Савенков К.О.

ИСПОЛЬЗОВАНИЕ ЗАВИСИМОСТЕЙ ПРИ МАСШТАБИРОВАНИИ ИМИТАЦИОННЫХ МОДЕЛЕЙ[1]

Лаборатория Вычислительных Комплексов ВМиК МГУ,
Москва,
savenkov@cs.msu.su

Аннотация

В данной статье мы показываем, как понятие зависимостей, используемое при анализе последовательных программ, можно адаптировать для поддержки масштабирования имитационных моделей – приведения описания имитационной модели к определённому уровню абстракции с сохранением заданных свойств.

Помимо известных зависимостей по данным и управлению, позволяющих сохранить при преобразованиях поток данных и управления модели, мы также вводим зависимости по времени выполнения и зависимость по исполнителю, позволяющие сохранить временные свойства имитационной модели.

В работе рассматриваются дискретно-событийные имитационные модели, построенные для среды DYANA[1].

Введение

При работе с моделями сложных нелинейных систем часто возникает необходимость их анализа и модификации. Для её решения требуется подобрать такое представление системы, которое бы позволило в задачах анализа глубже понять её, а при модификации – сохранить проверяемые свойства исходной модели. Одна из таких задач – масштабирование имитационных моделей [7], при котором требуется абстрагироваться от излишне детального поведения модели, не влияющего на выполнимость её проверяемых свойств.

Под имитационной моделью мы понимаем совокупность параллельно выполняющихся последовательных процессов, взаимодействующих при помощи механизма передачи сообщений [1]. У каждого процесса есть набор входных и выходных буферов. Сообщение, посланное одним процессом другому при помощи оператора передачи сообщения, помещается во входной буфер последнего, и может быть считано оттуда оператором приёма сообщения [1].

Также в состав модели входит набор исполнителей, определяющих скорость выполнения процессов, привязанных к ним. Каждый процесс обладает своей временной шкалой (модельным временем). При выполнении процессом того или иного оператора генерируется событие, вклю-

[1] Работа выполнена при частичной поддержке РФФИ, грант № 04-01-00556

чающее в себя описание выполненного оператора и его операндов, а также временную метку. Эти события фиксируются наблюдателем при выполнении модели с конкретными входными данными и заносятся в трассу имитационного эксперимента. Далее собранная трасса используется для проверки требуемых свойств модели.

Масштабирование имитационных моделей

В случае, если для проверки свойств достаточно наблюдать поведение имитационной модели на высоком уровне абстракции (например, достаточно информации о выполнении операций посылки сообщений), а в трассу заносится более детальная информация (например, содержание пересылаемых сообщений), может оказаться полезным повысить уровень абстракции описания имитационной модели так, чтобы при выполнении полученной модели машинное время не расходовалось на моделирование поведения, не существенного для выполнения проверяемых свойств. Эта задача решается при помощи масштабирования имитационной модели [7].

В имитационном моделировании все проверяемые свойства исходной системы формулируются в терминах наблюдаемых событий имитационной модели. При масштабировании имитационной модели нам дан набор событий, которые должны попасть в поле зрения наблюдателя при проведении имитационного эксперимента. Множество наблюдаемых событий может быть описано как совокупность операторов, информация о выполнении которых нас интересует, и набора операндов, информация о значении которых также должна включаться в событие.

Необходимо, с одной стороны, убрать из описания модели как можно больше операторов, не попавших в поле зрения наблюдателя, а с другой стороны – сохранить поведение модели неизменным с точки зрения наблюдателя. При этом необходимо позаботиться о сохранении временных свойств модели, поэтому часть таких операторов мы можем просто удалить, другую часть необходимо заменить на операторы продвижения модельного времени, остальные нужно оставить на своём месте.

Механизм зависимостей

Таким образом, нам требуется аппарат, ограничивающий нас в удалении ненаблюдаемых операторов из описания модели. Этот аппарат должен помочь нам определить, какие операторы могут быть удалены из описания модели, а какие – влияют на события, генерируемые при выполнении наблюдаемых операторов, и должны быть оставлены на своём месте.

В качестве подобного аппарата мы предлагаем использовать механизм зависимостей, аналогично тому, как зависимости используются при сечении последовательных программ [5]. Все зависимости мы раз-

деляем на два основных типа: зависимости между операторами одного последовательного процесса (внутрипроцессные), и зависимости, связывающие операторы различных процессов (межпроцессные).

Тогда масштабирование имитационной модели можно провести по аналогии с сечением последовательных программ [5], используя межпроцессные зависимости для транзитивного распространения внутрипроцессных зависимостей за рамки одного последовательного процесса. В этом случае масштабирование имитационной модели может быть выполнено по следующей схеме [7]:

1. Построение множества внутри- и межпроцессных зависимостей между операторами имитационной модели.

2. Построение на основе проверяемых свойств «окна наблюдателя» − множества наблюдаемых операторов модели.

3. Построение остаточных моделей последовательных процессов при помощи транзитивного замыкания множества внутрипроцессных зависимостей по множеству операторов из «окна наблюдателя».

4. Расширение «окна наблюдателя» на основе множества межпроцессных зависимостей.

5. Выполнение шагов 3-4, пока остаточная модель не сойдётся.

6. Удаление или замена на операторы продвижения модельного времени операторов, не попавших в остаточную модель.

Для последовательных процессов имитационной модели мы будем использовать известные зависимости по данным и по управлению [3,4], добавив к ним зависимость по времени выполнения. Для того, чтобы учесть зависимости, возникающие между процессами при их взаимодействии, мы распространим действие зависимостей по данным, управлению и времени выполнения за рамки одного процесса, добавив к межпроцессным зависимостям по данным и управлению зависимости по синхронизации и по исполнителю.

Внутрипроцессные зависимости

Определения основных зависимостей в последовательных программах заимствованы нами не из классических статей Ферранте и Харрольда [3,4], а из статьи Подгурски и Кларка [6], где они изложены в форме, подходящей не только для последовательных программ, но и для реактивных систем.

Отношение зависимости между операторами последовательного процесса может быть выявлено по тексту описания процесса и используется для предсказания возможного хода его выполнения. Существует два основных типа зависимостей в последовательной программе: зависимости по управлению, которые определяются управляющими конструкциями программы, и зависимости по данным, которые определяются переменными, используемыми в программе.

1. input (X,Y); 2. if X > Y then

```
3.  Max := X;
    else
4.  Max := Y;
    endif;
5. output (Max);
```

Рис 1. Фрагмент процесса, вычисляющего значение максимума.

Оператор s *зависит по управлению* от предиката c, который содержится в операторе условного ветвления, если, согласно структуре потока управления программы, от выбора пути выполнения, на который потенциально влияет c, зависит, будет ли выполнен оператор s. Например, в программе на рис. 1 оператор 3 зависит по управлению от условия ветвления в строке 2.

Оператор s *зависит по данным* от оператора s', если данные, определяемые в s', и используетмые в s, потенциально могут достичь s через последовательность присваиваний переменных. Например, в программе на рис. 1 оператор 5 зависит по данным от опратора 1, поскольку значение Max в операторе 5 потенциально зависит от значений X и Y, определяемых в операторе 1.

Оператор s *синтаксически зависит* от оператора s', если s транзитивно зависит от s' через цепочку зависимостей по данным и управлению. Оператор s *семантически зависит* от оператора s’, если существует такая интрепретация модели, что функция, вычисляемая в s, зависит от функции, вычисляемой в s’.

Потенциально, синтаксическая зависимость одного оператора программы от другого может также означать семантическую зависимость [6]. Можно сформулировать утверждение, что s семантически зависит от s’, если s синтаксически зависит от s’, и s’ достижим из s.

Оператор s зависит от оператора s’ *по времени выполнения*, если оператор s’ может быть выполнен до оператора s’, и, тем самым, временная метка выполнения оператора s будет зависеть от временной метки выполнения оператора s’. Так, в приведённом примере оператор 5 зависит по времени выполнения от оператора 4.

Использование внутрипроцессных зависимостей

Итак, мы рассматриваем зависимости как фактор, ограничивающий возможность удаления операторов из модели. Если событие, отмечающие факт выполнения оператора s, входит в наблюдаемое поведение модели, то множество операторов S_c, от которых s зависит по управлению, также войдут в остаточную модель. Также туда войдут все операторы, от которых операторы из множества S_c зависят синтаксически.

Аналогичным образом, если значения некоторых операндов оператора s отражаются в наблюдаемом поведении модели, то множество операторов S_d, от которых зависят значения данных операндов в точке выполнения s, также войдут в остаточную модель вместе со всеми операторами, зависящими от них синтаксически.

Зависимость по времени выполнения используется на этапе замены части описания модели, не попавшей в остаточную модель, на операторы продвижения модельного времени. Мы можем объединить в одном операторе продвижения модельного времени задержки для ряда операторов, которые в исходной модели связаны отношением непосредственного доминирования [6]. Те операторы, от которых зависят по управлению операторы, заменяемые на задержки модельного времени, должны быть добавлены в остаточную модель.

Межпроцессные зависимости

Существующие работы, посвящённые зависимостям в распределённых программах, рассматривают программы над общей памятью, лишь вскользь упоминая зависимости, возникающие при передаче сообщений [2].

Для того, чтобы учесть влияние взаимодействующих процессов друг на друга, мы вводим ряд зависимостей между операторами посылки и приёма сообщений различных процессов. Это зависимости по данным и управлению, определяемых очевидным образом, а также *зависимость по синхронизации*, которая связывает два оператора различных процессов, один из которых должен дождаться выполнения другого. Данный вид зависимости используется для того, чтобы распространить зависимость по времени выполнения на процессы, с которыми взаимодействует данный процесс.

Коснёмся ещё одного вида зависимостей – *зависимости по исполнителю*. Если два последовательных процесса привязаны к одному исполнителю, то это накладывает ограничения на частичный порядок выполнения их действий. Их параллельное исполнение будет организовано по принципу чередования, что приведёт к дополнительным зависимостям по времени выполнения между действиями данных процессов.

Заключение

В данной работе мы адаптировали известные зависимости, используемые для анализа последовательных программ, для анализа дискретно-событийных имитационных моделей, разработанных для среды DYANA. Также мы предложили две новых зависимости, по времени выполнения и исполнителю, которые позволяют анализировать временные свойства модели.

Описанные подобным образом зависимости можно использовать и для масштабирования систем взаимодействующих процессов, описанных в других парадигмах имитационного моделирования.

Подгурски и Кларк [6] доказали, что синтаксическая зависимость между двумя операторами является необходимым, не достаточным условием существования между ними семантической зависимости. Этот результат означает, что при масштабировании имитационной модели ряд

операторов, оставленных в ней согласно построенному множеству синтаксических зависимостей, могут оказаться несущественными для выполнения проверяемых свойств модели.

Наиболее перспективным способом уточнения синтаксической зависимости нам представляется использование анализа выполнимости путей в имитационной модели, в частности, статический анализ потока данных между процессами.

1. A. Bakhmurov, A. Kapitonova, R. Smeliansky, DYANA: An Environment for Embedded System Design and Analysis, *Proc. of 32-nd Annual Simulation Symposium*, San Diego, California, USA, April 11-15, 1999.
2. Jingde Cheng: Dependence Analysis of Parallel and Distributed Programs and Its Applications. *APDC 1997*: 370-377
3. Jeanne Ferrante , Karl J. Ottenstein , Joe D. Warren, The program Dependence Graph and its Use in Optimization, *Proceedings of the 6th Colloquium on International Symposium on Programming*, p.125-132, April 17-19, 1984
4. Harrold, M. J., Malloy, B., and Rothermel, G. 1993. Efficient construction of program dependence graphs. In *Proceedings of the 1993 ACM SIGSOFT ISSTA* '93. ACM Press, New York, NY, 160-170.
5. Horwitz, S., Reps, T., and Binkley, D. 1990. Interprocedural slicing using dependence graphs. *ACM Trans. Program. Lang. Syst.* 12, 1 (Jan. 1990), 26-60.
6. Podgurski, A. and Clarke, L. A. 1990. A Formal Model of Program Dependences and its Implications for Software Testing, Debugging, and Maintenance. *IEEE Trans. Softw. Eng.* 16, 9 (Sep. 1990), 965-979
7. Савенков К.О., Смелянский Р.Л., Автоматическое масштабирование дискретно-событийных имитационных моделей, готовится к публикации.