

ROZDZIAŁ YY

VERICS 2006 – WERYFIKATOR DLA SYSTEMÓW CZASOWYCH I WIELOAGENTOWYCH

Rozdział przedstawia obecny stan rozwoju systemu Verics - weryfikatora przeznaczonego do testowania systemów czasu rzeczywistego i systemów wieloagentowych. W zależności od rodzaju badanego systemu umożliwia on testowanie różnych klas własności - od osiągalności po formuły logik temporalnych, epistemicznych i deontycznych. Dostępne w systemie metody weryfikacji obejmują zarówno techniki symboliczne (bazujące na metodach SAT), jak i enumeratywne (wykorzystujące modele abstrakcyjne badanych systemów). Niniejszy opis poświęcony jest przede wszystkim nowym elementom weryfikatora: możliwości testowania systemów wieloagentowych oraz rozszerzeniom i ulepszeniom dostępnych wcześniej technik weryfikacji systemów czasu rzeczywistego.

1. WPROWADZENIE

Rozdział przedstawia obecny stan rozwoju systemu Verics – weryfikatora przeznaczonego do testowania systemów czasu rzeczywistego (ang. *real-time systems*) i systemów wieloagentowych (ang. *multi-agent systems*), opracowanego i rozwijanego w Instytucie Podstaw Informatyki PAN. W zależności od rodzaju badanego systemu Verics umożliwia testowanie różnych klas własności – od osiągalności stanu o określonych cechach (ang. *reachability*) po bardziej skomplikowane właściwości

Magdalena Kacprzak: Politechnika Białostocka; ul. Wiejska 45A, 15-351 Białystok;
mdkacprzak@wp.pl

Wojciech Nabiałek, Artur Niewiadomski: Instytut Informatyki, Akademia Podlaska;
ul. Sienkiewicza 51, 08-110 Siedlce; wojtek.artur@iis.ap.siedlce.pl,

Wojciech Penczek: IPI PAN, ul. Ordona 21, 01-237 Warszawa i Instytut Informatyki, Akademia Podlaska, ul. Sienkiewicza 51, 08-110 Siedlce; penczek@ipipan.waw.pl

Agata Półroła: Wydział Matematyki i Informatyki Uniwersytetu Łódzkiego; ul. Banacha 22,
90-238 Łódź; polrola@math.uni.lodz.pl

Maciej Szreter: IPI PAN, ul. Ordona 21, 01-237 Warszawa; mszreter@ipipan.waw.pl

Bożena Woźna, Andrzej Zbrzezny: Instytut Matematyki i Informatyki, Akademia im. Jana Długosza; ul. Armii Krajowej 13/15, 42-200 Częstochowa; b.wozna.a.zbrzezny@ajd.czest.pl

systemów, opisywane za pomocą formuł logik temporalnych, epistemicznych i deontycznych (w tym z parametrami czasowymi). Metody weryfikacji zaimplementowane w Vericsie obejmują techniki bazujące na testowaniu spełnialności formuł zdaniowych (metody SAT) oraz metody enumeratywne, polegające na wykorzystaniu wygenerowanego uprzednio odpowiedniego *modelu abstrakcyjnego* testowanego systemu. Nasze pierwsze prace dotyczące systemu Verics [2,3] poświęcone były przede wszystkim weryfikacji systemów czasowych. Niniejszy opis prezentuje nowe możliwości weryfikatora: testowanie własności systemów wieloagentowych za pomocą metod bazujących na SAT oraz rozszerzenia i ulepszenia technik weryfikacyjnych dla systemów czasowych dostępnych w poprzedniej wersji programu.

2. PRZEGLĄD DOSTĘPNYCH NARZĘDZI

VerICS umożliwia testowanie systemów czasu rzeczywistego i systemów wieloagentowych. Przegląd i porównanie narzędzi weryfikacyjnych dla systemów czasu rzeczywistego, uwzględniające również nasze narzędzie, przedstawione zostały w pracy [13]. Porównując z kolei weryfikatory przeznaczone dla systemów wieloagentowych stwierdzamy, że VerICS jest (według naszej wiedzy) jednym z trzech dostępnych narzędzi umożliwiających ich bezpośrednią weryfikację (tzn. weryfikację bez uprzedniej translacji), w tym jedynym używającym do tego celu metod bazujących na SAT. Pozostałe wspomniane powyżej narzędzia: MCMAS [15] i MCK [5] wykorzystują techniki oparte na BDD (binarnych diagramach decyzyjnych). Inne programy [1,18] umożliwiają testowanie systemów wieloagentowych poprzez ich translację do postaci akceptowanej przez weryfikatory „ogólnego przeznaczenia”, takie jak SPIN czy JavaPathFinder.

3. PODSTAWY TEORETYCZNE

Podstawowym formalizmem, służącym do modelowania systemów poddawanych weryfikacji za pomocą Vericsa, są sieci komunikujących się automatów (w tym również automatów czasowych). Automaty czasowe (także z ograniczeniami czasowymi wyrażanymi za pomocą różnic wartości zegarów) stosowane są głównie do modelowania systemów czasowych. Natomiast systemy wieloagentowe opisywane są zazwyczaj za pomocą automatów bezczasowych, niekiedy rozszerzonych o elementy pozwalające na specyfikację pewnych dodatkowych własności (na przykład automaty deontyczne w [7]).

Stan globalny rozpatrywanego systemu jest krotką złożoną ze stanów lokalnych automatów w sieci N opisującej ten system. Zbiór wszystkich nieskończonych *przebiegów* (tj. nieskończonych ciągów przekształceń systemu, rozpoczynających się od

stanu początkowego) systemu czasowego modelowanego przez N stanowi *drzewo wykonań systemu*. Etykietując stany systemu zmiennymi zdaniowymi z pewnego zbioru PV prawdziwymi w tych stanach, przekształcamy drzewo wykonań w *model*, w którym interpretowane są formuły czasowych i bezczasowych logik temporalnych (jak CTL - ang. *Computation Tree Logic* i TCTL - ang. *Timed Computation Tree Logic*), wyrażające testowane własności. W przypadku systemów wieloagentowych powyższy model wzbogacany jest dodatkowo o epistemiczną lub deontyczną relację dostępności (ang. *epistemic/deontic accessibility relation*). Otrzymana w ten sposób struktura pozwala interpretować formuły zawierające operatory temporalne, epistemiczne (umożliwiające wyrażenie *wiedzy* agenta [4]) i deontyczne (umożliwiające wnioskowanie o *poprawności zachowania* agenta).

Metody weryfikacyjne dla systemów wieloagentowych bazujące na SAT obejmują ograniczoną i nieograniczoną weryfikację modelową (oznaczaną odpowiednio jako BMC i UMC, od ang. *Bounded* i *Unbounded Model Checking*). BMC jest techniką przeznaczoną przede wszystkim do poszukiwania kontrprzykładów dla formuł uniwersalnych fragmentów logik epistemicznych czasu rozgałęzionego. Idea tej metody polega na zakodowaniu rozwinięcia relacji przejścia modelu do pewnej głębokości k w postaci formuły boolowskiej φ_1 , zakodowaniu negacji testowanej formuły nad ścieżkami o długości k w postaci formuły boolowskiej φ_2 , a następnie automatycznym sprawdzeniu, czy koniunkcja $\varphi_1 \wedge \varphi_2$ jest spełnialna. Jeśli tak jest, to kontrprzykład istnieje. W pierwszym kroku algorytmu rozpatrywana jest ścieżka o długości $k=1$, w dalszych krokach k jest stopniowo zwiększane, aż do znalezienia kontrprzykładu lub do osiągnięcia *średnicy* modelu. W praktyce wykazanie, że kontrprzykład nie istnieje, jest zazwyczaj niewykonalne ze względu na ograniczenia nałożone na pamięć i czas zużywane przez testery SAT.

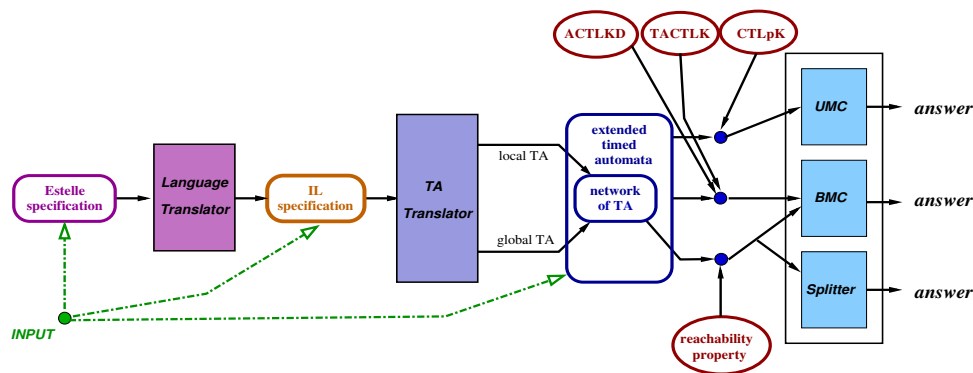
W odróżnieniu od BMC nieograniczona weryfikacja modelowa jest metodą umożliwiającą testowanie dowolnych formuł rozpatrywanej logiki. Technika ta wykorzystuje charakterystykę podstawowych modalności poprzez formuły QBF (kwantyfikowane formuły boolowskie, ang. *quantified boolean formulas*) oraz algorytm translacji formuł QBF i równań stałopunktowych nad formułami QBF do postaci formuł zdaniowych. Rezultatem powyższych przekształceń jest formuła boolowska ψ , charakteryzująca wszystkie stany modelu w których testowana formuła jest prawdziwa. Sprawdzenie, czy formuła jest prawdziwa w modelu, polega na przetestowaniu spełnialności koniunkcji ψ i formuły kodującej stan początkowy.

Aktualna wersja Vericsa - 2006 zawiera implementację UMC dla formuł logiki CTLpK (logiki CTL rozszerzonej o operatory wiedzy i operatory opisujące przeszłość) oraz implementację BMC dla logik ACTLKD (uniwersalnego fragmentu logiki CTL rozszerzonej o operatory wiedzy i deontyczne) i TACTLK (uniwersalnego fragmentu czasowego CTL rozszerzonego o operatory wiedzy) [7,8,14,19,20]. Wersja systemu opisana w pracach [2,3] rozszerzona została również o nowe moduły przeznaczone do weryfikacji systemów czasu rzeczywistego. Moduły te włączają implementację BMC umożliwiającą dowodzenie (nie)osiągalności (również dla automatów z różnicami

zegarów) oraz implementację UMC dla formuł logiki CTL i pewnego podzbioru automatów czasowych [17,21,22].

4. IMPLEMENTACJA

Ogólna architektura systemu Verics przedstawiona jest na rys. 1. System składa się z następujących modułów:



Rys. 1. Architektura systemu Verics

- **translatora z języka Estelle do języka bazowego IL**, który umożliwia weryfikację specyfikacji zapisanych w podzbiorze języka Estelle [6] (standaryzowanego języka specyfikacji protokołów komunikacyjnych i systemów rozproszonych);
- **translatora z języka IL do automatów czasowych**, który na podstawie specyfikacji w języku bazowym (ang. *Intermediate Language* – IL) generuje odpowiadającą jej sieć automatów czasowych lub automat produktowy (globalny);
- **modułu BMC** - zbioru programów implementujących ograniczoną weryfikację modelową dla klas własności wymienionych na rysunku. Moduł korzysta z SAT-testerów MiniSAT [9], Rsat [16] lub Zchaff [23]. Może być również skonfigurowany do pracy z innymi testerami;
- **modułu UMC**, zawierającego wstępne wersje implementacji nieograniczonej weryfikacji modelowej dla opisanych wyżej klas własności. Moduł jest zintegrowany ze zmodyfikowaną wersją SAT-testera Zchaff;
- **modułu Splitter**, umożliwiającego testowanie osiągalności w modelach abstrakcyjnych budowanych dla automatów czasowych.

Moduły weryfikacyjne Vericsa oraz translatory zostały zaimplementowane w języku C++. Usługi systemu dostępne są poprzez interfejs napisany w języku Java. Aktualną wersję systemu (binaria kompletnego programu uruchamianego pod Linuxem lub wersji klient możliwej do uruchomienia pod dowolnym systemem ze wsparciem dla Javy) można pobrać ze strony internetowej <http://verics.ipipan.waw.pl>.

5. WYNIKI EKSPERYMENTALNE

W poniższym podrozdziale zaprezentujemy efektywność systemu Verics przytaczając wyniki weryfikacji kilku znanych systemów czasowych i wieloagentowych. Wszystkie przykłady reprezentują typowe (skalowalne) programy testowe, które zazwyczaj badane są podczas sprawdzania efektywności modułów weryfikacyjnych niskiego poziomu i tym samym pozwalają na porównanie naszego narzędzia z innymi.

- **Biesiadujący kryptografowie:** system składa się z n agentów – kryptografów jedzących obiad w restauracji. Uczestnicy obiadu chcieliby wiedzieć kto za ten obiad zapłacił - któryś z uczestników czy może agencja. W celu zachowania anonimowości fundatora każdy z agentów rzuca monetą i ogłasza, czy wyniki rzutów jego monety i sąsiada znajdującego się z prawej strony są jednakowe czy różne. Jeśli kryptograf zapłacił za posiłek, komunikuje stan przeciwny do rzeczywistego. Nieparzysta liczba wygłoszonych różnic wskazuje na to, że fundatorem jest jeden z kryptografów, ale nie wiadomo który z nich. W przeciwnym przypadku fundatorem jest agencja. Poprawność powyższego protokołu pokazujemy dowodząc prawdziwość specyfikujących go formuł lub fałszywość formuł przeciwnych. Testowane własności systemu to:

$$\varphi_D^1 := AG(\text{odd} \wedge \neg \text{paid}_1 \Rightarrow K_1(\text{paid}_2 \vee \dots \vee \text{paid}_n))$$

$$\varphi_D^2 := AG(\text{odd} \wedge \neg \text{paid}_1 \Rightarrow (K_1(\text{paid}_2) \vee \dots \vee K_1(\text{paid}_n)))$$

$$\varphi_D^3 = AG(\neg \text{paid}_1 \Rightarrow K_1(\text{paid}_2 \vee \dots \vee \text{paid}_n))$$

Tabela 1. Wyniki weryfikacji protokołu biesiadujących kryptografów

Formuła	N	BMC [s/MB]	SAT [s/MB]	n	UMC [s/MB]
φ_D^1	4	0.8 / 9.2	81224 / -	17	367.0 / -
φ_D^2	4	15.97 / 25.8	9359.24 / -	9	392.0 / -
φ_D^3	1000	329 / 1885.0	29.67 / -	15	421.0 / -

Formuła φ_D^1 wyraża, że zawsze gdy liczba wypowiedzianych przez kryptografów różnic jest nieparzysta i pierwszy kryptograf nie zapłacił za obiad, to wie on, że któryś z pozostałych kryptografów jest fundatorem.

Formuła φ_D^2 jest zaprzeczeniem stawianych protokołowi wymagań i wyraża, że zawsze gdy liczba wypowiedzianych różnic jest nieparzysta i pierwszy kryptograf nie fundował obiadu, to wie on, kto płacił. Formuła ta jest oczywiście fałszywa, gdyż zgodnie z protokołem ta informacja powinna być poufna i żaden z kryptografów nie może wiedzieć, który z nich dokonał płatności. Nieparzysta liczba różnic może wskazywać jedynie na fakt, że fundatorem nie jest agencja.

Formuła φ_D^3 wyraża, że zawsze gdy pierwszy kryptograf nie zapłacił za obiad, to wie, że fundatorem jest któryś z pozostałych kryptografów. Taka własność jest fałszywa w badanym modelu, gdyż płatnikiem może być również agencja.

Wśród wyników zaprezentowanych w tabeli 1 na szczególną uwagę zasługuje fakt, iż zastosowane tutaj metody weryfikacji wzajemnie się uzupełniają. Dla formuł φ_D^1 i φ_D^2 zdecydowanie lepsze wyniki uzyskujemy stosując metodę UMC, natomiast w przypadku formuły φ_D^3 bardziej efektywna okazuje się metoda BMC - nawet do 1000 kryptografów! Jest to możliwe, ponieważ kontrprzykład znajdujący się na bardzo krótkiej ścieżce. Dla pierwszych dwóch formuł model musi być przeszukany do pełnej głębokości, a ponadto formuły muszą być testowane na wielu ścieżkach symbolicznych.

- **Protokół wzajemnego wykluczania Fischera:** system składa się z n jednakowych procesów próbujących uzyskać dostęp do sekcji krytycznej oraz procesu koordynującego ich działania. Zachowanie procesów zależne jest od wartości parametrów czasowych δ i Δ (zależność $\delta < \Delta$ powoduje naruszenie zasady wzajemnego wykluczania).

Badana własność systemu to:

$$\varphi_M := EF \left(\bigvee_{1 \leq i, j \leq n, i \neq j} (crit_i \wedge crit_j) \right)$$

Własność ta oznacza, że dwa różne procesy mogą znaleźć się jednocześnie w sekcji krytycznej i jest prawdziwa wtedy, gdy zasady wzajemnego wykluczania są naruszone.

Tabele 2 i 3 prezentują wyniki weryfikacji protokołu Fischera za pomocą metody ograniczonej weryfikacji modelowej. W tabeli 2 kontrprzykład, który jest dowodem naruszenia zasady wzajemnego wykluczania, jest znajdujący się na głębokości 17. Natomiast w tabeli 3 znajdują się wyniki testowania **niesiagłości** badanej własności w systemie, który spełnia zasadę wzajemnego wykluczania. W tym celu naprzemiennie sprawdzane jest, czy w systemie istnieje

Tabela 2. Wyniki weryfikacji protokołu Fischera z parametrami naruszającymi zasadę wzajemnego wykluczenia (n=80, $\Delta=2$, $\delta=1$)

k	BMC		MiniSAT		
	t[s]	MB	t[s]	MB	spełn.
0	0.2	4.6	0.0	2.9	nie
2	1.0	8.5	0.8	9.8	nie
5	2.0	13.8	27.7	37.3	nie
8	3.2	19.2	1270.8	521.1	nie
11	4.3	24.6	2408.3	852.1	nie
14	5.6	29.9	4267.4	1527.7	nie
17	7.0	35.3	590.8	419.4	TAK
Σ	23.3	35.3	8565.8	1527.7	

Tabela 3. Wyniki weryfikacji protokołu Fischera zachowującego zasadę wzajemnego wykluczenia (n=10, $\Delta=1$, $\delta=2$)

k	ścieżka wolna	BMC		MiniSAT		
		t[s]	MB	t[s]	MB	spełn.
0	-	0.0	2.8	0.0	1.7	Nie
2	+	0.0	3.2	0.0	2.0	Tak
2	-	0.0	3.2	0.0	2.1	Nie
...						
20	+	0.2	6.8	1.6	5.6	Tak
20	-	0.2	6.8	5.9	8.8	Nie
...						
50	+	0.6	12.6	2670.9	206.1	Tak
50	-	0.5	12.6	148.1	71.6	Nie
53	+	0.6	13.2	3992.1	233.1	NIE
Σ		10.4	13.2	10037.9	233.1	

'wolna' ścieżka o podanej długości [21] i czy testowana własność jest prawdziwa na ścieżce o takiej długości, której pierwszym stanem jest stan początkowy systemu. Następnie długość ścieżek jest zwiększana. Okazuje się, że w badanym systemie, wolna ścieżka o długości 53 nie istnieje, a na wszystkich krótszych ścieżkach badana własność nie jest prawdziwa. Stąd wniosek, że własność jest nieprawdziwa.

Tabela 4. Porównanie wyników weryfikacji protokołu Fischera metodą UMC (Verics) z narzędziami Uppaal i RED

Parametry	Czas [s]		
	Uppaal	RED	VerICS UMC
$N=10, \Delta=1, \delta=2$	37	53	34
$N=11, \Delta=1, \delta=2$	121	141	46
$N=12, \Delta=1, \delta=2$	580	304	59
$N=13, \Delta=1, \delta=2$	-	657	88
$N=15, \Delta=1, \delta=2$	-	-	154
$N=18, \Delta=1, \delta=2$	-	-	376
$N=20, \Delta=1, \delta=2$	-	-	491
$N=10, \Delta=3, \delta=4$	33	53	50
$N=11, \Delta=3, \delta=4$	125	133	71
$N=10, \Delta=2, \delta=1$	7	49	97

Tabela 4 prezentuje wyniki porównania metody UMC zaimplementowanej w systemie Verics z istniejącymi na rynku narzędziami weryfikacyjnymi. W większości przypadków Verics wykazuje się lepszą efektywnością.

- **Czasowa wersja protokołu alternującego bitu:** system składa się z dwóch agentów: nadawcy (S) i odbiorcy (R), wymieniających komunikaty za pośrednictwem podwójnego kanału transmisyjnego, który nie gwarantuje dostarczenia danych. Kanał używany do przesłania danych jest wybierany na podstawie analizy czasu obiegu (ang. *round trip time*) bitu kontrolnego. Tym razem weryfikujemy negacje następujących uniwersalnych własności: (φ_A^1) „dla każdego wykonania protokołu jest tak, że jeśli nadawca otrzymał potwierdzenie w czasie nie przekraczającym t_1 i wartość wysłanego bitu była równa zero, to odbiorca zna wartość tego bitu”, oraz (φ_A^2) „dla każdego wykonania protokołu jest tak, że jeśli nadawca otrzymał potwierdzenie w czasie nie przekraczającym t_1 i wartość wysłanego bitu była równa zero, to nadawca wie, że odbiorca zna wartość tego bitu”. Obydwie powyższe własności można wyrazić za pomocą następujących formuł TACTLK:

$$\varphi_A^1 := AG_{[0,t_1]}((recack \wedge bit0) \Rightarrow K_R(bit0))$$

$$\varphi_A^2 := AG_{[0,t_1]}((recack \wedge bit0) \Rightarrow K_S K_R(bit0))$$

Tabela 5 prezentuje wyniki dla protokołu alternującego bitu, jakie uzyskano stosując moduł BMC Vericsa przeznaczony do weryfikacji własności wyrażalnych w logice TACTLK. W przeciwieństwie do poprzednich przykładów, nie przedstawia ona porównania z innymi narzędziami weryfikacyjnymi, gdyż o ile nam wiadomo nie istnieją inne automatyczne weryfikatory dla logiki TACTLK.

Tabela 5. Wyniki weryfikacji czasowego protokołu alternującego bitu

K	BMC		MiniSAT			BMC		MiniSAT		
	t[s]	MB	t[s]	MB	Spełnialne	t[s]	MB	t[s]	MB	Spełnialne
	$\neg\varphi_A^1$					$\neg\varphi_A^2$				
0	0.0	1.4	0.0	3.6	Nie	0.0	1.4	0.0	3.6	Nie
1	0.0	1.8	0.0	3.9	Nie	0.0	1.8	0.0	4.0	Nie
2	0.0	2.3	0.0	4.4	Nie	0.1	2.3	0.0	4.4	Nie
	...									
9	0.9	10.1	0.3	11.2	Nie	5.6	10.3	0.2	11.0	Nie
10	1.2	12.7	0.3	13.1	Nie	8.6	12.7	0.4	13.3	Nie
11	1.6	14.8	1.6	15.4	TAK	13.0	14.8	1.4	15.4	TAK
Σ	5.6	14.8	2.6	15.4		35.0	14.8	2.4	15.4	

6. PODSUMOWANIE

W niniejszym rozdziale przedstawiony został system weryfikacyjny Verics 2006. Modułowa architektura i dostępność wielu formalizmów specyfikacji oraz algorytmów weryfikacyjnych stanowią o elastyczności systemu z perspektywy końcowego użytkownika. W dziedzinie automatycznego testowania poprawności systemów i programów często zdarza się, że określone rodzaje algorytmów wykazują najlepszą efektywność dla pewnych klas własności. Udostępnienie wielu metod zwiększa szansę skutecznej weryfikacji. Przystępny interfejs graficzny ułatwia realizację zadań oraz umożliwia zapoznanie się z dziedziną weryfikacji modelowej.

Obecnie w ramach trzyletniego grantu MNiSW trwają intensywne prace nad rozszerzeniem funkcjonalności systemu Verics o translatory z języków wysokiego poziomu do języka IL i automatów czasowych. Dzięki temu nowa edycja Vericsa (planowana na rok 2008) będzie wspierać weryfikację systemów specyfikowanych w podzbiórach takich języków jak Java, Promela, UML czy też Ada.

LITERATURA DO ROZDZIAŁU

- [1] Bordini R. H., Visser W., Fisher M., Pardavila C., Wooldridge M.: Model Checking Multi-Agent Programs with CASP. In Proc. of the 15th Conf. on Computer-Aided Verification (CAV'03), LNCS, vol. 2725, pp. 110-113. Springer Verlag, 2003.
- [2] Dembiński P., Janowska A., Janowski P., Penczek W., Pórola A., Woźna B., Szreter M., Zbrzezny A.: VerICS: A Tool for Verifying Timed Automata and Estelle Specifications. Proc. of the 9th Int. Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS'03), LNCS, vol. 2619, pp. 278-283. Springer-Verlag, 2003.
- [3] Dembiński P., Janowska A., Janowski P., Penczek W., Pórola A., Woźna B., Szreter M., Zbrzezny A.: VerICS: weryfikator dla automatów czasowych i specyfikacji zapisanych w języku Estelle. Mat. X Konferencji Systemy Czasu Rzeczywistego (SCR'03), str. 17-26. Instytut Informatyki Politechniki Śląskiej, 2003.
- [4] Fagin R., Halpern J., Moses Y., Vardi M.: Reasoning about Knowledge. MIT Press, Cambridge, 1995.
- [5] Gammie P., van der Meyden R.: MCK - Model Checking the Logic of Knowledge. Proc. of CAV'04, LNCS, vol. 3114, pp. 479-483. Springer Verlag, 2004.
- [6] ISO/IEC 9074(E), Estelle – A Formal Description Technique Based on an Extended State-Transition Model. International Standards Organization, 1997.
- [7] Kacprzak M., Lomuscio A., Niewiadomski A., Penczek W., Raimondi F., Szreter M.: Comparing BDD and SAT Based Techniques for Model Checking Chaum's Dining Cryptographers Protocol. Fundamenta Informaticae 72(1-2), pp. 215-234. IOS Press, 2006.
- [8] Lomuscio A., Woźna B., Zbrzezny A.: Bounded Model Checking Real-Time Multi-Agent Systems with Clock Differences: Theory and Implementation. Technical Report RN/06/03, Department of Computer

- Science, University College London, March 2006.
- [9] MiniSat: <http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat>, 2006.
- [10] Nabiałek W., Janowski P.: Towards Promela Verification using Verics, Proc. of CS&P'06, vol. 2, pp. 219-230. Berlin, 2006.
- [11] Niewiadomski A., Penczek W., Lasota S., Kowalski J.: Weryfikacja UML z wykorzystaniem systemu Verics. Mat. XIII Konferencji Systemy Czasu Rzeczywistego (SCR'06) Systemy informatyczne z ograniczeniami czasowymi, str. 79, Wyd. Komunikacji i Łączności, 2006.
- [12] Orzechowski M., Woźna B., Siwiak T.: Towards Verification of Java Programs in Verics. Raport IPI PAN, Nr 997 Warszawa, grudzień 2006.
- [13] Penczek W., Pórola A.: Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach. Springer-Verlag, 2006.
- [14] Penczek W., Lomuscio A.: Verifying Epistemic Properties of Multi-Agent Systems via Bounded Model Checking. Proc. of the 2nd Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'03), pp. 209-216. ACM, 2003.
- [15] Raimondi F., Lomuscio A., MCMAS: a Tool for Verifying Multi-Agent Systems, Proc. of the 12th Int. Conf. on Tools and Algorithms for the Construction and Analysis of System (TACAS'06), LNCS, vol. 3920, pp. 450-454. Springer Verlag, 2006.
- [16] Rsat: <http://reasoning.cs.ucla.edu/rsat>, 2006.
- [17] Szreter M.: SAT-Based Model Checking of Distributed Systems. PhD thesis, ICS PAS, January 2007.
- [18] Wooldridge M., Fisher M., Huet M.-P., Parsons S.: Model Checking Multiagent Systems with MABLE. In Proc. of 1st International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02), vol. II, pp. 952-959. ACM, 2002.
- [19] Woźna B., Lomuscio A., Penczek W.: Bounded Model Checking for Deontic Interpreted Systems. Proc. of the 2nd Int. Workshop on Logic and Communication in Multi-Agent Systems (LCMAS'04), ENTCS, vol. 126, pp. 93-114. Elsevier, 2005.
- [20] Woźna B., Lomuscio A., Penczek W.: Bounded Model Checking for Knowledge and Real Time. Proc. of the 4th Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'05), pp. 165-172. ACM, 2005.
- [21] Zbrzezny A.: Improvements in SAT-Based Reachability Analysis for Timed Automata. Fundamenta Informaticae 60(1-4), pp. 417-434. IOS Press, 2004.
- [22] Zbrzezny A.: SAT-Based Reachability Checking for Timed Automata with Diagonal Constraints. Fundamenta Informaticae 67(1-3), pp. 303-322. IOS Press, 2005.
- [23] Zhang L., Zchaff. <http://www.ee.princeton.edu/~chaff/zchaff.php>, 2001.