора S_{j+1}, до поступления такого подтверждения. Отметим, что возможности построения производных средств синхронизации, рассмотренным случаем не исчерпываются.

Решив задачу распараллеливания в общем виде, сделаем некоторые уточнения, для чего, с учетом определений 2 и 7, введем следующее определение.

Определение 9. Вводимые $D^R = D^R \cup D^r$ и выводимые $D^W = D^W \cup D^w$ данные делятся на глобальные $(D^R \cup D^W) \subseteq D^G$ и локальные $(D^r \cup D^w) \subseteq D^L$. Глобальные в КС специфицируются на входе и выходе как исходного, так и любого производного Д-оператора. Локальные $D_i^r, D_i^w \subseteq D^L$, специфицируются только на входе и выходе производных Д-операторов.

КС, с учетом введенного определения, будет записана в виде:

$$(D, D^R)O(D', D^W) =$$

$$= (D_1, D_1^R, D_1^r)O_1(D_1', D_1^W, D_1^w)*...$$

$$...*(D_k, D_k^R, D_k^r)O_k(D_k', D_k^W, D_k^w).$$

Такая форма записи усложняет процесс описания алгоритма. Однако, при некоторой громоздкости, она обеспечивает возможность синхронизации операций ввода-вывода и контроль допустимости их последовательности, что является необходимым условием для распараллеливания алгоритма. Кроме того, спецификация вводимых и выводимых данных является необходимой при детальном описании алгоритма.

Задачи распараллеливания и синхронизации подсистем и выполнения Д-операторов, входящих в КС, решается совместно. При этом как в первом, так и во втором случае выбор момента введения средств синхронизации в описание алгоритма влияет на гранулярность параллелизма. Чем на более позднем этапе разработки это выполняется, тем большая мелкозернистость параллелизма обеспечивается.

Заключение

Показана возможность описания в рамках алгебры алгоритмов с данными параллельных алгоритмов для класса информационно-управляющих систем. Рассмотрены возможности распараллеливания, как подсистем, так и фрагментов алгоритмов внутри подсистем. Первый случай рассмотрен для двухмашинной, второй для многозадачной реализации. Для обоих случаев введены средства синхронизации параллельно выполняемых ветвей алгоритма.

Отметим, что передача сообщений – наиболее общее средство синхронизации, остальные – являются её частными случаями. При этом передача сообщений может использоваться не только в том случае, когда подсистемы функционируют в различных ВС, а и в случае многозадачной реализации в рамках одной ВС.

Продемонстрирована возможность построения производных средств синхронизации.

Рассмотрение ИУС с различной архитектурой и разработка средств синхронизации, ориентированных на специфику реализации таких систем, является ближайшей перспективой развития полученных результатов. Вопросы, связанные с оптимизацией процесс распараллеливания при ограничениях на степень зернистости параллелизма – направление дальнейших исследований.

1. *Пьявченко Т.А., Финаев В.И.* Автоматизированные информационно-управляющие системы. – Таганрог: Изд-во ТРТУ, 2007. – 271 с.
2. *Акуловский В.Г.* Алгебра алгоритмов, базирующаяся на данных // Кибернетика и системный анализ. – 2012. – № 2. – С. 151–166.
3. *Акуловский В.Г.* Основы алгебры алгоритмов, базирующейся на данных // Проблеми програмування. – 2010. – № 2–3. – С. 89–96.
4. *Акуловский В.Г.* Алгебра для описания данных в композиционных схемах алгоритмов // Поблеми програмування. – 2012. – № 2–3. – С. 234–240.

5. *Дорошенко А.Е., Акуловский В.Г.* Нисходящее проектирование алгоритмов в рамках алгероалгоритмического подхода // Математические машины и системы. – 2012. – № 3. – С. 97–102.
6. *Закревский А.Д.* Параллельные алгоритмы логического управления. – М.: Эдиториал УРСС, 2012. – 200 с.
7. *Ющенко Е.Л., Цейтлин Г.Е., Грицай В.П. и др.* Многоуровневое структурное проектирование программ: Теоретические основы, инструментарий. – М.: Финансы и статистика, 1989. – 208 с.
8. *Дорошенко А.Ю., Акуловський В.Г.* Висхідне проектування алгоритмів при алгеброалгоритмичному підході // Вісник Київського національного університету імені Тараса Шевченка. Серія фізико-математичні науки, 2012. – Вип. 1. – С. 167–172.

Получено 11.01.2013

Об авторах:

Акуловский Валерий Григорьевич,
кандидат технических наук,
доцент кафедры информационных систем и технологий,

Дорошенко Анатолий Ефимович,
доктор физико-математических наук,
профессор,
заведующий отделом
теории компьютерных вычислений.

Место работы авторов:

Академия таможенной службы Украины,
49000, г. Днепропетровск,
ул. Дзержинского 2\4.
E-mail: academy@amsu.dp.ua.
Тел. 050 941 05 66,
E-mail: valeryakulovskiy@rambler.ru

Институт программных
систем НАН Украины.
Тел. (044) 526 3559.
E-mail: dor@isofts.kiev.ua