Generowanie automatów czasowych dla systemów czasu rzeczywistego

Agata Janowska

Autoreferat rozprawy doktorskiej

Systemy czasu rzeczywistego, nazywane też systemami czasowymi, to systemy, których działanie jest uzależnione od upływu czasu. Weryfikacja systemów czasowych jest ważnym zagadnieniem współczesnej informatyki ze względu na to, że coraz więcej takich systemów ma wpływ na nasze bezpieczeństwo — na przykład systemy sterowania ruchem drogowymi i kolejowym, czy oprogramowanie sprzętu medycznego.

Modelowanie systemów współbieżnych, w tym także czasowych, opiera się głównie na trzech podejściach: algebrze procesów, komunikujących się automatach i sieciach Petriego. Języki wyższego rzędu opisujące systemy czasowe bazują głównie na powyższych modelach, co umożliwia odpowiednie adaptowanie metod weryfikacji. Duża część badań w tej dziedzinie dotyczy opracowania i rozwijania metod weryfikacji dla modeli systemów współbieżnych z czasem takich jak automaty czasowe [AD90] i sieci Petriego z czasem [MF76]. Intensywny rozwój przeżywają zwłaszcza metody weryfikacji modelowej automatów czasowych, czego dowodem jest powstanie i stałe udoskonalanie narzędzi takich jak Kronos [DOTY95], UppAal [PL00] i VerICS [DJJ$^+$03], rozwijany w Instytucie Podstaw Informatyki PAN.

Z drugiej strony, komercyjne systemy czasu rzeczywistego są projektowane głównie w językach wyższego poziomu, wspomaganych przez narzędzia ułatwiające ich testowanie, symulację, czy generowanie kodu wykonywalnego, ale rzadko wspierające ich formalną weryfikację. Przykładami języków wysokiego poziomu, które pozwalają na wyrażanie zależności czasowych są Estelle [ISO97] i SDL [MT01], języki opisu protokołów komunikacyjnych i systemów rozproszonych. Również popularne języki programowania takie jak Java, czy UML są obecnie wzbogacane o elemety umożliwiające definiowanie warunków czasowych [BG00, DJPV02].

Aby móc zastosować metody i narzędzia weryfikacji modelowej automatów czasowych do sprawdzania poprawności systemów czasowych, opisanych w językach wysokiego poziomu, autorka rozprawy proponuje dokonać tłumaczenia z tych języków do automatów czasowych.

Istotnym problemem w praktycznym wykorzystaniu metod weryfikacji modelowej jest wykładnicza eksplozja liczby stanów modelu. Dlatego, podczas generowania automatów czasowych wykorzystane są metody redukcji modeli.

Cele rozprawy

Podstawowym zamierzeniem rozprawy jest dostarczenie metod i narzędzi generowania automatów czasowych na podstawie opisu systemów w językach wysokiego poziomu. W tym celu wprowadzamy reprezentację pośrednią, tak zwany język bazowy. Generowanie automatów odbywa się w dwóch krokach. Najpierw opis systemu w danym języku jest tłumaczony do reprezentacji pośredniej, a z niej do automatów czasowych. Na pierwszy krok powinny się składać przekształcenia wyłącznie syntaktyczne. Dlatego język bazowy musi być wystarczająco bogaty, aby umożliwić wyrażanie wszystkich istotnych aspektów komunikacji i synchronizacji systemów współbieżnych z ograniczeniami czasowymi.

Dla systemu opisanego w języku bazowym opracowano metody konstruowania automatów czasowych, a także metodę redukcji, dzięki której budowane automaty mają mniejsze rozmiary (w sensie liczby zegarów, stanów i tranzycji). Język bazowy i obie metody zostaną krótko omówione w kolejnych punktach, które odpowiadają głównym rozdziałom rozprawy.

Język bazowy

System czasu rzeczywistego opisywane są w języku bazowym jako zbiór procesów, zmiennych całkowitych i buforów komunikacyjnych. Każdy proces jest przedstawiany w sposób automatowo-zorientowany. Procesy wykonują pewne akcje (synchroniczne) wspólnie, a inne (lokalne) asynchronicznie. Język dostarcza także mechanizmów do komunikacji asynchronicznej — poprzez wymianę wiadomości za pomocą buforów. W języku nie występują jawnie zmienne reprezentujące czas, jednak z każdą akcją może być związane tak zwane dozwolone opóźnienie, które określa czas w jakim akcja może być wykonana.

W języku można również definiować zmienne zdaniowe, które są używane do budowy formuł logicznych różnych logik temporalnych, opisujących własności systemu typowe dla systemów współbieżnych, czyli własności bezpieczeństwa i żywotności. Dla przykładu można zdefiniować zmienną, która ma wartość *prawda* zawsze wtedy, gdy proces znajduje się w określonym stanie lub kiedy wartość zmiennej systemu spełnia określony warunek.

W rozprawie przedstawiono składnię i semantykę operacyjną języka bazowego, a także kilka przykładów zastosowania języka do opisu systemów z czasem.

Generowanie automatów czasowych dla języka bazowego

Automaty czasowe zostały zdefiniowane w pracy [AD90] jako skończenie stanowe automaty Büchiego rozszerzone o zmienne rzeczywiste reprezentujące zegary. Stan w automacie czasowym jest zwyczajowo nazywany *lokacją*, aby odróżnić go od stanu w jakim może znajdować się automat, na który oprócz lokacji składa się także wartościowanie zegarów. Najczęściej używana definicja automatów (także w tej

rozprawie) pochodzi z pracy [HNSY94], gdzie zamiast warunków akceptujących Büchiego zostały wprowadzone niezmienniki lokacji.

Sprowadzanie jednego formalizmu do innego jest powszechnie stosowanym rozwiązaniem. Tłumaczenie do automatów czasowych zostało wykonane dla algebry procesów ATP [NSY92], języka ET-LOTOS [DOY94] i języka Esterel wzbogaconego o zależności czasowe [BCP+01]. Wspólną cechą tych modeli jest brak mechanizmów komunikacji asynchronicznej. Współdziałanie między procesami opiera się jedynie na synchronicznym wykonywaniu pewnych akcji. W tym przypadku tłumaczenie polega na zbudowaniu osobnego automatu czasowego dla każdego procesu i naturalnym przeniesieniu metody synchronizacji (w automatach czasowych przejścia o tej samej etykiecie wykonywane są wspólnie). Pewne trudności pojawiają się, gdy język opisu systemu czasowego wykorzystuje komunikację asynchroniczną, na przykład przez wymianę komunikatów, ponieważ takie bezpośrednie tłumaczenie nie jest wtedy możliwe.

W rozprawie zostały przedstawione dwie metody generowania automatów czasowych dla systemu czasowego opisanego w języku bazowym. Pierwsza metoda konstruuje jeden automat czasowy (tak zwany automat globalny) dla całego systemu. Lokacja globalnego automatu czasowego odpowiada konfiguracji systemu określonej przez stany wszystkich jego składowych (procesów, zmiennych i buforów komunikacyjnych). Intuicyjnie, w globalnym automacie czasowym istnieje przejście z jednej lokacji do drugiej, jeżeli system będac w konfiguracji odpowiadającej pierwszej lokacji może wykonać akcję, w wyniku której system znajdzie się w konfiguracji odpowiadającej drugiej lokacji. Ograniczenia czasowe wykonywanych akcji są modelowane przez odpowiednie warunki na zegarach automatu.

Niektóre metody i narzędzia weryfikacji modelowej działają dużo bardziej efektywnie, jeżeli zamiast jednego (najczęściej dużego) automatu mogą operować na zbiorze mniejszych automatów, opisujących poszczególne składowe systemu. Oczywiście, w czasie weryfikacji analizowane są przebiegi automatu produktowego automatów składowych. Dlatego opracowano drugą metodę budującą zbiór automatów czasowych dla systemu czasowego, w którym poszczególne automaty odpowiadają składowym systemu. Lokacje składowych automatów czasowych odpowiadają stanom poszczególnych elementów systemu.

W rozprawie zostało wykazane, że dana własność wyrażona jako formuła logiki CTL* jest prawdziwa dla systemu opisanego w języku bazowym wtedy i tylko wtedy, gdy jest prawdziwa dla automatu globalnego zbudowanego dla tego systemu. Analogiczne twierdzenie jest prawdziwe dla zbioru automatów czasowych. Dowód polega na wykazaniu, że modele programu i automatu czasowego (globalnego lub produktowego automatów składowych) skonstruowanego dla tego programu są w relacji silnej bisymulacji.

W konstrukcji zwrócono szczególną uwagę na to, aby liczba zegarów w generowanych automatach była jak najmniejsza, ponieważ złożoność problemu weryfikacji modelowej automatów czasowych jest wykładnicza względem liczby użytych zegarów [AD94].

Rozważamy również metody generowania automatów dla pewnych podklas systemów języka bazowego, takich jak systemy nie wykorzystujące mechanizmów ko-

munikacji asynchronicznej.

Implementacja opisanej metody generowania automatu globalnego dla systemu w języku bazowym została wykonana przez Annę Doroś i była przedmiotem jej pracy magisterskiej pt. *Translacja z pewnego języka specyfikacji do automatów z czasem* obronionej w 2003 roku. Metodę generowania zbioru automatów zaimplementowała autorka rozprawy wspólnie z Pawłem Janowskim. Obydwie implementacje stanowią część systemu weryfikacyjnego VerICS [DJJ+03] dostępnego pod adresem `http://verics.ipipan.waw.pl`.

Redukcja przestrzeni stanów metodą cięcia

Głównym problemem w automatycznej weryfikacji modelowej jest wykładnicza eksplozja liczby stanów modelu, która ogranicza rozmiary systemów, z którymi można się zmierzyć w procesie weryfikacji. Z tego powodu metody ograniczania eksplozji stanów mają duże znaczenie praktyczne. Dla systemów bez czasu zostały opracowane różne metody, które redukują rozmiary przestrzeni stanów użytej w procesie weryfikacji. Metody te są dostosowane do używanych języków specyfikacji, w których wyrażane są weryfikowane własności, takich jak logiki temporalne. Do najważniejszych z nich należą metody redukcji częściowo-porządkowych [Val89, GKPP99], abstrakcje [DGG94] czy metody symboliczne [Bry86, McM93]. Niektóre z powyższych metod zostały uogólnione dla systemów z czasem [BJLY98, DGKK98, DT98].

Jednym z wymienionych sposobów jest konstrukcja abstrakcyjnego modelu systemu, który zachowuje własności modelu konkretnego. W rozprawie przedstawiamy metodę konstrukcji abstrakcyjnego modelu opartą na statycznej analizie opisu systemu, a konkretnie na metodzie cięcia (ang. *slicing*).

Metoda ta zaproponowana przez Weisera [Wei84] polega na wykorzystaniu statycznej analizy opisu systemu do eliminacji jego nieistotnych fragmentów, czyli w naszym przypadku takich, które nie mają wpływu na weryfikowane własności. Pozwala to na uproszczenie opisu, co z kolei powoduje zmniejszenie przestrzeni stanów potrzebnej do jego weryfikacji. Zaletą metody jest to, że pełna przestrzeń stanów nie musi być generowana, ponieważ można ją skonstruować już dla zredukowanego opisu systemu.

Pierwotnie metoda cięcia była stosowana do uproszczania procesów śledzenia (ang. *debugging*) i symulacji programów. Była także wykorzystywana do redukcji przestrzeni stanów w weryfikacji modelowej systemów bez czasu opisanych w Promeli [MT98] oraz wielowątkowych programów (również nie uwzględniających zależności czasowych) w języku Java [HCD+99]. Analiza systemów czasowych różni się od analizy systemów bezczasowych, ponieważ ograniczenia nałożone na czas wykonywania akcji systemu wprowadzają nowego rodzaju zależności.

Metoda cięcia zależy od danej formuły logicznej wyrażającej własność, której prawdziwość chcemy sprawdzić, a dokładniej — od zbioru zmiennych zdaniowych, które wchodzą w skład formuły. Punktem wyjściowym jest ustalenie tak zwanego *kryterium cięcia*, czyli zbiorów akcji i stanów, od których bezpośrednio zależą wartości zmiennych zdaniowych z formuły. Następnie, algorytm redukcji bada reku-

rencyjnie zależności między akcjami i stanami poszczególnych procesów systemu zaczynając od kryterium cięcia.

W systemie czasowym występują trzy podstawowe rodzaje zależności. Zależność danych i zależność przepływu sterowania są klasycznymi pojęciami metody cięcia [Tip95]. Natomiast pojęcie zależności czasowej jest nowym rodzajem zależności występującym tylko w systemach czasowych, które zostało po raz pierwszy zdefiniowane przez autorkę rozprawy.

Intuicyjnie, akcja a_1 zależy od akcji a_2 i jest to zależność danych, jeżeli akcja a_2 ma wpływ na wartość danych używanych w akcji a_1. Zależności przepływu sterowania i czasu dotyczą wyłącznie stanów jednego procesu, to znaczy nie ma zależności tych typów między stanami dwu różnych procesów. Definicja zależności przepływu sterowania mówi, że stan q_1 zależy od stanu q_2, jeżeli osiągalność stanu q_1 zależy od decyzji podjętej w stanie q_2. Natomiast, stan q_1 zależy od stanu q_2 i jest to zależność czasowa, jeżeli czas w jakim stan q_1 będzie osiągnięty zależy od decyzji podjętej w stanie q_2.

Na podstawie relacji zależności, algorytm redukcji oblicza, które elementy systemu (akcje i stany) są *istotne* dla danego kryterium cięcia, czyli mają wpływ na badaną własność. Początkowo elementy należące do kryterium cięcia są zaznaczone jako istotne. Następnie, algorytm sukcesywnie zaznacza jako istotne te elementy, od których zależą istotne elementy zaznaczone wcześniej. Zredukowany system składa się wyłącznie z istotnych fragmentów oryginalnego systemu.

W weryfikacji modelowej własności programów są opisywane między innymi przez formuły logik temporalnych będących podzbiorami logiki CTL* (LTL, CTL) lub TCTL (czasowe rozszerzenie CTL). Ze względu na charakter proponowanych redukcji (między innymi eliminację pewnych akcji) w naszych rozważaniach ograniczamy się do podlogik nie zawierających operatora następnego kroku (X). Zachowanie odpowiednich klas własności zapewniamy przez wykazanie odpowiedniej relacji między modelem konkretnym a abstrakcyjnym. Dla logiki $\mathrm{CTL}^*_{-\mathrm{X}}$ jest to relacja bisymulacji z powtórzeniami (ang. *stuttering bisimulation*).

Uwagi końcowe

Podsumowując, do głównych wyników rozprawy należy opracowanie:

- Języka bazowego do opisu systemów czasowych,
- Metody generowania automatów czasowych dla specyfikacji w języku bazowym,
- Metody redukcji systemów czasowych wykorzystującej metodę cięcia.

Przedstawiona metoda generowania automatów czasowych została wykorzystana między innymi w procesie werfikacji systemów (także bezczasowych) opisanych w językach Estelle [DJJ02], UML [NPLK06], Promela [NDJ06] i Java [OWS06], a także do weryfikacji protokołów kryptograficznych uwzględniających zależności czasowe [JP07], gdzie jako język opisu został wykorzystany język bazowy.

Literatura

[AD90] R Alur and D Dill. Automata for modelling real-time systems. In *Proc. of the Int. Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *LNCS*, pages 322–335. Springer-Verlag, 1990.

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[BCP+01] V. Bertin, E. Closse, M. Poize, J. Pulou, J. Sifakis, P. Venier, D. Weil, and S. Yovine. Taxys = esterel + kronos: A tool for verifying real-time properties of embedded systems. *Proc. of the 40th IEEE Conf. Decision and Control*, 3, 2001.

[BG00] G. Bollella and J. Gosling. The real-time specification for Java. *Computer*, 33(6):47–54, 2000.

[BJLY98] J. Bengtsson, B. Jonsson, J. Lilius, and W. Yi. Partial order reductions for timed systems. In *Proc. of the 9th Int. Conf. on Concurrency Theory (CONCUR'98)*, volume 1466 of *LNCS*, pages 485–500. Springer-Verlag, 1998.

[Bry86] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transaction on Computers*, 35(8):677–691, 1986.

[DGG94] D. Dams, O. Grumberg, and R. Gerth. Abstract interpretation of reactive systems: Abstractions preserving ACTL*, ECTL* and CTL*. In *Proc. of the IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET'94)*. Elsevier, 1994.

[DGKK98] D. Dams, R. Gerth, B. Knaack, and R. Kuiper. Partial-order reduction techniques for real-time model checking. In *Proc. of the 3rd Int. Workshop on Formal Methods for Industrial Critical Systems*, pages 157–169, 1998.

[DJJ02] A. Doroś, A. Janowska, and P. Janowski. From specification languages to Timed Automata. In *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'02)*, volume 161(1) of *Informatik-Berichte*, pages 117–128. Humboldt University, 2002.

[DJJ+03] P. Dembiński, A. Janowska, P. Janowski, W. Penczek, A. Półrola, M. Szreter, B. Woźna, and A. Zbrzezny. VerICS: A tool for verifying timed automata and Estelle specifications. In *Proc. of the 9th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03)*, volume 2619 of *LNCS*, pages 278–283. Springer-Verlag, 2003.

[DJPV02] W. Damm, B. Josko, A. Pnueli, and A. Votintseva. Understanding UML: A formal semantics of concurrency and communication in real-time uml, 2002.

[DOTY95] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Hybrid Systems III*, volume 1066 of *LNCS*, pages 208–219. Springer-Verlag, 1995.

[DOY94] C. Daws, A. Olivero, and S. Yovine. Verifying ET-LOTOS programs with KRONOS. In *Proc. of the 7th IFIP WG.G.1 Int. Conf. on Formal Description Techniques (FORTE'94)*, pages 227–242. Chapman & Hall, October 1994.

[DT98] C. Daws and S. Tripakis. Model checking of real-time reachability properties using abstractions. In *Proc. of the 4th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'98)*, volume 1384 of *LNCS*, pages 313–329. Springer-Verlag, 1998.

[GKPP99] R. Gerth, R. Kuiper, D. Peled, and W. Penczek. A partial order approach to branching time logic model checking. *Information and Computation*, 150:132–152, 1999.

[HCD+99] J. Hatcliff, J. C. Corbett, M. B. Dwyer, S. Sokolowski, and H. Zheng. A formal study of slicing for multi-threaded programs with JVM concurrency primitives. In *Proc. of the Int. Symposium on Static Analysis (SAS'99)*, pages 1–18, 1999.

[HNSY94] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[ISO97] ISO/IEC 9074(E), Estelle - a formal description technique based on an extended state-transition model. International Standards Organization, 1997.

[JP07] G. Jakubowska and W. Penczek. Is your security protocol on time? In *Proc. of Int. Symposium on Fundamentals of Software Engineering (FSEN'07)*, volume XXX of *LNCS*. Springer-Verlag, 2007.

[McM93] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[MF76] P. Merlin and D. J. Farber. Recoverability of communication protocols – implication of a theoretical study. *IEEE Trans. on Communications*, 24(9):1036–1043, 1976.

[MT98] L. Millett and T. Teitelbaum. Slicing promela and its applications to model checking, simulation and protocol understanding. In *Proc. of the 4th Int. SPIN Workshop*, 1998.

[MT01] A. Mitschele-Thiel. *Systems Engineering with SDL - Developing Performance-Critical Communication Systems.* John Wiley & Sons, 2001.

[NDJ06] W. Nabiałek, P. Dembiński, and P. Janowski. Towards Promela verification using VeriCS. In *Proc. of the Workshop on Concurency, Specification and Programming (CSP'06)*, volume 206 of *Informatik Berichte*, pages 219–230. Humboldt University, 2006.

[NPLK06] A. Niewiadomski, W. Penczek, S. Lasota, and J. Kowalski. Weryfikacja UML z wykorzystaniem systemu VeriCS. In *Systemy Czasu Rzeczywistego 2006*, Systemy informatyczne z ograniczeniami czasowymi, 2006.

[NSY92] X. Nicollin, J. Sifakis, and S. Yovine. Compiling real-time specifications into extended automata. *Software Engineering*, 18(9):794–804, 1992.

[OWS06] M. Orzechowski, B. Woźna, and T. Siwiak. Towards verification of Java programs in VeriCS. Technical Report 997, ICS PAS, 2006.

[PL00] P. Pettersson and K. G. Larsen. UPPAAL2k. *Bulletin of the European Association for Theoretical Computer Science*, 70:40–44, February 2000.

[Tip95] F. Tip. A survey of program slicing techniques. *Journal of Programming Languages*, 3, 1995.

[Val89] A. Valmari. Stubborn sets for reduced state space generation. In *Proc. of the 10th Int. Conf. on Applications and Theory of Petri Nets (ICATPN'89)*, volume 483 of *LNCS*, pages 491–515. Springer-Verlag, 1989.

[Wei84] M. Weiser. Program slicing. *IEEE Trans. on Software Eng.*, 10(4), 1984.