ях сложного сдвига [Текст] / С.А. Бостаджиян, В.И. Боярченко, Г.Н. Каргополова // Наука и техника. – Минск, 1970. – С. 111–121.
3. Балешов, И.М. Решение некоторых задач, связанных с течением расплавленных полимеров в червячных прессах [Текст] / И.М. Балешов, А.Н. Левин // Хим. Машиностроение. – М.: 1961, №6. – С. 29–33.
4. Басов, Н.И. Гидродинамика и теплообмен при плавлении в винтовом канале шнекового аппарата [Текст] / Н.И. Басов, И.Н. Володин, Ю.В. Казанков [и др.] // Наука. Теоретические основы химической технологии. – М.: 1983, т. 17, №1. – С. 72.
5. Первадчук В.П., Математическая модель плавления полимерных материалов в пластицирующих экструдерах. Влияние физических свойств полимера и режимов переработки на скорость плавления [Текст] / В.П. Первадчук, Н.М. Труфанова, В.И. Янков // Химические волокна. – М.: 1984, №6. – С. 46-48.
6. Торнер Р.В. Теоретические основы переработки полимеров (механика процессов) [Текст] / Р.В. Торнер/ – М.: «Химия», 1977. – 464 с.: ил.

*Аспирант **Нечаев А.С.** тел. 8-902-321-72-09, nechaev-as@mail.ru; д.т.н., проф. **В.Н. Митрошин** - vmitroshin@mail.ru; каф. автоматики и управления в технических системах Самарского государственного технического университета*

УДК 004.41

МОДЕЛИРОВАНИЕ И ОТЛАДКА ЗАВИСИМЫХ ПОТОКОВ В ПАРАЛЛЕЛЬНЫХ СИСТЕМАХ

А.В. Сикерин, Е.Н. Крючкова

В статье рассматривается технология разработки и тестирования параллельного программного обеспечения на основе моделей, обеспечивающих описание многопоточных взаимодействий в параллельных системах. Предложена модель параллельного автомата, исследованы его свойства, предложен подход его использования для тестирования параллельного программного обеспечения.

Ключевые слова: параллельные программирование, отладка многопоточных программ, теория автоматов, model checking, тестирование ПО.

Введение

Снижение стоимости параллельных вычислений, увеличение количества ядер, асинхронная природа пользовательских интерфейсов вынуждают разработчиков использовать многопоточность. Однако параллельные системы намного сложнее, чем последовательное программное обеспечение (ПО). Промышленное ПО часто подвержено изменениям, при каждом из них разработчик обязан обеспечивать требуемое поведение ПО. Возрастают размеры и сложность параллельных систем, что означает, увеличение стоимости производства параллельного ПО. Более того, порядок многопоточных взаимодействий в параллельной системе не детерминирован, и необходимо обеспечить корректное поведение параллельной системы для всех возможных исполнений.

Это приводит к острой необходимости создания новых принципов разработки инструментов для анализа, отладки и тестирования параллельных алгоритмов. Инструменты автоматического анализа позволяют разработчику непрерывно обеспечивать требуемые свойства, тестировать, поддерживать высокое качество ПО. Такие инструменты можно рассматривать как некоторое сито, через которое отсеиваются все версии ПО с неправильным поведением.

Проблемы параллельного программирования

Написание параллельного ПО согласно заданной спецификации является задачей хотя и трудной, но, в общем-то, выполнимой. Гораздо сложнее сохранить требуемые инварианты при модификациях. Широко распространёнными при разработке параллельного ПО являются следующие виды ошибок:

- гонки данных (data races);
- тупики (deadlocks);
- потоки в состоянии ожидания (stalled threads);
- потерянные сигналы (lost signals);
- заброшенные замки (abandoned locks).

Модель многопоточных взаимодействий в параллельных системах

На основе метаинформации, содержащейся в коде программы, построим математическую модель параллельного алгоритма как абстракцию вычислительного процесса. Модель строится в предположении, что код и модель – это одно целое. Рассмотрим поведение параллельного алгоритма как работу

расширенного иерархического конечного автомата с конечным количеством стеков.

Будем считать, что параллельная система выполняется на абстрактном вычислительном узле, удовлетворяющем вычислительной модели *PRAM* (параллельная машина с произвольным доступом к памяти) [1]. Большинство промышленных языков программирования являются объектными, поэтому будем считать, что параллельная система реализована на объектном языке.

Пусть параллельная система $PSys$ есть взаимодействие между элементами множества потоков $T=\{mThread,t_1,t_2,...\}$, где t_i - уникальный идентификатор потока, и элементами множества объектов $O=\{program,o_1,o_2,...\}$, где o_i - уникальный идентификатор объекта. Каждый объект $o\in O$ относится к некоторому классу c из множества $Cls=\{System,c_1,c_2,...\}$. Будем считать, что в системе всегда существуют единственный объект $program$ класса $System$ и главный поток $mThread$. Класс $c\in Cls$ определяет множество операций $Ops(c)$, которые может выполнять поток $t\in T$ над объектом $o\in O$ класса $c\in Cls$.

Разработчик размечает определенную семантически значимую точку кода для некоторого класса $c\in Cls$ операции $op\in Ops(c)$ метаинформацией о том, что данный фрагмент является состоянием модели $State_{op,c}$, где $State$ - уникальный идентификатор состояния для кортежа $\langle op,c\rangle$. Для пары $\langle op,c\rangle$ определим начальное и конечное состояния $s_{op,c}$ и $f_{op,c}$ соответственно. Обозначим s_{main} и f_{main} начальное и заключительное состояние объекта $program$ класса $System$. Построим автоматную модель, отражающую многопоточные взаимодействия на основе состояний и переходов между ними в соответствии с кодом приложения.

Автомат имеет три вида переходов, соответствующих поведению реальной параллельной системы:

- *call*-переход (синхронное взаимодействие) – последовательные вызовы функций, которые происходят в одном потоке. После завершения работы вызванной функции должен выполняться переход в точку возврата, которая находится в стеке программы;
- *return*-переход (возвратный переход) – соответствует событию возврата в некоторую точку, адрес которой находится в стеке программы;
- *fork*-переход (асинхронное взаимодействие) – операция создания нового потока. После выполнения операции программа продолжает выполняться последовательно по коду.

Для того чтобы модель выражала семантику взаимодействий реальных многопоточных программ, введем понятия синхронной и асинхронной *call*-вершины. *Call*-вершина – это примитив автоматной модели, который имеет две составляющие: множество исполняемых переходов и возвратный переход. Множество исполняемых переходов содержит только последовательные или асинхронные переходы для синхронной и асинхронной вершин соответственно. Автомат в *call*-вершине c, должен выполнить все исполняемые переходы в некотором порядке. После этого он должен перейти в состояние, указанное возвратным переходом вершины c. C*all*-вершины добавляются искусственно, на основании кода параллельной системы.

Граф состояний параллельной системы

Определим множества вершин автомата и свяжем их соотношениями:

$$
\begin{aligned}
&S=S^{start}\bigcup S^{call}\bigcup S^{mark}\bigcup S^{finish},\\
&S^{start}=\bigcup_{\substack{c\in Cls\\ op\in Ops(c)}} s_{op,c},\\
&S^{call}=S^{call,sync}\bigcup S^{call,async},\\
&S^{mark}=\bigcup_{\substack{c\in Cls\\ op\in Ops(c)}} S^{mark}_{op,c},\\
&F^{finish}=\bigcup_{\substack{c\in Cls\\ op\in Ops(c)}} f_{op,c},\\
&S^{call,sync}=\bigcup_{\substack{c\in Cls\\ op\in Ops(c)}} S^{call,sync}_{op,c},\\
&S^{call,async}=\bigcup_{\substack{c\in Cls\\ op\in Ops(c)}} S^{call,async}_{op,c},\\
&S^{call}_{op,c}=S^{call,sync}_{op,c}\bigcup S^{call,async}_{op,c},\\
&S^{all}_{op,c}=s_{op,c}\bigcup S^{mark}_{op,c}\bigcup S^{call}_{op,c}\bigcup f_{op,c}.
\end{aligned}
\qquad (1)
$$

Графом состояний параллельной системы $PSys$ назовем ориентированный граф $G=(S,E,\mathrm{T})$ с помеченными ребрами, где:

- S - множество вершин графа;
- $\mathrm{T}=\{sync, async, return\}$ - множество типов переходов;
- E - множество T - помеченных ребер.

Множество E имеет вид:

$$E=E^{sync}\bigcup E^{async}\bigcup E^{return},$$
$$E^{sync}\subseteq(S\setminus S^{call})\times S\times\{sync\},$$
$$E^{async}\subseteq S^{call,async}\times S^{start}\times\{async\},\quad(2)$$
$$E^{return}\subseteq S^{call}\times(S\setminus S^{start})\times\{return\}.$$

Будем изображать примитивы автоматной модели согласно рисунку 1. Как любая графовая структура, граф состояний параллельной системы отображает связи между объектами и характер взаимодействий между ними. Свяжем историю исполнения [2] параллельной системы с некоторым способом обхода вершин графа состояний. С этой целью преобразуем граф состояний к виду, удовлетворяющему классическому определению инициального автомата-распознавателя с конечным множеством стеков [3].

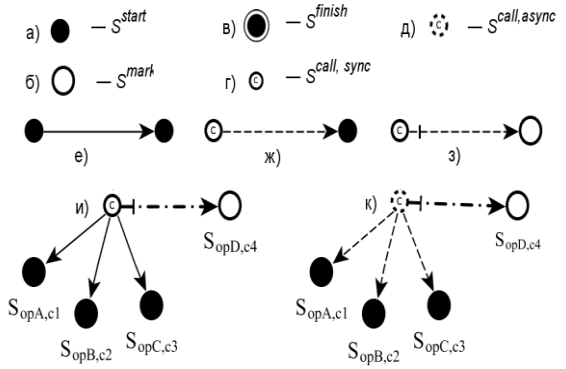

Рисунок 1 - Примитивы параллельного автомата

У результирующего автомата имеется только одна входная лента, конечное множество стеков CU, стеки RS и TS. Автомат проверяет допустимость истории исполнения параллельной система как цепочку входных данных, т.е. проверяет принадлежность некоторого слова m над заданным алфавитом Σ заданному языку L:

- каждому ребру (a,b), где $b\in(S\setminus S^{call})$, поставим в соответствие метку b;
- каждому ребру (a,b), где $b\in S^{call}$, поставим в соответствие метку ε.

Определение параллельного автомата

Параллельным автоматом A над n вычислительными узлами назовем набор объектов вида:

$$A=(Q,\Sigma,\Gamma,\delta,q_0,F=\{f_0\},\ n,CU,TS,RS).\quad(3)$$

Здесь Q – множество состояний автомата, включающее специальные состояния $\{start, run, finish, swap_i\,|\,1\le i\le n\}$; Σ – входной алфавит истории исполнения параллельной системы; Γ – алфавит стеков CU,TS,RS; δ – функция переходов; q_0 – начальное состояние; $F=\{f_0\}$ – множество заключительных состояний. Специальные состояния $start, run, finish$ соответствуют работе многопоточного приложения без переключения между потоками (запуск потока $mThread$, работа автомата без переключения между потоками, завершение потока $mThread$). Состояние $swap_i$ - соответствует переключению контекста на i - ом вычислительном устройстве. Структура параллельного автомата A представлена на рисунке 2.

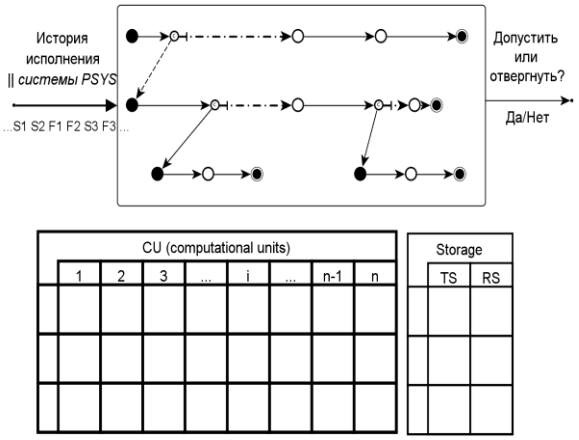

Рисунок 2 - Структура параллельного автомата

Операция смены контекста (*Switch context*) выполняется следующим образом:

- выбирается некоторое число i - номер вычислительного устройства;
- автомат переходит в состояние $swap_i$, в котором содержимое стека CU_i заменяется фрагментом содержимого стека TS (при этом в качестве рабочего стека используется стек RS).

Используя известные свойства автоматов с n стеками [3], попытаемся рассмотреть

проблему проверки правильности истории исполнения как проблему проверки пустоты пересечения распознаваемого автоматом языка и запроса, сформулированного в форме параллельного регулярного выражения.

Операция параллельного объединения на множествах слов

Пусть задан некоторый алфавит Σ. Словом будем называть последовательность символов данного алфавита. Обозначим через $l(a)$ длину слова a. Пусть x, y – слова над алфавитом Σ. Рассмотрим все перестановки π длины $l(x)+l(y)$, которые обладают следующим свойством:

$$\pi : \forall 1 \le i < l(x) : (\pi_i < \pi_{i+1}) \,\&\, \forall l(x)+1 \le j < l(x)+l(y) : (\pi_j < \pi_{j+1}). \quad (4)$$

Тогда для слов x и y введем бинарный оператор параллельного объединения $||$

$$x \,||\, y = \{c_i : \forall i = 1 \ldots C^{l(x)}_{l(x)+l(y)}, c_i = \pi(xy), \text{ для } \pi, \text{ указанного выше вида}\}. \quad (5)$$

Пусть A и B - некоторые множества слов над алфавитом Σ, тогда под параллельным объединением двух множеств будем понимать оператор следующего вида:

$$\forall A, B \subset \Sigma^* : A \ddot{\bigcup} B = \bigcup_{\substack{x \in A, \\ y \in B}} x \,||\, y. \quad (6)$$

Параллельные регулярные выражения и операции над ними

Введем понятие параллельного регулярного выражения, которое является расширением классического понятия регулярного выражения. Это позволит нам оперировать базовыми свойствами классических регулярных выражений, если они не содержат примитивов, характерных для параллельных регулярных выражений.

Пусть задан алфавит Σ. Будем обозначать через $L(E)$ язык, который порождает (соответствует) параллельному регулярному выражению E. Параллельным регулярным выражением называется выражение следующего вида:

- $\varepsilon, \varnothing$ - параллельное регулярное выражение;
- $a \in \Sigma$ - параллельное регулярное выражение;
- Если E - параллельное регулярное выражение, то E^*, E^+, (E) параллельные регулярные выражения (усеченная итерация, итерация, скобки соответственно);
- Если E и F - параллельные регулярные выражения, то EF или $E \square F$ - параллельные регулярные выражения (конкатенация или произведение);
- Если E и F - параллельные регулярные выражения, то $E + F$ - параллельное регулярное выражение (оператор синхронного объединения, или просто объединения);
- Если E и F - параллельные регулярные выражения, то $E \,||\, F$ - параллельное регулярное выражение (асинхронное объединение, $L(E \,||\, F) = L(E) \ddot{\bigcup} L(F)$).

Как и в классической алгебре регулярных выражений, данные операторы имеют определенные приоритеты вычислимости:

- скобки;
- оператор итерации
- оператор конкатенации
- оператор асинхронного объединения
- оператор синхронного объединения

Проверка историй исполнения

Пусть история исполнения задается параллельным регулярным выражением, которое дает возможность программисту формулировать инвариант, неизменный в течении всего цикла разработки ПО. Проверка пустоты пересечения языка, распознаваемого параллельным автоматом, и параллельного регулярного выражения позволяет проверить истинность инварианта (рисунок 3).

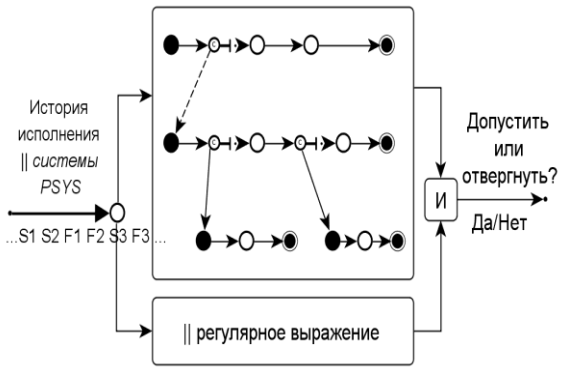

Рисунок 3 – Пересечение параллельного регулярного выражения и автомата

Вывод

Такой подход дает возможность программисту формулировать наборы ограничивающих условий и проверять их выполнимость на этапе разработки многопоточного приложения, а не на этапе тестирования и эксплуатации.

СПИСОК ЛИТЕРАТУРЫ

1. Миллер Р., Боксер Л. Последовательные и параллельные алгоритмы. [Текст] /Р.Миллер, Л. Боксер/ - М.: БИНОМ. Лаборатория знаний, 2006.
2. Line-up: a complete and automatic linearizability checker. [Текст] / S. Burckhardt, C. Dern, M. Musuvathi, R. Tan // Proceedings of the 2010 ACM SIGPLAN conference on Programming language design and implementation. — PLDI'10.— New York, NY, USA: ACM, 2010.— Pp. 330–340.
3. Atig, M. F. Emptiness of multi-pushdown automata is 2etime-complete [Текст] /M. F. Atig, B. Bollig, P. Habermehl // Proceedings of the 12th internationalconference on Developments in Language Theory.— DLT '08.— Berlin, Heidelberg: Springer-Verlag, 2008.— Pp. 121–133.

*Аспирант **Сикерин А.В.** тел. 8-913-238-69-48, andrey.sikerin@yandex.ru; к.ф.-м.н., проф. **Крючкова Е.Н.** – kruchkova_elena@mail.ru, (3852) 36-75-83; - каф. прикладной математики Алтайского технического университета им. И.И. Ползунова.*

УДК 338.001

МОДЕЛЬ ОЦЕНКИ И ПРОГНОЗИРОВАНИЯ ТРУДОЕМКОСТИ ТЕХНИЧЕСКОГО ОБСЛУЖИВАНИЯ ТРАНСПОРТНЫХ И ТЕХНОЛОГИЧЕСКИХ МАШИН

Ю.М. Осипов, С.Г. Пятов

Предлагается автоматизация процессов оценки и прогнозирования трудоемкости при техническом обслуживании и ремонте транспортных и технологических машин применением цифровой видеозаписи и компьютерных технологий, позволяющая получать больший объем данных для оптимизации рабочих процессов в автотранспортном предприятии. Использование цифровой видеозаписи дает возможность создания обучающих видеопрограмм для повышения квалификации персонала и качества его труда.

Ключевые слова: трудоемкость технического обслуживания, оптимизация рабочих процессов, себестоимость затрат, автоматизация хронометража.

Введение

Значительную долю в себестоимости работ транспортных и технологических машин (ТТМ) составляют затраты на техническое обслуживание и ремонт (ТОР). Определение оптимальных норм времени для ТОР может привести к снижению себестоимости затрат. В настоящее время для контроля норм времени используется ручной хронометраж.

На предприятиях автомобильного транспорта так же, как и в других отраслях народного хозяйства, важнейшим фактором роста производительности труда является техническое перевооружение его и более полное использование имеющейся техники. Повышение производительности труда на автотранспортных предприятиях непосредственно связано с выпуском автомобильной промышленностью новых типов автомобилей, характеризующихся повышенной грузоподъемностью, надежностью, прочностью, более высокими динамическими и экономическими качествами. Наряду с оснащением автомобильного транспорта технически совершенным парком подвижного состава и прежде всего специализированными автомобилями, к важнейшей группе факторов роста производительности труда на автомобильном транспорте – повышению его технического уровня относятся:

- внедрение новых видов гаражного оборудования;
- совершенствование технологии ТОР автомобилей.

Предлагается применение цифровой видеозаписи и компьютерных технологий при ТОР ТТМ, которая позволяет получать бóльший объем данных для оптимизации процессов ТОР и своевременно корректировать нормы времени для любого автотранспортного предприятия и различных марок подвижного состава. Кроме того, использование цифровой видеозаписи при ТОР дает возможность создания обучающих видеопрограмм для повышения квалификации персонала и качества его труда, а также юридического сопровождения технологии ТОР и соблюдения определенных норм времени.

Алгоритм технического обслуживания и ремонта подвижного состава на предприятиях автомобильного транспорта.

Техническое обслуживание подвижного состава по периодичности, перечню и трудоемкости выполняемых работ подразделяется на: