Автоматический синтез минимальных конечно-автоматных моделей функциональных блоков по примерам поведения и темпоральным свойствам[1]

Чухарев К.И., магистрант 1 курса факультета информационных технологий и программирования Университета ИТМО, kchukharev@itmo.ru

Аннотация

Конечные автоматы широко используются в программной инженерии, особенно при разработке систем управления. Поведение контроллера может быть представлено с помощью конечно-автоматной модели, которая позволяет описать реакции системы на входные воздействия и выработку выходных воздействий. Конечно-автоматные модели используются при тестировании и верификации разрабатываемых систем, а также при их обратной разработке [1]. Обычно такие системы разрабатываются вручную, однако поддержание их в актуальном состоянии при изменении системы требует дополнительных усилий. Для того чтобы облегчить процесс поддержки, используются автоматизированные подходы, позволяющие синтезировать модели по примерам поведения и/или темпоральным свойствам [2–4]. Слабой стороной такого подхода является высокая вычислительная сложность задач синтеза минимальных моделей по заданным примерам поведения [5]. Сильной же стороной является гарантия корректности моделей, удовлетворяющих темпоральным свойствам, что подтверждается их формальной верификацией.

В данной работе рассматривается синтез минимальных конечно-автоматных моделей функциональных блоков, служащих базовыми блоками при представлении систем управления в соответствии с международным стандартом распределенных систем управления и автоматизации IEC 61499.

Введение

В международном стандарте систем управления и автоматизации IEC 61499 системы управления описываются функциональными блоками. Алгоритм управления внутри функционального блока представляется в виде расширенного конечного автомата Мура, в котором состояния ассоциированы с выходными событиями и алгоритмами, изменяющими значения выходных переменных, а

1 Работа поддержана грантом Правительства Российской Федерации 08-08.

переходы ассоциированы с входными событиями и охранными условиями – булевыми функциями от входных переменных. Задача синтеза конечных автоматов является фундаментальной задачей информатики и в данной работе рассматривается задача синтеза конечно-автоматных моделей функциональных блоков. В качестве модельной системы в данной работе была рассмотрена система Pick-and-Place манипулятора, состоящая из объекта управления и контроллера, реализующего алгоритм управления – отметим, что рассматривается именно задача синтеза модели контроллера.

Для синтеза в данной работе используются сценарии выполнения (в дальнейшем называемые позитивными, так как они описывают желаемое поведение конечно-автоматной модели) и LTL (*Linear Temporal Logic* – линейная темпоральная логика) спецификация [6]. Сценарии выполнения собираются, например, путем симуляции существующего функционального блока и представляют собой последовательность входных действий, подаваемых на вход системы, и выходных действий – соответствующих реакций системы. LTL спецификация – это набор свойств линейной темпоральной логики. Существует два основных типа LTL свойств: свойства безопасности (*safety*, «с системой *никогда* не происходит ничего *плохого*») и свойства живости (*liveness*, «что-то *хорошее* происходит *бесконечно часто*»). LTL спецификация обычно составляется вручную на основе знаний о внутреннем устройстве системы. Примером свойства безопасности в модельной системе Pick-and-Place манипулятора может быть запрет одновременного сжатия и расширения цилиндра манипулятора (LTL формула: $\mathbf{G}\neg(\text{c1Extend} \wedge \text{c1Retract})$), а примером свойства живости – непосредственная задача манипулятора – перемещение рабочих деталей ($\mathbf{G}(\text{lifted} \Rightarrow \mathbf{F}(\text{dropped}))$).

Отметим также задачу синтеза минимальных моделей – компактность моделей не только значительно упрощает ручной анализ моделей человеком, но и играет решающую роль во встраиваемых системах. Также стоит отметить, что минимизация моделей обеспечивает их *обобщение* – способность корректно реагировать на события, отсутствующие в «обучающем» наборе примеров поведения. Обычно обобщение достигается путём минимизации числа состояний автомата, но в данной работе мы дополнительно принимаем во внимание сложность охранных условий – суммарный размер соответствующих булевых формул.

Итак, задача формулируется следующим образом – найти минимальный конечный автомат, удовлетворяющий заданным позитивным сценариям и LTL спецификации. Предлагаемый метод отличается от существующих тем, что одновременно позволяет учесть и позитивные сценарии выполнения, и LTL свойства, а также синтезировать минимальные модели – как в терминах числа состояний, так и в терминах суммарной сложности охранных условий.

Предлагаемый подход

Для того чтобы проверить модель на соответствие LTL спецификации, используются специальные программные средства – верификаторы (*model checker*), например, NuSMV [7]. Верификатор находит контрпример для каждого LTL свойства, не выполняющегося для модели. Например, контрпример для свойства живости представляет собой бесконечную, но периодическую последовательность состояний автомата, нахождение в которых нарушает свойство живости. Контрпримеры преобразуются в негативные сценарии, описывающие нежелательное поведение конечно-автоматной модели. Таким образом, формулировка задачи уточняется до «нахождения минимального конечного автомата, удовлетворяющего заданным позитивным сценариям и неудовлетворяющего негативным».

Для непосредственного синтеза конечно-автоматных моделей в данной работе применяется метод программирования в ограничениях – исходная задача сводится к задаче о выполнимости булевой формулы (*Boolean satisfiability problem* – SAT) [8], для решения которой применяются современные высокопроизводительные программные средства – SAT-решатели. Процесс сведения заключается в кодировании структуры искомого автомата и факта соответствия/несоответствия позитивным/негативным сценариям в виде булевой формулы в конъюнктивной нормальной форме (КНФ). SAT-решатель либо находит удовлетворяющую подстановку значений булевых переменных в формуле (случай SAT), на основе которых можно восстановить искомый автомат, либо гарантирует отсутствие решения (случай UNSAT). Схема описанного подхода изображена на Рис. 1.

Для синтеза минимальных моделей применяется итеративный подход – параметры искомого автомата перебираются до тех пор, пока

автомат не будет найден (случай SAT на Рис. 1). Например, число состояний C искомого автомата сначала принимается равным 1, а затем последовательно увеличивается на 1, пока решение не будет найдено.

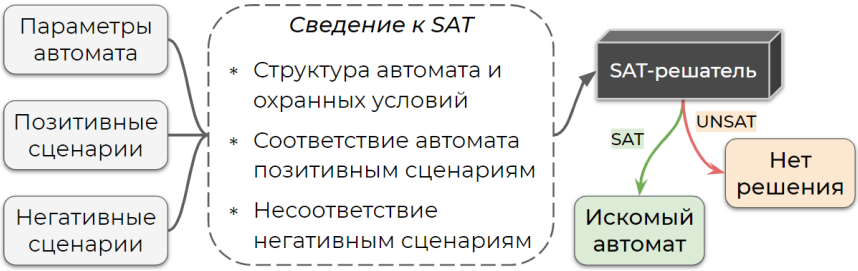

Рисунок 1: Схема общего подхода к синтезу конечно-автоматных моделей с применением программирования в ограничениях

Описанный выше подход позволяет строить минимальные модели, удовлетворяющие заданным позитивным сценариям и не удовлетворяющие некоторым негативным сценариям. Для построения модели, удовлетворяющей заданной LTL спецификации, применяется подход CEGIS (*Counterexample-Guided Inductive Synthesis*), заключающийся в повторении цикла «синтез – верификация» до тех пор, пока model checker не подтвердит отсутствие контрпримеров к заданным LTL свойствам у последней синтезированной конечно-автоматной модели. Описанный подход CEGIS изображен на Рисунке 2.

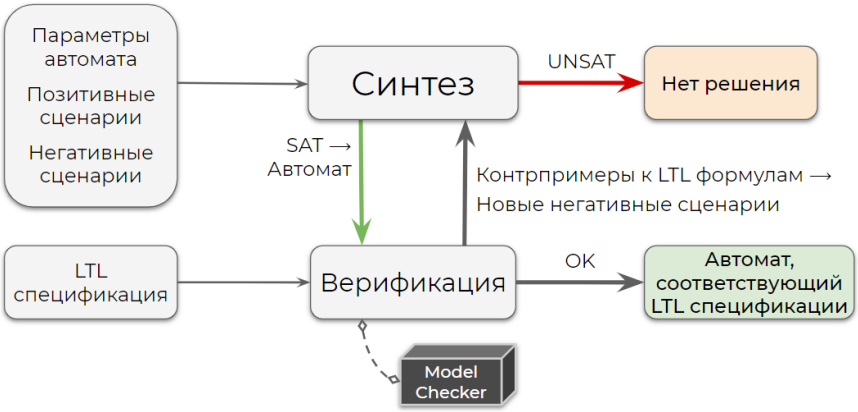

Рисунок 2 – Подход CEGIS (Counterexample-Guided Inductive Synthesis) для учета LTL спецификации при синтезе конечно-автоматных моделей

Заключение

В данной работе был предложен метод синтеза минимальных конечно-автоматных моделей функциональных блоков по примерам поведения и LTL спецификации. Предложенный метод был реализован в виде программного средства fbSAT, написанного на языке Kotlin и доступного онлайн [9]. Отличительной особенностью разработанного метода является минимизация охранных условий синтезируемого автомата, что позволяет находить *обобщенные* модели.

Было проведено экспериментальное исследование, заключающееся в синтезе модели контроллера Pick-and-Place манипулятора, подтвердившее эффективность разработанного метода по сравнению с существующими методами. При синтезе только по примерам поведения было проведено сравнение с двухэтапным методом *two-stage* из [10], а при синтезе с учетом LTL свойств – с методом fbCSP+LTL [11].

Дальнейшее исследование может включать в себя синтез модульных автоматов, применение более эффективных способов кодирования целочисленных переменных и ограничений мощности (*cardinality constraints* [12]) в SAT, а также исследование применимости быстрых эвристических методов минимизации охранных условий, например, Espresso [13].

Литература

1. B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke. A systematic survey of program comprehension through dynamic analysis // IEEE Trans. Softw. Eng., vol. 35, no. 5, pp. 684–702, 2009.
2. J.E. Cook and A.L. Wolf. Discovering models of software processes from event-based data // ACM Trans. Softw. Eng. Methodol., vol. 7, no. 3, pp. 215–249, 1998.
3. N. Walkinshaw, R. Taylor, and J. Derrick. Inferring extended finite state machine models from software executions // Empirical Softw. Eng., vol. 21, no. 3, pp. 811–853, 2016.
4. V. Ulyantsev, I. Buzhinsky, and A. Shalyto. Exact finite-state machine identification from scenarios and temporal properties // Int. Journ. Softw. Tools Techn. Transf., pp. 1–21, 2016.
5. E.M. Gold. Complexity of automaton identification from given data // Information and Control, vol. 37, no. 3, pp. 302–320, 1978.
6. Manna, Z., Pnueli, A.: Temporal verification of reactive systems, 1995.
7. Cimatti, A., Clarke, E., Giunchiglia, F., Roveri, M.: NuSMV: a new symbolic model checker // International Journal on Software Tools for Technology Transfer 2(4), pp. 410–425, 2000.
8. Biere, A., Heule, M., van Maaren, H., Walsh, T.: Handbook of satisfiability // Frontiers in artificial intelligence and applications, vol. 185, p. 980, 2009.
9. ctlab/fbSAT [Электронный ресурс]. URL: https://github.com/ctlab/fbSAT (дата обращения: 20.04.2019).
10. Chivilikhin, D., Ulyantsev, V., Shalyto, A., Vyatkin, V.: Function block finite-state model identification using SAT and CSP solvers // IEEE Transactions on Industrial Informatics pp. 1–1, 2019.
11. Chivilikhin, D., Buzhinsky, I., Ulyantsev, V., Stankevich, A., Shalyto, A., Vyatkin, V.: Counterexample-guided inference of controller logic from execution traces and temporal formulas // 23rd IEEE International Conference on Emerging Technologies and Factory Automation. pp. 91–98, 2018.
12. Petke, J. and Jeavons, P.: The order encoding: from tractable CSP to tractable SAT // SAT'11 Proceedings of the 14th international conference on Theory and application of satisfiability testing, pp. 371–372, 2011.
13. Brayton, R.K., Sangiovanni-Vincentelli, A.L., McMullen, C.T., Hachtel, G.D.: Logic Minimization Algorithms for VLSI Synthesis // Kluwer Academic Publishers, Norwell, MA, USA, 1984.