УДК 004.89

МЕТОДЫ И СРЕДСТВА АВТОМАТИЧЕСКОГО СИНТЕЗА ПАРАЛЛЕЛЬНЫХ ПРОГРАММ НА БАЗЕ ТЕОРИИ СТРУКТУРНЫХ ФУНКЦИОНАЛЬНЫХ МОДЕЛЕЙ

Д.А. Коваленко

Томский политехнический университет
E-mail: msmaklaud@gmail.com

Рассматриваются методы и алгоритмы автоматического синтеза параллельной программы по заданным непроцедурным спецификациям. Исследуются подходы к синтезу параллельных программ с использованием ярусно-параллельной формы графа алгоритма. Приводится вычислительная сложность предлагаемых алгоритмов. Рассматриваются вопросы практической реализации разработанных подходов.

Введение

Со времени появления первой ЭВМ сфера их применения охватила практически все области человеческой деятельности. Наиболее важным остается использование их в том направлении, для которого они собственно и создавались, а именно, для решения больших задач, требующих выполнения «громадных» объемов вычислений. Такие задачи возникли в середине прошлого века в связи с развитием атомной энергетики, авиастроения, ракетно-космических технологий и ряда других областей науки и техники. При решении подобного рода задач инженеры и ученые зачастую сталкиваются с проблемами вычислительной сложности. Даже самые быстрые однопроцессорные вычислительные системы не могут справиться с обработкой «больших» объемов, данных за «разумно короткое» время.

Сфера применения многопроцессорных вычислительных систем постоянно растет. Таких систем с 70-х гг. прошлого века было разработано много, однако при работе с ними пользователи столкнулись с рядом проблем, одной из которых является процесс разработки параллельных программ.

В [1] автор отмечает, что множество проблем параллельного программирования не имеют и в наши дни хорошего решения. Отладка и сопровождения параллельных программ является весьма сложным делом и помимо этого возникают ситуации, когда долго и правильно работающая программа вдруг выдает ошибочные результаты. Основной причиной возникающих проблем он считает программирование межпроцессорных коммуникаций. В работах [2] тоже отмечаются проблемы, связанные с разработкой параллельных программ. Причиной тому называют отсутствие четкой формальной модели параллельного алгоритма и дефицит математических аппаратов. Для частичного устранения данной причины вводятся такие понятия, как граф алгоритма, ярусно-параллельная форма графа алгоритма, граф-машина. Но на практике эти аппараты не решают многих проблем, например, они никак не учитывают время передачи данных между процессорами и не решают проблему отладки.

Проблемы, связанные с разработкой параллельных программ, можно решить двумя способами. *Первый*, это направленная подготовка специалистов по разработке параллельных программ, однако этот процесс достаточно трудоемкий, длительный и дорогостоящий. *Второй*, это разработка системы, которая бы дала возможность строить модели предметных областей на их основе автоматически синтезировать параллельные программы. Очевидно, что второй способ решения этой проблемы более привлекателен.

1. Существующие подходы к синтезу параллельных программ

Среди работ по синтезу параллельных программ необходимо отметить труды проф. В.Э. Малышкина [3], ИВМ и ГМ, г. Новосибирск. Идея его заключается в том, чтобы из базы знаний существующих реализованных параллельных алгоритмов, конструировать новые параллельные алгоритмы решения более крупных задач. В данном подходе предполагается наличие в базе знаний хороших алгоритмов, однако не приводится расшифровки того, что значит «хороший» алгоритм. Недостатком этого подхода является то, что он не может гарантировать наличия в изначальной базе алгоритмов эффективно распараллеленные алгоритмы, т. к. базовые алгоритмы реализуются вручную. Нельзя не отметить тот факт, что в базе может иметься более одного алгоритма решения одной и той же задачи и при синтезе более крупного блока из нескольких вариантов может быть выбран один.

Еще одно крупное достижение в области синтеза параллельных программ по непроцедурным спецификациям – это язык НОРМА [4]. НОРМА – декларативный язык для спецификации задач вычислительного характера. Непроцедурный язык НОРМА предназначен для записи численных методов решения задач математической физики разностными методами. Полученные формулы программируются на некотором языке, который обеспечивает решение задачи на мультипроцессорной вычислительной машине.

В подходах, разработанных в [5], предлагается модель пространственно-распределенной дискретной системы, описывающая совместные свойства параллельного алгоритма численного моделирования и аппаратуры.

Все перечисленные подходы хорошо проработаны, однако обладают существенным недостатком, коим является ограниченность области задач, для которых могут быть получены алгоритмы, например, язык НОРМА предназначен для решения уравнений математической физики. В случае структурного синтеза параллельных программ [3] исходный набор алгоритмов все же должен реализовываться человеком вручную, что вносит определенную вероятность наличия ошибки.

2. Основные понятия

В качестве базовой теории используется хорошо разработанная теория структурных функциональных моделей (ТСФМ) и алгоритм вывода [6]. Данная теория хорошо подходит для решения задачи синтеза последовательных рекурсивных программ. Посмотрим, каким образом ТСФМ можно использовать для построения параллельных программ. В ходе рассмотрения будем опираться на понятия ТСФМ.

Для того, чтобы ТСФМ можно было использовать для построения схем параллельных программ, необходимо правило, которое позволяет строить *параллельные* предложения вычислимости (ПВ). Пусть дано некоторое множество исходных величин A, два ПВ F_1 и F_2, которые по A доставляют множества величин X и Y соответственно. Тогда, если $X \cap Y = \varnothing$ и $X \not\subset A$ и $Y \not\subset A$, то F_1 и F_2 будем называть *независимыми* и обозначать их как $F_1 \| F_2$ (т. е. $F_1|_I$ и $F_2|_I$ могут быть выполнены параллельно). Построенное таким образом предложение вычислимости назовем *параллельным* ПВ. Определим правило тотального параллелизма (правило ТП) и запишем его в следующем виде:

$$\frac{F_1 : A \to X \quad F_2 : A \to Y}{F_1 \| F_2 : A \to X, Y}.$$

Добавим данное правило в исчисление ***SR*** и назовем расширенное исчисление $\boldsymbol{SR}_{par}$ Для исчисления $\boldsymbol{SR}_{par}$ справедливы две следующие теоремы.

Теорема 1. *Исчисление $\boldsymbol{SR}_{par}$ является корректным относительно понятия предложение вычислимости.*

Теорема 2. *Исчисление $\boldsymbol{SR}_{par}$ является полным относительно понятия предложение вычислимости.*

Далее введем несколько понятий.

Определение 1. Величину $a \in A$ будем называть *строго исходной величиной* некоторого множества предложений вычислимости Ψ, если и только если она является лишь аргументом какого-либо ПВ $F \in \Psi$ и не является результатом ни одного $F \in \Psi$.

Определение 2. *Выражение вида*

$$\begin{array}{l} F_1 : A_1 \to X_1 \\ \ldots\ldots\ldots\ldots \\ F_i : A_i \to X_i \\ \ldots\ldots\ldots\ldots \\ F_m : A_m \to X_m \end{array} \quad , \qquad (1)$$

будем называть линейный участок.

В то случае, когда все ПВ в линейном участке являются ФС, т. е.

$$\begin{array}{l} f_1 : A_1 \to x_1 \\ \ldots\ldots\ldots\ldots \\ f_i : A_i \to x_i \\ \ldots\ldots\ldots\ldots \\ f_m : A_m \to x_m \end{array} \quad , \qquad (2)$$

то такой линейный участок будем называть *простым* или *неделимым*.

Определение 3. Длиной линейного участка l будем называть количество ПВ в линейном участке.

Будем говорить, что некоторое F_i является *подпредложением* вычислимости ПВ F в том случае, когда F_i является составляющей компонентой F и отражать данный факт следующим образом $Fi \sqsubset F$.

Скажем так же и то, что каждый линейный участок, как отдельный алгоритм, может быть представлен в виде *графа алгоритма* и приведен к *ярусно-параллельной форме* (ЯПФ) [2].

Определение 4. *Матрицей зависимостей* графа алгоритма линейного участка из m ПВ будем называть такую квадратную матрицу $m \times m$, где $a_{ij}=1$, если дуга из X_j идет в F_i.

3. Подход с использованием ярусно-параллельной формы графа алгоритма

Поиск независимых ПВ в линейном участке будем сводить к построению ЯПФ графа алгоритма линейного участка и назовем *расстановкой* ПВ по ярусам ЯПФ.

Рассмотрим два алгоритма расстановки по ярусам.

Идея первого, назовем его «Анализ матрицы зависимостей», состоит в том, что алгоритм последовательно анализирует строки матрицы зависимостей заданного линейного участка, и относит ПВ соответствующее одной из строк, которая удовлетворяет некоторым критериям, к какому-либо ярусу ЯПФ. Входом его соответственно является матрица зависимостей заданного линейного участка.

На вход второму алгоритму, назовем его «Решение в лоб», поступает непосредственно линейный участок. Его идея заключается в сравнении пар вход-выход для каждого фиксированного ПВ с парами вход-выход всех остальных ПВ данного линейного участка.

Алгоритмы описаны с использованием некоторого псевдокода, конструкции которого совпадают с конструкциями большинства императивных языков программирования. Перед описанием каждого алгоритма будем пояснять используемые обозначения.

Как для первого, так и для второго алгоритма m – номер яруса, M_m – множество, эквивалентное m-му ярусу ЯПФ рассматриваемого линейного участка.

Алгоритм расстановки по ярусам «Анализ матрицы зависимостей»: n – количество строк и столбцов (количество ПВ в линейном участке $n=l$); c – количество упорядоченных строк и столбцов; k – количество единиц в i-ой строке при $c<j\le n$; $col1$, $col2$ – наименьший и наибольший номера строк матрицы, упорядоченных на предыдущим этапе; p – количество неупорядоченных строк; B – имеющаяся матрица зависимостей; *КолЕд*(i) – количество единиц в i-й строке матрицы B при $c<j\le n$; *ИмеютсяНужЕд*(i) – предикат, значение которого есть ИСТИНА в том случае, когда $col1=col2=0$ или когда в i-й строке матрицы B при $col1\le j\le col1$, $\exists bij=1$.

$c=0$
$m=0$
$p=n$
$col1=0$
$col2=0$
while p≠0 ***do***
  $i=c+1$
  $M_m=\varnothing$
  while $i\le n$ ***do***
    $k=$*КолЕд*(i)
    if $k=0$ ***and*** *ИмеютсяНужЕд*(i) ***then***
      поменять местами строки с номерами i и $c+1$
      поменять местами столбцы с номерами i и $c+1$
      добавить ПВ, соотв. данной строке в M_m
      $c=c+1$
      $p=p-1$
    fi
    $i=i+1$
  od
  $m=m+1$
  $col1=col2+1$
  $col2=c$
od

Алгоритм расстановки по ярусам. «Решение в лоб»: n – количество предложений вычислимости в линейном участке; A – множество аргументов текущего ПВ; X – множество результатов текущего ПВ; Y, Y_T – множество всех результатов, упорядоченных на данном этапе и на данный момент ПВ соответственно; p – количество оставшихся неупорядоченных ПВ; *Out* – множество результатов всех ПВ включенных в линейный участок; k – количество рассматриваемых ПВ на принадлежность к m-му ярусу.

$m=0$
$p=n$
$k=p$
$Y=\varnothing$
$M_m=\varnothing$
$Y_T=\varnothing$
while $p\ne0$ ***do***
  while $k\ne0$ ***do***
    if $((A\cap Y\ne\varnothing)$ ***or*** $(Y=\varnothing))$ ***and*** $(A\cap Out\setminus X=\varnothing)$ ***then***
      добавить текущее *ПВ* в M_m
      удалить его из набора рассматриваемых
      $Y_T=Y_T\cup X$
      $p=p-1$
    fi
    $k=k-1$
  od
  $k=p;$
  $m=m+1;$
  $Y=Y_T$
  $Out=Out\setminus Y_T$
  $Y_T=\varnothing$
  $M_m=\varnothing$
od

Для приведенных алгоритмов справедливы следующие теоремы.

Теорема 3. *Алгоритм «Анализ матрицы зависимостей» расстановки по ярусам корректен, т.е. всегда останавливается и обеспечивает нахождение канонической формы ЯПФ графа линейного участка.*

Теорема 4. *Алгоритм «Решение в лоб» расстановки по ярусам корректен, и позволяет найти каноническую ЯПФ.*

Рассмотрим вопросы эффективности предлагаемых алгоритмов, каждый из которых обладает своими преимуществами и недостатками. Так, например, для первого необходим препроцессинг в виде построения матрицы зависимостей, для второго характерна пространственная сложность алгоритма, обусловленная необходимостью построения множества *Out*.

Вычислительная сложность алгоритма «Анализ матрицы зависимостей», без учета препроцессинга, в худшем случае составляет $O(n^2)$, где n – количество строк и столбцов матрицы. Сложность алгоритма «Решение в лоб» составляет $O(l^2)$, однако, как уже было сказано, данный алгоритм обладает высокой пространственной сложностью.

Далее рассмотрим задачу получения ПВ, в котором найдены все независимые ПВ, т. е., построены все имеющиеся параллельные ПВ.

Процесс построения ПВ, в котором найдены все независимые ПВ, будем называть *распараллеливанием*, а ПВ, полученное путем распараллеливания, – *максимально параллельным* ПВ.

Прежде чем приступить к описанию метода решения, вспомним, что любая F_i может представлять ФС, линейный участок, композицию некоторых ...;F_j;..., оператор ветвления *if P then* F_1 *else* F_2

fi где *P* некоторый предикат, оператор рекурсии h=*if P then* F_1 *else* $F_2[h/g1]...[h/gs]$ *fi*. Это позволяет разбить некоторое ПВ на подпредложения вычислимости, в свою очередь полученные подпредложения разбить на другие подпредложения и так далее, пока не будут достигнуты ФС.

Алгоритм приведенный в [6] позволяет доказать достижимость искомых атрибутов по указанным исходным, а так же получить некоторое ПВ, посредством которого искомые атрибуты могут быть получены. В свою очередь это ПВ состоит из последовательности других ПВ F_i, которая в контексте данной работы соответствует выражению (1). В свою очередь каждая из ПВ F_i состоит из другой последовательности ПВ. И так далее пока не будут достигнуты простые линейные участки. Разработаем алгоритм *распараллеливания*, который позволит нам найти все операции, которые могут быть выполнены на многопроцессорной вычислительной машине независимо друг от друга.

Алгоритм распараллеливания.

На вход алгоритму подается некоторое ПВ *F*, представляющее линейный участок длинной *l*.

Будем использовать следующие обозначения.

Распараллелить(*F*) – выполнить расстановку ПВ составляющих *F* по ярусам.

Элементарная(F_i) – предикат, который означает, что F_i является элементарным (неделимым), т.е. такое ПВ уже нельзя разбить на некоторое количество других ПВ.

Будем говорить, что *F* имеет сложную структуру тогда и только тогда когда оно не является элементарным и $\exists Fi \sqsubset F$, что *Fi* – не элементарное ПВ.

```
if F имеет сложную структуру then
    Распараллелить(F)
    i=1
    while i≤m do
        if not Элементарная(Fi) then
         Распараллелить(Fi)
        fi
        i=i+1
    od
else
    Распараллелить(F)
fi
```

Для приведенного алгоритма справедлива следующая теорема.

Теорема 5. *алгоритм распараллеливания корректен, т. е. обеспечивает построение максимально параллельного ПВ.*

Алгоритм имеет рекурсивную природу. На верхнем уровне рекурсии распараллеливается линейный участок верхнего уровня, затем для каждой вершины ЯПФ опускаемся на следующий уровень рекурсии и распараллеливаем линейный участок на уровне ниже и так, пока не будут распараллелены все простые линейные участки (2).

В случае распараллеливания операторов ветвления и рекурсии имеются свои особенности. Рассмотрим особенности работы с операторами ветвления и рекурсии. Пусть мы имеем две посылки правила ТП $F_1{:}A \to X$ и *if P then* F_2 *else* F_3 *fi*:$A \to Y$ второе предложение вычислимости состоит из двух ПВ F_2 и F_3, где F_2 имеет смысл при условии *P*, а F_3 имеет смысл при условии $-P$. Применяя правило ТП получаем $F_1[$ *if P then* F_2 *else* F_3 *fi*$] : A \to X, Y$. Здесь заранее неизвестно значение *P*. Именно поэтому оператор ветвления необходимо рассматривать как отдельное ПВ со своими исходными и искомыми атрибутами. Задача распараллеливания решается отдельно для каждого ПВ из альтернативных ветвей. Получаем следующее ПВ *if P then* $F_1 \| F_2$ *else* $F_1 \| F_3$ *fi*:$A \to X, Y$. Внутренняя структура F_1 и F_2 рассматриваются отдельно. При наличии двух посылок правила ТП $F_1{:}A \to X$ и *if P then* F_2 *else* $F_3[h/g1]...[h/gs]$ *fi*:$A \to Y$ поступаем аналогичным образом.

4. Вопросы реализации

Рассмотрим вопросы реализации системы автоматического синтеза параллельных программ.

Главной программой в рассматриваемой системе, её основой, реализующей основную функцию системы, является программа синтеза параллельных программ (ПСПП).

ПСПП реализует алгоритм построения спецификаций параллельных программ (программа построения спецификации параллельной программы ПППрП) и дальнейшую трансляцию полученных спецификаций в параллельные программы. Как оговаривалось ранее, на вход алгоритму построения параллельной программы подается спецификация последовательной программы, которая может быть получена при помощи алгоритмов, разработанных в [6]. Задача на синтез и информация о структурной модели (С-модели) подается на вход программе построения спецификации последовательной программы (ПППсП), затем полученная спецификация подается на вход программе построения спецификации параллельной программы, после чего данная спецификация подается на вход программе трансляции.

В свою очередь ПППсП и программа трансляции представляют многомодульную структуру.

ПППсП состоит из программы вывода, программы чистки доказательства (ПЧД) и программы построения спецификации последовательной программы (ППСПсП). Программа вывода позволяет доказать достижимость искомых величин по заданным исходным, получить доказательство достижимости искомых величин, а так же велечин которые нас не интересуют. Поэтому доказательство получается избыточным. Получить необходимые нам части доказательства помогает ПЧД.

Программа трансляции представляет собой программу, которая состоит из множества программ трансляторов. Это необходимо для поддержания межплатформенности получаемых параллельных программ. Например, нам необходимо получить программу для суперЭВМ, которая использует интерфейс MPI. А затем нам необходимо выполнить те же расчеты для суперЭВМ, поддерживающей OpenMP. Для того чтобы можно было получить программу для MPI и для OpenMP ,нам нужно две программы-транслятора. Поэтому программа трансляции состоит из некоторого количества трансляторов T_i, с каждым из которых связан свой набор интерпретирующих модулей. Результатом каждого T_i является параллельная программа на конкретном языке программирования PP_i. Структурная схема ПСПП изображена на рисунке.

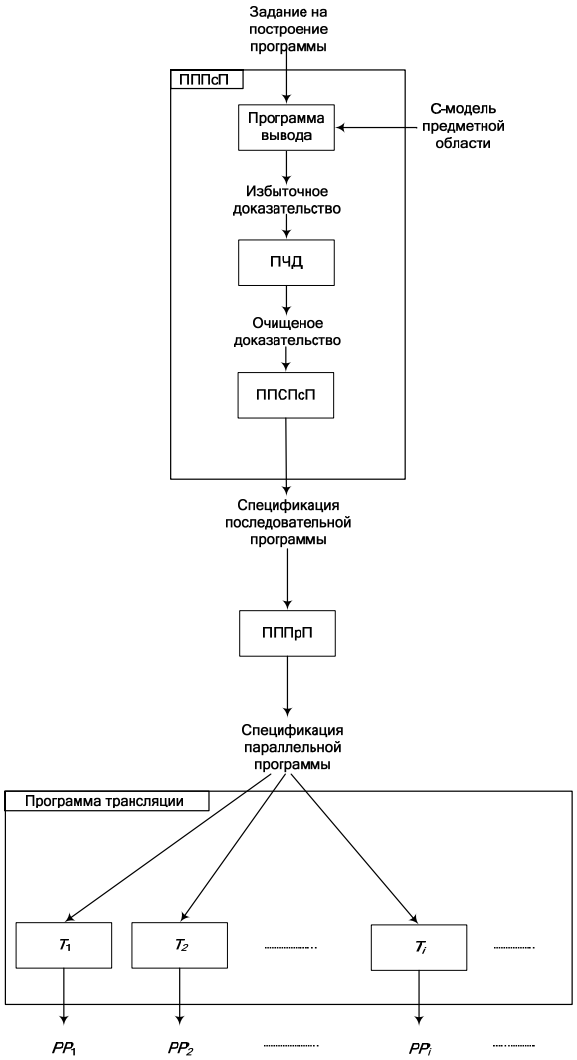

Рисунок. *Структурная схема программы синтеза параллельных программ*

С-модель в системе представляет набор взаимосвязанных XML-документов. В каждом из них содержится описания части модели, т. е. совокупности схем, объединенных по некоторым разумным соображениям. Такое представление модели облегчает анализ схем при синтезе, который сводится к анализу узлов дерева XML-документа. Ниже приведен пример описания схемы на языке XML.

```
<scheme>
   <name_scheme>число_фиб</name_scheme>
   <const_part>
      <attribute>
         <name_atr>n</name_atr>
         <type_atr    KindType="elem">int
         </type_atr>
      </atribute>
   ..................................
   </const_part>
   <selection_part>
   <cond_part>
      <condition>n=0</condition>
      <atribute> ..... </atribute>
   </cond_part>
   ..........................
   </selection_part>
   ..........................
   <function_part>
   <function_point>
      <name_func>I</name_func>
      <args>
         <argument>"1"</argument>
      </args>
      <result>fa</result>
   </function_point>
   ..................................
</scheme>
```

Для взаимодействия с пользователем в настоящее время, ведется разработка специальной графической надстройки для используемого языка описания предметной области, в которой определенные графические объекты соответствуют предложениям языка. Данная надстройка разрабатывается в целях облегчения описания пользователем моделей предметных областей и удобной постановки задач на синтез.

Заключение

Большинство проблем, связанных с разработкой параллельных программ, может решить разработка системы автоматического синтеза параллельных программ по непроцедурным спецификациям.

Проанализированы существующие подходы к синтезу параллельных программ, выявлен ряд их недостатков и пути их устранения. Разработаны и апробированы алгоритмы и программы, позволяющие получать эффективные параллельные программы по непроцедурным спецификациям.

СПИСОК ЛИТЕРАТУРЫ

1. Корнеев В.Д. Параллельное программирование в MPI. – Новосибирск: ИВМиМГ, 2002. – 215 с.
2. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2004. – 608 с.
3. Вальковский В.А., Малышкин В.Э. Синтез параллельных программ и систем на вычислительных моделях. – Новосибирск: Наука, 1988. – 128 с.
4. Андрианов А.Н., Бугеря А.Б., Ефимкин К.Н., Задыхайло И.Б. Норма. Описание языка. Рабочий стандарт. – Препринт ИПМ им. М.В. Келдыша РАН, № 120, 1995. – 50 с.
5. Востокин С.В. Графическая объектная модель параллельных процессов и ее применение в задачах численного моделирования. – Самара: Изд-во Самарского научного центра РАН, 2007. – 286 с.
6. Новосельцев В.Б. Теория структурных функциональных моделей // Сибирский математический журнал. – 2006. – Т. 47. – № 6. – С. 1342–1354.

Поступила 16.04.2008 г.

Ключевые слова:

Многопроцессорная вычислительная система, суперЭВМ, предложение вычислимости, параллелизм, линейный участок, синтез программ, граф алгоритма, ярусно-параллельная форма, параллельная программа, С-модель.

УДК 519.67(004.02)

АЛГОРИТМЫ ЛОКАЛИЗАЦИИ ТОЧКИ В ТРЕХМЕРНОМ ПРОСТРАНСТВЕ ДЛЯ ГЕНЕРАЦИИ ОБЪЕКТА ПРИ МОДЕЛИРОВАНИИ МЕТОДОМ ЧАСТИЦ

В.В. Сергеев, С.Ю. Коростелев, С.Г. Псахье

Институт физики прочности и материаловедения СО РАН, г. Томск
E-mail: svalera@ispms.tsc.ru

Для решения задачи построения трехмерных объектов сложной конфигурации и заполнения расчетной области моделируемыми частицами предложено два алгоритма. Представлены основные положения, достоинства и недостатки каждого из алгоритмов. На основе тестовых расчетов предложены способы увеличения эффективности решения данной задачи с помощью современных компьютерных технологий.

1. Введение

При компьютерном моделировании в рамках метода частиц [1] одной из топологических проблем является построение трехмерных объектов сложной конфигурации. Процесс генерации объекта для моделирования состоит из двух этапов. На первом этапе необходимо создать геометрическую модель объекта представленную одним или несколькими многогранниками. Для этого можно использовать любую коммерческую или свободно распространяемую программу, например, Autocad®. На втором этапе полученные многогранники заполняются моделируемыми частицами. Однако для этого не существует реализованных алгоритмов, поэтому приходится разрабатывать новые.

Предлагаемые в работе алгоритмы тестировались в программах для моделирования методом молекулярной динамики и методом подвижных клеточных автоматов [2, 3]. Основное применение рассматриваемые подходы могут получить в методе подвижных клеточных автоматов, поскольку этот метод позволяет моделировать сложные системы и конструкции.

В рамках этого метода моделируемая система представляет собой ансамбль взаимодействующих автоматов, имеющих конечный размер. Образец представляет собой трехмерное тело произвольной формы. В настоящее время самым удобным является способ описания 3D-объекта с помощью полигональной сетки. Это позволяет использовать все доступные на сегодняшний день трехмерные редакторы для создания геометрии образца любой сложности. Имея описанный сеткой образец произвольной геометрической формы, необходимо представить его в виде набора автоматов. Автомат может иметь различную форму в зависимости от упаковки (подробнее в разделе 2). Основной задачей является определение принадлежности автомата (представленного точкой) к образцу (представленному триангуляционной сеткой), т. е. локализация точки в 3D-пространстве.

При заполнении внутренней области многоугольника необходимо определить те точки, которые находятся внутри многоугольника, и те, которые находятся снаружи. Для этого проводится луч, выходящий из проверяемой точки и уходящий в бесконечность в любом направлении и не проходящий через вершины. Если луч пересекает ребра многоугольника нечетное число раз, то точка лежит во внутренней области. Такой подход получил название правило «нечетного равенства». Первый алгоритм, использовавший данную схему, описан в работах [4, 5]. Д.Т. Ли и Ф.П. Препарата предложили алгоритм локализации точки в трехмерном пространстве относительно выпуклого многогранника [6].