g4144_s107	26.04.16

Статья. Библиографические данные:

Мунасыпов Р.А., Житников А.П.
Структурные формулы параллельных алгоритмов робототехнических систем.
// Сб. матер. междунар. научно-практ. форума: Современные технологии преподавания естественно-научных дисциплин в системе общего и профессионального образования.
– Борисоглебск: ООО "Кристина и К", 2016.
– С. 73-85.

СТРУКТУРНЫЕ ФОРМУЛЫ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

СОДЕРЖАНИЕ

Введение 1
Структурные формулы алгоритмов 1
Структурные формулы алгоритмов базового структурного класса 2
Значение алгоритмов базового структурного класса 3
Определение алгоритмов базового структурного класса 3
Расширения исходных определений алгоритмов 7
Литература 12

СТРУКТУРНЫЕ ФОРМУЛЫ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

Мунасыпов Р.А., Житников А.П.
Уфимский государственный авиационный технический университет

Введение

Статья ориентирована на учителей и преподавателей образовательной робототехники и информатики общеобразовательной, начальной, средней и высшей профессиональной школы [1].

Кратко (в упрощенной форме) представлен ***синтаксис структурных формул*** параллельных (и последовательных) алгоритмов ***базового структурного класса*** для технических, технологических и робототехнических приложений алгоритмов. Указываются направления расширения базового структурного класса алгоритмов. Отражается взаимосвязь структурных формул со структурными схемами алгоритмов (блок-схемами, штрих-схемами и т.п.), а также с разного типа псевдокодами алгоритмов и, в частности, типа учебного (школьного) алгоритмического языка.

Возможен самостоятельный выбор содержания и методов преподавания параллельных алгоритмов школьникам, а также студентам первых курсов. В частности, не целесообразно пытаться "загнать" изучение всей алгоритмической проблематики в ограниченное число часов учебного времени – "лучше меньше, да лучше". Но целесообразно при этом наглядное обзорное ознакомление аудитории с общей проблематикой в целом и с конкретными примерами типовых структур параллельных алгоритмов:
в сопровождении с демонстрациями работы роботов и робототехнических систем, их физических и программных моделей – начиная с первого же обзорно-установочного занятия (с периодическими повторами и в развитии).

Структурные формулы алгоритмов

Вводятся структурные формулы алгоритмов (СФА), параллельных и последовательных. В исходную основу принимаются так называемые ***логические схемы алгоритмов*** (ЛСА), положившие начало формирования прикладной (структурной) теории алгоритмов (1953 г.), и их расширение – ***параллельные логические схемы алгоритмов*** (ПЛСА) [2]. На самом деле это особые стрелочные ***структурные формулы*** алгоритмов с "поперечными" стрелками нелинейных переходов между операторами алгоритмов.

Например – простая (последовательная) ЛСА (СФА = ЛСА):

$A_{10} = Z_1Z_2p_1\uparrow^1Z_3\downarrow^1Z_4$.

При выполнении логического условия (предиката) p_1 (p_1= 1) выполняется последовательность операторов $Z_1Z_2Z_3Z_4$. Но если условие p_1 не выполняется (p_1= 0), то выполняется последовательность операторов $Z_1Z_2Z_4$ – в обход оператора Z_3 по стрелкам $\uparrow^1 \downarrow^1$ условного нелинейного перехода.

ЛСА в общем случае достаточно громоздки, особенно ПЛСА, и они не получили широкого практического распространения в программировании. Это связано с таким достаточно ***противоречивым*** обстоятельством:

- ЛСА – это изначально ***высокоуровневые средства*** алгоритмического описания программ на ранних этапах развития программирования – машинные коды, первые задачи системного программирования:
это ***программы низкого системного уровня*** и затем даже еще ниже – микропрограммы выполнения отдельных команд в микропрограммировании;
- они обеспечивают точное ***высокоуровневое*** отображение структуры ***низкоуровневых*** программных и микропрограммных структур – по потоку исполнения или потоку управления (порядком выполнения операторов);
- такая "врожденная" целевая ***низкоуровневая структурная ориентация*** затрудняет их прямое применение для алгоритмического описания высокоуровневых программных структур, особенно, параллельных:
появляется загромождение алгоритмов бесчисленными простыми и сложными условными и безусловными стрелочными переходами (пример далее).

Необходимы более компактные ***высокоуровневые синтаксические средства*** для описания ***высокоуровневых*** программных ***структур***. Однако при этом целесообразно:

- не "забраковывать" ЛСА (и ПЛСА), а обеспечивать на их основе точную низкоуровневую интерпретацию высокоуровневых синтаксических средств;
- реализовать при этом естественным образом глубокую ***объективную*** причинную первооснову высокоуровневых структур (без необходимости прямого аксиоматического задания их видов и свойств "из головы" и т.п.).

Это перспективная, но, тем не менее, достаточно проблемная задача (большой "семантический разрыв" между верхними и нижними уровнями).

Структурные формулы алгоритмов базового структурного класса

Алгоритмы базового структурного класса – это двухполюсные структуры (с одним входом и одним выходом по потоку управления) без условных ветвлений и циклов. Например, далее приводятся две СФА:

$A100 = (Z0 \rightarrow Z1 \rightarrow Z2 \rightarrow Z3) = Z0–Z1–Z2–Z3 = Z_0Z_1Z_2Z_3$

задается последовательное (во времени) выполнение команд Z0..Z3 – это линейный алгоритм, причем последняя форма записи СФА – это линейная ЛСА;

$A200 = (Z0 \rightarrow (Z1 \parallel Z2) \rightarrow Z3) = Z0–(Z1 \parallel Z2)–Z3 = Z_0(Z_1 \parallel Z_2)Z_3$

задается последовательное выполнение трех частей (участков, секций) Z0, (Z1 || Z2), Z3 алгоритма, во второй части задается параллельное выполнение (совмещение выполнения во времени) двух команд Z1, Z2.

Для последней СФА приводится ПЛСА (в упрощении ее записи):

$Z_0(Z_1 \parallel Z_2)Z_3 = Z_0R\uparrow^1\uparrow^2 \downarrow^1Z_1\uparrow^3 \downarrow^2Z_2\uparrow^4 \downarrow^3\downarrow^4SZ_3$

это сложное, но точное выражение низкоуровневой реализации простого параллельного алгоритма, где $R\uparrow^1\uparrow^2$ и $\downarrow^3\downarrow^4 S$ – структурные операторы распараллеливания и соединения потоков управления.

Оба алгоритма представляют собой алгоритмические двухполюсники:

A100 = →Z0→Z1→Z2→Z3→	A200 = →Z0→(Z1 || Z2)→Z3→

они имеют одно начало (один вход) и одно окончание (один выход) по потоку исполнения или потоку управления.

Значение алгоритмов базового структурного класса

Далее приводится определение класса таких двухполюсных структур алгоритмов произвольно большой величины и двухполюсной сложности. Это самый простой структурный класс последовательных и параллельных алгоритмов, но ключевой базисной значимости:

- это широко распространенный класс алгоритмов, который имеет большое самостоятельное практическое и теоретическое значение;
- такие алгоритмы часто используются как общий каркас для вставки в них фрагментов алгоритмов других структурных классов или сами они включаются в алгоритмы других структурных классов;
- на их основе могут определяться все прочие структурные классы алгоритмов методом поэтапного расширения структурных определений;
- на основе алгоритмов этого структурного типа решается многочисленная и многоаспектная алгоритмическая проблематика общего значения, или четко и конкретно ставятся задачи на обобщение разных аспектов.

Определение алгоритмов базового структурного класса

Далее приводятся определения структурных формул алгоритмов базового структурного класса, включая:

1) Определение ***исходных обозначений*** – первичных (Табл. 1) и дополнительных (по ходу последующих определений: Табл. 2 – Табл. 7).

2) ***Индуктивное определение*** – поэтапное определение ***правильно построенных формул*** (ППФ) двухполюсных алгоритмов базового структурного класса, включая этапы:

БИ: ***База индукции*** – определение ППФ первичных двухполюсных структур для последующих построений составных структур (Табл. 2).

ШИ: ***Шаг (шаги) индукции*** – пошаговые определения ППФ всевозможных новых составных двухполюсных структур (Табл. 3 – Табл. 5). Это задается последовательным и параллельным объединением ранее определенных двухполюсных структур.

ЗИ: ***Заключение индукции*** – подведение итогов определения ППФ алгоритмов базового структурного класса (Табл. 6) с возможностью расширения видов

производных ППФ в пределах базового структурного класса. В частности, задается уточнение ППФ параллельных структур (Табл. 7).

Записи типа E =. Ai или E =. Ai' обозначают:
выражение E есть по определению (=.) ППФ двухполюсного (Ai) или многополюсного (Ai') алгоритма.

В практических выкладках знак =. (равно по определению) может заменяться на простой знак равенства (=) с обращением записей типа:
Ai = E, Ai' = E – имя алгоритма Ai или Ai' обозначает тело E двухполюсного или многополюсного алгоритма.

Знак .=. в выражениях типа o(A1, A2) .=. (A1oA2) .=. (A1, A2)o означает разные эквивалентные между собой аффиксные формы записи некоторой структурной операции "o" (префиксная, инфиксная, постфиксная форма). Они обозначают одну и ту же структуру. Но это разные языки записи формул, их нельзя смешивать в составе ППФ алгоритмов.

Табл. 1. Первичные УЛО: Условные литерные обозначения полюсников операторов

$Zi = Zi^{11} = \rightarrow Zi = Zi\rightarrow = \rightarrow Zi\rightarrow$ $Ai = Ai^{11} = \rightarrow Ai = Ai\rightarrow = \rightarrow Ai\rightarrow$	Двухполюсники операторов алгоритмов Ai и команд Zi (по потоку управления): навешивание (и удаление) внешних связей
$Zi = \rightarrow Zi\rightarrow = -{>}Zi-{>} = –Zi–$ $Ai = \rightarrow Ai\rightarrow = -{>}Ai-{>} = –Ai–$	Разные способы явного представления внешних связей операторов (по потоку управления)
Zi = (Zi) = [Zi] = Zi Zi = (Zi) = –┤–Zi–├–> = = –┼–Zi–┼–> = –|–Zi–|–>	Навешивание (или удаление) внешней скобочной или контурной оболочки операторов команд Zi. Для операторов алгоритмов Ai аналогично.
$Ai' = Ai^{mn}$ $Ai' = Ai^{nn}$	Многополюсники операторов алгоритмов: m входов, n выходов, m, n ≥ 1 (по потоку управления) в общем случае. В частности: m = n. В частности, для m = n = 1, $Ai' = Ai^{11}$ - двухполюсник

Вырожденные ***пустые и единичные алгоритмы*** имеют первичное синтаксическое значение для построения всех других синтаксических структур. Кроме того, они имеют вспомогательное практическое значение:

- это могут быть работающие (на проход) ***заготовки*** в начале разработки и отладки программ и подпрограмм:
разработку и отладку программ целесообразно начинать с пустых и единичных алгоритмов и их программной реализации (сразу решается и отлаживается много вспомогательных вопросов общего значения);
- это могут быть так называемые ***заглушки*** вызовов подпрограмм при разработке программ по нисходящей технологии "сверху вниз":

сразу допускается запуск программ при отладке (с выводом на печать вызовов подпрограмм, если Zi – операторы контрольного вывода на печать).

Табл. 2. БИ: База индукции. Первичные (вырожденные) алгоритмы

( ) =. Ai = Ai^{11}	Ai = ( ) = ▯	Пустая (скобочная или контурная) оболочка есть ППФ (двухполюсного) алгоритма: пустой алгоритм – две интерпретации (1, 2)
1. Ai = ( ) = (−) = = − (−) −> = −▯−>		Постоянно замкнутая перемычка: свободный пропуск передачи управления // Псевдокод алгоритма: alg Ai: begin end
2. Ai = ( ) = (|) = − (−|−) −>		Постоянно разомкнутая перемычка: модель зависания передачи управления
(Zi) =. Ak = Ak^{11}	Ak = (Zi) = = −(−Zi−)−> = −▯−Zi−▯−>	Оператор команды в (скобочной или контурной) оболочке есть ППФ (двухполюсного) алгоритма: единичный алгоритм (по числу команд)

Табл. 3. Шаг индукции 0. Комплект (двух или более) двухполюсных алгоритмов

СФА: Структурная формула алгоритма		
ОМФ: Одномерная форма – запись строкой		
(A1, A2) =. Ai^{22} = Ai'	Ai' = (A1, A2)	Если A1, A2 есть ППФ двухполюсных алгоритмов
то список (двух) операторов в оболочке (A1, A2) есть ППФ многополюсного (четырехполюсного) алгоритма Ai' = Ai^{22} (m + n = 2 + 2 = 4)		
(A1, A2, …, An) =. Ai^{nn} = Ai'		Обобщение записи многополюсных алгоритмов (n ≥ 2)
ДМФ: Двухмерная форма = ШСС: Шаблон структурной схемы алгоритма		

```
Ai' = (A1,A2) = (A1) = -(-A1)-> = -[-A1-]-> =
                (A2)   -(-A2)->   -[-A2-]->
   +----+                ┌────┐
= -|-A1-|-> = ──┼──A1──┼──>
  -|-A2-|->   ──┼──A2──┼──>
   +----+                └────┘
```

Выражения типа **−▯−>** и **−▯-Zi-▯−>** базы индукции – это шаблоны для построения схем алгоритмов (блок-схем, штрих-схем и т.п.). Одновременно это особые схемы пустых и единичных алгоритмов, представленные в литерной псевдографике.

В конце определений по шагам индукции 0 и 1 задаются шаблоны построения структурных схем (ШСС) алгоритмов. Одновременно это также особые схемы алгоритмов, представленные в литерной псевдографике:
двухмерная (ДМФ) и одномерная (ОМФ) форма записи СФА.

Табл. 4. Шаг индукции 1. Секвенция (последовательная связь) операторов алгоритмов

$\rightarrow Ai^{nn} = Ak = Ai^{11}$	Общее обозначение операции секвенции (для n-комплекта операторов)
СФА: Структурная формула алгоритма	
$\rightarrow(A1, A2) .=. (A1 \rightarrow A2) =. Ai = Ai^{11}$	Секвенция для комплекта двух или более операторов – их последовательная связь
$\rightarrow(A1, A2, \ldots, An) .=. (A1 \rightarrow A2 \rightarrow \ldots \rightarrow An) =. Ai = Ai^{11}$	
Все аффиксные формы записи секвенции операторов	
ПрФ: Префиксная форма $\rightarrow(A1, A2)$	ИнФ: Инфиксная форма $(A1 \rightarrow A2)$
ПоФ: Постфиксная форма $(A1, A2)\rightarrow$	
$\rightarrow(A1, A2) .=. (A1 \rightarrow A2) .=. (A1, A2)\rightarrow =. Ai$	
Если A1, A2 есть ППФ двухполюсных алгоритмов, то указанные аффиксные формы (ПрФ, ИнФ, ПоФ) являются ППФ двухполюсного алгоритма. Интерпретация: это разные эквивалентные формы записи одной и той же последовательной структуры алгоритма.	
$\rightarrow(A1, A2, \ldots, An) .=. (A1 \rightarrow A2 \rightarrow \ldots \rightarrow An) .=. (A1, A2, \ldots, An)\rightarrow =. Ai$	
Аналогично в общем случае – для комплекта двух или более операторов ($n \geq 2$)	
ШСС: Шаблон структурной схемы алгоритма / ОМФ: Одномерная форма	
`Ai = (A1→A2) = -(-A1-A2-)-> =` `= -[-A1-A2-]-> = -`<u>`-A1-A2-`</u>`-> `	

Табл. 5. Шаг индукции 2. Параллель (параллельная связь) операторов алгоритмов

$\|\|Ai^{nn} = Ak = Ak^{11}$	Общее обозначение операции параллели (для n-комплекта операторов)
$\|\|(A1, A2).=. (A1 \,\|\|\, A2) =. Ai = Ai^{11}$	Параллель для комплекта двух или более операторов: это параллельная их связь
$\|\|(A1, A2, \ldots, An).=. (A1 \,\|\|\, A2 \,\|\|\, \ldots \,\|\|\, An) =. Ai = A^{11}$	
Все аффиксные формы записи параллели операторов	
ПрФ: Префиксная форма $\|\|(A1, A2)$	ИнФ: Инфиксная форма $(A1 \,\|\|\, A2)$
ПоФ: Постфиксная форма $(A1, A2)\|\|$	
$\|\|(A1, A2) .=. (A1 \,\|\|\, A2) .=. (A1, A2)\|\| =. Ai$	
Если A1, A2 есть ППФ двухполюсных алгоритмов, то указанные аффиксные формы (ПрФ, ИнФ, ПоФ) являются ППФ двухполюсного алгоритма. Интерпретация: это разные эквивалентные формы записи одной и той же параллельной структуры.	
$\|\|(A1, A2, \ldots, An) .=. (A1 \,\|\|\, A2 \,\|\|\, \ldots \,\|\|\, An) .=. (A1, A2, \ldots, An)\|\| =. Ai$	
Аналогично в общем случае – для комплекта двух или более операторов ($n \geq 2$)	

Табл. 6. Заключение индукции. Структуры базового структурного класса алгоритмов

Перечислены все исходные формы записи структур базового структурного класса алгоритмов. Других исходных форм записи структур этого класса нет (не определены).
Обобщенные записи ППФ однородных многоместных структур алгоритмов для краткости изложения введены сразу: формально нестрогий их вывода из исходных определений двухместных ППФ, но понятно.
На основе исходных форм могу вводиться производные эквивалентные им формы записи: малоскобочные формы, компактные свертки длинных однородных записей и т.п.

Далее (Табл. 7) приводится структурное уточнение исходной записи (A1 || A2) параллельного соединения операторов алгоритмов (и команд). В конце также задается шаблон построения структурной схемы (ШСС) алгоритма. Одновременно это также особая схема алгоритмов, представленная в литерной псевдографике [1]:
двухмерная (ДМФ) форма записи СФА, причем операция "o" соединения потоков интерпретируется как конъюнкция или дизъюнкция o = &, V [1].

Табл. 7. Уточнение структуры параллельной связи операторов

Дополнительные УЛО: Условные литерные обозначения	
|| = #o	Парная операция параллельной связи операторов
# – вилка (fork)	Операция разделения (дивергенции) потоков
o – сборка (join)	Операция соединения (конвергенции) потоков
СФА: Структурная формула алгоритма	
ОМФ: Одномерная форма – запись строкой	
ИнФ: Инфиксная форма	ПрПоФ: Префиксно-постфиксная форма
Ai = (A1 || A2) = (A1 #o A2) .=. #(A1, A2)o	
Ai = (A1 || A2 || … || An) = (A1 #o A2 #o … #o An) .=. #(A1, A2, …, An)o	
ДМФ: Двухмерная форма = ШСС: Шаблон структурной схемы алгоритма	
Ai = (A1||A2) = (A1#oA2) = = #(A1,A2)o = = #(A1)o = -#|-A1-|o-> // o = &, V (A2) |-A2-|	

Расширения исходных определений алгоритмов

Строго говоря, был определены ***нормальные двухполюсники*** Ai^{11}:
есть один и только одни вход (m = 1), один и только один выход (n = 1) по потоку (передачи) управления порядком выполнения действий.

При этом в шаге индукции 0 были определены комплекты алгоритмов, представляющие собой многополюсники $Ai^{mn} = Ai^{nn}$ (m = n ≥ 1). Именно на их основе определены нормальные двухполюсники алгоритмов.

Возможны разные расширения исходных определений на разные классы алгоритмических полюсников (систематически полюсники в информатике и их классификации рассматриваются в работе [3]):

1) Расширения определений в пределах класса постоянных (жестких, не переключаемых) структур – без условных ветвлений и циклов:

1.1) ***Особые*** двухполюсники:
Ai^{02} – 0 входов (m = 0) и 2 выхода (n = 2) по передаче управления:
исток, источник, генератор двух потоков управления;
Ai^{20} – 2 входа (m = 2) и 0 выходов (n = 0) по передаче управления:
сток 2-х потоков управления.

1.2) ***Однополюсники*** Ai^{01} (исток, источник, генератор одного потока управления), Ai^{10} (сток одного потока управления), Ai^{01} и ***нульполюсники*** Ai^{00} (замкнутая автономная система).

1.3) Возможны в частности по п.4 (далее), но в пределах постоянных (жестких, не переключаемых) структур.

2) Расширения определений на ***переключаемые (переменные, гибкие) структуры***. Например:

2.1) Структуры типа **if then**: // **I**f p **T**hen A

$(Ip?TA) = (IpTA) = (p\uparrow^1 A\downarrow^1) = ((p \to A) \vee (Np \to R)) = (pA \vee NpR)$,
где N – оператор инверсии (логической операции отрицания Не, Not);
R – пустой оператор-повторитель (репитер): y = R(x) = Rx = x;
$\vee$ – оператор дизъюнкции (логической операции Или, Or).

2.2) Структуры типа **if then else**: // **I**f p **T**hen A_1 **E**lse A_2

$(IpTA_1EA_2) = (p\uparrow^1 A_1\, \omega\uparrow^2\downarrow^1 A_2\downarrow^2) = ((p \to A_1) \vee (Np \to A_2)) = (pA_1 \vee NpA_2)$,
где $\omega\uparrow^i$ – оператор безусловного перехода по адресу $\downarrow^i$ (типа goto):
$\omega = p = const \equiv 0$ – нулевой предикат (постоянно ложное условие).

Примечание. Для разных языков программирования возможны разные варианты записи таких синтаксических конструкций:

$(IpTA) = IpTA$

$(IpTA_1EA_2) = IpTA_1EA_2$

допустимо полное отсутствие внешней оболочки (типа **begin end**);

$(IpTA) = IpTA)$

$(IpTA_1EA_2) = IpTA_1EA_2)$

оператор if (I) играет роль начальной (открывающей) скобки (типа **begin**), закрывающая скобка заменяет служебными словами типа **fi**, **endif**, **end**.

3) Расширение определений на циклические структуры (кратко):

$(Wp_i?A_k) = (Wp_iA_k) = (\downarrow^m p_i \uparrow^j A_k \omega \uparrow^m \downarrow^j)$ – цикл типа **W**hile;

$(Tp_i?) = (Tp_i) = \downarrow^j p_i \uparrow^J$ – структуры типа Wai**t**: ждать (сигнал) pi = 1 и т.п.

4) Вводятся ***приведенные двухполюсники*** – применяется приведение многополюсников к условным двухполюсникам:

- определяется основная пара полюсов – основной вход и основной выход алгоритма (возможно фиктивные) – по ним проводятся структурные операции как с обычными двухполюсниками:

$Ai = \rightarrow Ai \rightarrow$;

- прочие выходы и входы определяются как побочные:

$Ai = \rightarrow Ai\ (\uparrow\uparrow\uparrow, \downarrow\downarrow\downarrow) \rightarrow$

На этой основе возможно описание любых классов структур алгоритмов (путем комбинирования двухполюсников). И это универсальное средство для первичного (стрелочного) описания взаимодействующих потоков, асинхронных конвейерных режимов робототехнических систем [4] и т.п. На этой основе могут вводиться разные специальные обозначения для дополнительных входов и выходов взаимодействия потоков управления:
по аналогии с реализацией алгоритмов в разных парадигмах программирования и с разной служебной лексикой взаимодействия процессов.

Псевдокоды параллельных и последовательных алгоритмов

Псевдокоды алгоритмов – это промежуточная форма между структурными формулами и схемами алгоритмов и исходными кодами программ. Она ориентирована на лексику и синтаксис разных языков программирования (при этом сколько существует языков, столько может быть и псевдокодов с их дополнительными модификациями).

Псевдокоды могут быть получены на основе разных синтаксических (аффиксных) форм структурных формул алгоритмов простыми подстановками обозначений. Далее приводятся примеры псевдокодов (для примеров алгоритмов складского робота из предыдущей статьи сборника).

Табл. 8. Исходные примеры алгол-подобных псевдокодов (связь с учебным языком)

ПсАлг: Последовательный алгоритм / ИнФ: Инфиксная форма записи	
СФА: Структурная формула алгоритма = ШПК: Шаблон псевдокода	
A11 = (Z0–Z1– Z2–Z3)	
ПКА: Псевдокод алгоритма / АлгПТ: Алгол-подобный текст	
Стиль: Algol-68 / Латиница: En	Стиль: Алгол-68 / Кириллица: Ru
alg A11: begin Z0**;** Z1**;** Z2**;** Z3 end	алг A11: нач Z0**;** Z1**;** Z2**;** Z3 кон
алг A11: **нач** загрузка тары**;** горизонт_ход**;** верт_ход**;** разгрузка тары **кон**	
ПрАлг: Параллельный алгоритм / ИнПрФ: Инфиксно-префиксная форма	
СФА: Структурная формула алгоритма = ШПК: Шаблон псевдокода	
A2 = (Z0–(Z1 || Z2)–Z3) /=. (Z0–||(Z1**,** Z2)–Z3)	
ПКА: Псевдокод алгоритма / АлгПТ: Алгол-подобный текст	
Стиль: Algol-68 / Латиница: En	
alg A11: begin Z0**;** par begin Z1**,** Z2 end; Z3 end	
Стиль: Алгол-68 / Кириллица: Ru	
алг A11: нач Z0**;** пар нач Z1**,** Z2 кон**;** Z3 кон	
алг A11: **нач** загрузка**; пар нач** гор_ход**,** верт_ход **кон;** разгрузка **кон**	

Табл. 9. Примеры псевдокодов оккам-подобного типа

ПрАлг: Параллельный алгоритм / ПрФ: Префиксная форма	
СФА: Структурная формула алгоритма = ШПК: Шаблон псевдокода	
A2 = (Z0–(Z1 || Z2)–Z3) .=. –(Z0**,** ||(Z1**,** Z2)**,** Z3)	
ПКА: Псевдокод алгоритма / ОкПТ: Оккам-подобный текст	
alg A11: seq begin Z0**,** par begin Z1**,** Z2 end**,** Z3 end	// – ~ seq, || ~ par
алг A11: пос нач Z0**,** пар нач Z1**,** Z2 кон**,** Z3 кон	
алг A11: **пос нач** загрузка**, пар нач** гор_ход**,** верт_ход **кон,** разгрузка **кон**	
В вертикальной записи ПКА (с отступами строк) begin, end, нач. кон исключаются.	

Табл. 10. Дополнительный алгол-подобный пример (связь с учебным языком)

ПрАлг: Параллельный алгоритм ИнПрПоФ: Инфиксно-префиксно-постфиксная форма
СФА: Структурная формула алгоритма = ШПК: Шаблон псевдокода
A2 = (Z0–(Z1 || Z2)–Z3) = (Z0–(Z1 || Z2)–Z3) = (Z0–(Z1 #o Z2)–Z3) = = (Z0–#(Z1, Z2)o–Z3)
ПКА: Псевдокод алгоритма / АлгПТ: Алгол-подобный текст
Стиль: Algol-60 / Латиница: En
alg A11: begin Z0**;** parbegin Z1**,** Z2 parend**;** Z3 end
Стиль: Алгол-60 / Кириллица: Ru
алг A11: нач Z0**;** парнач Z1**,** Z2 паркон**;** Z3 кон
алг A11: **нач** загрузка**; парнач** гор_ход**,** верт_ход **паркон;** разгрузка **кон**

Псевдокод в стиле языка Алгол-68 (Табл. 8) для последовательного алгоритма – это есть учебный (школьный) алгоритмический язык (УАЯ), для параллельного алгоритма – это простое (в данном случае) его расширение. Дополнительно приведены псевдокоды:
в стиле языка Алгол-68 (Табл. 10) – соотношение с УАЯ аналогичное;
в стиле языка Оккам (Табл. 9) – для демонстрации аффиксных структур.

В целом в статье представлена обширная первичная информация для обеспечения систематического теоретико-алгоритмического анализа задач параллельного программирования в области образовательной робототехники. В статье [1] определяются задачи отражения логики механизмов управления в структурах алгоритмов технического назначения.

Литература

1. Житников А.П. Логико-временная интерпретация параллельных алгоритмов робототехнических систем. // Пропедевтика формирования инженерной культуры учащихся в условиях модернизации российского образования [Электронный ресурс]: сборник статей. – М. : БИНОМ. Лаборатория знаний, 2015. – С. 63-83.
http://paralg.ucoz.com/g4140/v5-g4144-s102-log_vrem_interpre.pdf

2. Лазарев В.Г., Пийль Е.И. Синтез управляющих автоматов. – М.: Энергоатомиздат, 1989. – 328 с.

3. Зверев Г.Н. Теоретическая информатика и ее основания. В 2 т. Т. 1. – М.: Физматлит, 2007. – 592 с.

4. Житников А.П. Параллельные и конвейерные алгоритмы робототехнологических модулей. // Proceedings of the 2nd International Conference “Intelligent Technologies for Information Processing and Management”, Volume 2, November 10-12, Ufa, Russia, 2014, pp. 159-165.
http://paralg.ucoz.com/g4140/v5-g4144-s101-parconv_alg_RTM.pdf