УДК 004.4’242

Построение конечно-автоматных моделей функциональных блоков по примерам поведения и темпоральным свойствам

Чухарев К.И., Университет ИТМО

Научный руководитель – Чивилихин Д.С., кандидат технических наук, научный сотрудник факультета информационных технологий и программирования Университета ИТМО

Введение и постановка задачи

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, как система реагирует на входные воздействия, и какие выходные события она вырабатывает. 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 темпоральные свойства [2–4]. Слабой стороной такого подхода является высокая вычислительная сложность задач синтеза минимальных моделей по заданным примерам поведения [5]. Сильной же стороной является высокая гарантия корректности моделей, удовлетворяющих темпоральным свойствам, что подтверждается формальной верификацией.

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

Целью настоящей работы является разработка метода синтеза минимальных конечно-автоматных моделей функциональных блоков по наборам примеров поведения и линейных темпоральных свойств.

Описание предлагаемого подхода

Предлагаемые методы основаны на сведении задачи построения конечно-автоматной модели к задаче выполнимости булевой формулы (Boolean Satisfiability – SAT) и использовании современных программных средств – SAT-решателей.

Первый из предлагаемых методов, *базовый*, позволяет синтезировать упрощенную конечно-автоматную модель минимального размера, достаточного для удовлетворения примеров поведения. Второй метод, *расширенный*, дополняет модель явно заданными охранными условиями в виде булевых функций, что позволяет минимизировать их суммарный размер, а также запрещает контрпримеры к заданным темпоральным свойствам, полученные в ходе формальной верификации итеративно синтезируемого автомата с помощью NuSMV. Таким образом, после цикла «синтез-верификация-запрет контрпримеров-минимизация», полученный конечный автомат удовлетворяет заданным примерам поведения и темпоральным свойствам, а также является минимальным в терминах выбранной метрики суммарной сложности автомата, что обеспечивает его обобщение, то есть способность корректно работать даже на тех входных данных, которые не были использованы в качестве примеров поведения при его построении.

Результаты

Были реализованы два предлагаемых метода – *базовый* и *расширенный*, а также реализован итеративный процесс «синтез-верификация-запрет контрпримеров». В качестве инструмента формальной верификации был использован символьный верификатор NuSMV[6], а в качестве инкрементального [7] SAT-решателя – cryptominisat [8].

Разработанные методы реализованы на языке Kotlin в виде программного средства fbSAT и доступны онлайн [9].

Список использованных источников

[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] NuSMV 2: An OpenSource Tool for Symbolic Model Checking A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani and A. Tacchella. In Proceeding of International Conference on Computer-Aided Verification (CAV 2002). Copenhagen, Denmark, July 27-31, 2002.

[7] lipen/incremental-cryptominisat [Электронный ресурс]. URL: https://hub.docker.com/r/lipen/incremental-cryptominisat (дата обращения: 20.02.2019).

[8] Soos M., Nohl K., Castelluccia C. Extending SAT Solvers to Cryptographic Problems // In Theory and Applications of Satisfability Testing, SAT 2009. Lecture Notes in Computer Science, vol 5584. Springer, Berlin, 2009.

[9] Lipen/fbSAT [Электронный ресурс]. URL: https://github.com/Lipen/fbSAT (дата обращения: 20.02.2019).