Welcome to ATVA!

Welcome to Chiba University! It is a great honor to host ATVA 2016.

You'll find our motto “Always Aim Higher” as well as our communication mark (two overlapping garnet regular pentagons) along the way to the tutorial venue. This mark stands for communication among students and staffs in our 10 faculties, on the basis of our vision “Global, Research, Innovation, Branding, and Synergy.”

From Chiba Station, you can reach several tourist sites such as Asakusa or Akihabara without changing trains. For the coming Tokyo 2020 Olympics, we are now welcoming visitors more than ever before with "OMOTENASHI" hospitality all over Japan.

Thank you for coming to Chiba and Japan. We hope you enjoy your stay.
Sponsors

Gold Sponsors

National Institute of Advanced Industrial Science and Technology (AIST)

Tateisi Science and Technology Foundation

Silver Sponsors

Springer

Chiba University

Chiba Convention Bureau and International Center
Committees

Programme Committee Co-Chairs

Cyrille Artho
AIST, Japan and KTH, Sweden

Axel Legay
Inria Rennes, France

Doron Peled
Bar Ilan University, Israel

Publicity Chair

Takashi Kitamura
AIST, Japan

Local Chairs

Mitsuharu Yamamoto
Chiba University, Japan

Yoshinori Tanabe
Tsurumi University, Japan
Venue

Mitsui Garden Hotel (Oct. 17–19)

The conference keynotes and research presentations take place at the Mitsui Garden Hotel.

People who stay at the Keisei Hotel Mira Mare can reach the Mitsui Garden Hotel on foot in about 7 minutes.

The grey line in the middle (タウンライナー) is the Chiba Monorail.

Rooms used at the Mitsui Garden Hotel

**Main conference**  Tenpyo (天平, 4th floor)

**Lunch**  varies, see signs

**Reception**  Hakuho (白鳳, 4th floor)

**Banquet**  Heian-Higashi (平安東, 3rd floor)
Mitsui Garden Hotel to Chiba University

Chiba University will host the tutorials on Oct. 20, from 9:00.

1. Leave the hotel at 8:25, walk to **JR Chiba Station** (10–15 minutes).
2. Take the Chuo-Sobu Line (local, yellow) towards Mitaka (platform 1 or 2); exit at JR **Nishi-Chiba station (1 stop, 2 minutes)**.
3. Take the North Exit (北口), enter Chiba University through its South Gate, Continue to University Library, a.k.a. Academic Link Center. (10 minutes).

![Map of Mitsui Garden Hotel Chiba to Chiba University](image)
Detailed directions to Chiba University Library, Building “I”:

Tutorials take place in University Library, Building "I" (content studio Hikari)

Detailed view of Library Building, a.k.a. Academic Link Center.
- Tutorials take place in the “I” building, in “Content Studio Hikari” on the 1st floor.
Keisei Miramare Hotel to Chiba University

Chiba University will host the tutorial on Oct. 20, from 9:00.

1. Leave the hotel at 8:25 via Chiba Chuo Station (connected to the hotel).
2. Take the Keisei Chiba Line towards Keisei-Tsudanuma (platform 1); exit at Keisei Modoridai station (4 stops, 5 minutes).
3. Take the exit, enter Chiba University through its Main Gate, Continue to University Library, a.k.a. Academic Link Center. (10 minutes).

Keisei Hotel Mira Mare to Chiba University

8:26 AM - 8:38 AM (12 min)
Detailed directions to Chiba University Library, Building “I”:

Detailed view of Library Building, a.k.a. Academic Link Center.

- Tutorials take place in the “I” building, in “Content Studio Hikari” on the 1st floor.
Invited speakers

Masahiro Fujita

Masahiro Fujita received his Ph.D. in Information Engineering from the University of Tokyo in 1985 on his work on model checking of hardware designs. From 1993 to 2000, he was director at Fujitsu Laboratories of America and headed a hardware formal verification group. Since March 2000, he has been a professor at VLSI Design and Education Center of the University of Tokyo.

Masahiro Fujita has done innovative work in the areas of hardware verification, synthesis, testing, and software verification—mostly targeting embedded software and web-based programs. He has authored and co-authored 10 books, and has more than 300 publications. He has been involved as program and steering committee member in many prestigious conferences on CAD, VLSI designs, software engineering, and more. His current research interests include synthesis and verification in SoC (System on Chip), hardware/software co-designs targeting embedded systems, digital/analog co-designs, and formal analysis, verification, and synthesis of cyber physical systems.
Javier Esparza holds the Chair for Foundations of Software Reliability and Theoretical Computer Science at the Technische Universität München since 2007. Previously he held the Chair of Software Reliability and Security at the University of Stuttgart (2003-2007), the Chair of Theoretical Computer Science at the University of Edinburgh (2001-2003), and worked as Associate Professor at the Technische Universität München (1994-2001). He has co-authored a book on Free Choice Petri nets with Jörg Desel, and a book on the unfolding approach to Model Checking with Keijo Heljanko. He has published over 150 scientific papers in the fields of automatic program verification, program analysis, concurrency theory, and automata theory.

Javier Esparza has contributed to the theory Petri nets, and was one of the initiators of the unfolding approach to model checking, the automata-theoretic approach to software model checking, and the verification of infinite-state systems. More recently he has conducted research on the fundamentals of program analysis and the verification of parametrized and stochastic systems. His group has developed several verification tools, including Moped and jMoped and Rabinizer. Javier Esparza received a honorary doctorate in Informatics from the Masaryk University of Brno in 2009 and is member of Academia Europaea since 2011.
Tevfik Bultan

Tevfik Bultan is a Professor in the Department of Computer Science at the University of California, Santa Barbara (UCSB). His research interests are in software verification, static analysis, software engineering, and computer security. He co-chaired the program committees of the 9th International Symposium on Automated Technology for Verification and Analysis (ATVA 2011), the 20th International Symposium on the Foundations of Software Engineering (FSE 2012) which is the flagship conference of ACM SIGSOFT, and the 28th IEEE/ACM International Conference on Automated Software Engineering (ASE 2013). He has served as the vice chair of the Department of Computer Science at UCSB from 2005 to 2009.

Tevfik Bultan received a NATO Science Fellowship from the Scientific and Technical Research Council of Turkey (TUBITAK) in 1993, a Regents’ Junior Faculty Fellowship from the University of California, Santa Barbara in 1999, a Faculty Early Career Development (CAREER) Award from the National Science Foundation in 2000, the ACM SIGSOFT Distinguished Paper Award and the Best Paper Award at the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005), the ACM SIGSOFT Distinguished Paper Award at the 29th IEEE/ACM International Conference on Automated Software Engineering (ASE 2014), and the UCSB Academic Senate Outstanding Graduate Mentor Award in 2016.
Lenore Zuck teaches at the University of Illinois at Chicago. She had spent several years as a program director at the National Science Foundation, where she was a member of the Trustworthy Computing program, the Software and Hardware Foundation program, and the Cyber Physical Systems program. Her background is in formal methods. Her recent work includes methodologies for automatic verification of infinite-state systems, translation validation of LLVM, and applications of formal methods to security. Prior to UIC, Lenore Zuck taught at NYU and at Yale University. She holds a PhD in Computer Science from the Weizmann Institute of Science.
Keynotes

Keynotes take place at the conference venue, room “Tenpyo” (天平, 4th floor).

Keynote: Masahiro Fujita

Unification of Synthesis and Verification in Topologically Constrained Logic Design

In logic synthesis, the search space is infinite in the sense that any number of gates can be connected using any topology to come up with the best circuit under performance and other constraints. In formal verification, to achieve full coverage, an n-input circuit must be checked either explicitly or implicitly using the complete set of 2n input values. There are, however, situations when the possible circuit topologies are limited. Logic optimization methods, in general, do not change circuit topologies dramatically. Most of them are based on series of local circuit transformations. In this paper, we discuss formal verification of circuits produced by logic synthesis where the search space is limited, and only gates or subcircuits are transformed whereas their interconnections never change. If there are p possible transformations for each gate or subcircuit, and there are m gates or subcircuits in the entire circuit, the number of all possible transformations for the entire circuit is pm. Logic synthesis with this restriction attempts to find the best circuit among the pm alternatives. The logic synthesis problem can be formulated as a sequence of incremental SAT problems and the complete set of test patterns can be computed, which detects all possible errors in the synthesized circuit. As long as the search space is limited to the pm alternatives, such complete set of test patterns can be used for formal verification. The number of test patterns needed in this case is experimentally shown to be very small, e.g., a few hundred, even for circuits having several hundred inputs and several thousand gates. With the complete set of test patterns generated for the circuit transformations, logic synthesis and formal verification are unified in the sense that the small number of test patterns allows for logic synthesis with 100% correctness.
Keynote: Tevfik Bultan

Side Channel Analysis Using a Model Counting Constraint Solver and Symbolic Execution

A crucial problem in software security is the detection of side-channels. Information gained by observing non-functional properties of program executions (such as execution time or memory usage) can enable attackers to infer secret information (such as a password). In this talk, I will discuss how symbolic execution, combined with a model counting constraint solver, can be used for quantifying side-channel leakage in Java programs. In addition to computing information leakage for a single run of a program, I will also discuss computation of information leakage for multiple runs for a type of side channels called segmented oracles. In segmented oracles, the attacker is able to explore each segment of a secret (for example each character of a password) independently. For segmented oracles, it is possible to compute information leakage for multiple runs using only the path constraints generated from a single run symbolic execution. These results have been implemented as an extension to the symbolic execution tool Symbolic Path Finder (SPF) using the SMT solver Z3 and two model counting constraint solvers LattE and ABC.
Keynote: Javier Esparza

From LTL to Limit-Deterministic Automata

Limit-deterministic Büchi automata can replace deterministic Rabin automata in probabilistic model checking algorithms, and can be significantly smaller. We present a direct construction from an LTL formula to a limit-deterministic Büchi automaton. Our translation is compositional and has a clear logical structure. Moreover, due to its special structure, the resulting automaton can be used not only for qualitative, but also for quantitative verification of Markov Decision Processes, using the same model checking algorithm as for deterministic automata. This allows one to reuse existing efficient implementations of this algorithm without any modification. Our construction yields much smaller automata for formulas with deep nesting of modal operators and performs at least as well as the existing approaches on general formulas.

Joint work with Stefan Jaax, Jan Kretinsky, and Salomon Sickert

Industry session: Akira Mori, AIST

Cyber Physical Software Engineering

The emerging technologies of the Internet of Things (IoT) and cyber-physical systems call for new methods of software development and software maintenance. In this talk, I will summarize the experiences from the humanoid robot project in AIST and explain the types of software analysis needed for making IoT and cyber-physical systems more reliable.
Tutorials

Tutorials take place at Chiba University, Nishi-Chiba Campus, in the Library Building. Detailed directions are shown further up in this booklet, from the Mitsui Garden Hotel and the Keisei Mira Mare Hotel.

Tutorial: Masahiro Fujita

Synthesizing and completely testing hardware based on templates through small numbers of test patterns

Here we first introduce Quantified Boolean Formula (QBF) based approaches to logic synthesis and testing in general including automatic corrections of designs. It is formulated as: If some appropriate values are assigned to what we call programmable variables, the resulting circuits behaves as our intentions for all possible input values, that is, they become the ones whose logic functions are the intended ones. In this paper we only target combinational circuits and sequential circuits which are time-frame expanded by fixed times. The QBF problems are solved by repeatedly applying SAT solvers, not QBF solvers, with incremental additions of new constraints for each iteration which come from counter examples for the SAT problems. The required numbers of iterations until solutions are obtained are experimentally shown to be pretty small (in the order of tens) even if there are hundreds of inputs, regardless of the fact that they have exponentially many value combinations. Then the applications of the proposed methodology to logic synthesis, logic debugging, and automatic test pattern generations (ATPG) for multiple faults are discussed with experimental results. In the case of ATPG, a test pattern is generated for each iteration, and programmable variables can represent complete sets of functional and multiple faults, which are the most general faults models.
Tutorial: Tevfik Bultan

String Analysis for Vulnerability Detection and Repair

The goal of string analysis techniques is to determine the set of values that string expressions can take during program execution. Like many other program analysis problems, it is not possible to precisely determine the set of string values that can reach a program point. However, one can compute over or under-approximations of possible string values. If the approximations are precise enough, they can enable analyzers to demonstrate existence or non-existence of bugs in string manipulating code. Analyzing string manipulating code is of interest to many areas of current research in program analysis and verification since string manipulation is a crucial part of modern software systems. For example string manipulation is extensively used in input validation and sanitization and in dynamic code and query generation.

String analysis has been an active research area in the last decade, resulting in a wide variety of string analysis techniques. In this tutorial, we will discuss automated string analysis techniques, focusing particularly on automata-based static string analysis. Topics we plan to discuss include symbolic string analysis, string constraint solving, symbolic execution using string constraints, automated repair for string manipulating code, and model counting for string constraints.

Tutorial: Lenore Zuck

Uniform verification of parameterized systems

A parametrized system is a parallel composition of some $N>2$ similar processes. Uniform verification attempts to verify such a composition for every value of $N$. This tutorial will describe some of the main methodologies for achieving such a task: finite abstractions, regular model checking, and generalizations from few processes to the many.
Verification of Population Protocols

Population protocols (Angluin et al., PODC 2004) are a formal model of sensor networks consisting of identical mobile devices. When two devices come into the range of each other, they interact and change their states. Computations are infinite sequences of pairwise interactions where the interacting processes are picked by a fair scheduler.

A population protocol is well specified if for every initial configuration C of devices and for every fair computation starting at C, all devices eventually agree on a consensus value that only depends on C. If a protocol is well-specified, then it is said to compute the predicate that assigns to each initial configuration its consensus value. The main two verification problems for population protocols are: Is a given protocol well-specified? Does a given protocol compute a given predicate?

While the class of predicates computable by population protocols was already established in 2007 (Angluin et al., Distributed Computing), the decidability of the verification problems remained open until 2015, when my colleagues and I finally proved it (Esparza et al., CONCUR 2015, improved version to appear in Acta Informatica). In the talk I report on our results and discuss some new developments.

Joint work with Pierre Ganty, Jerome Leroux, and Rupak Majumdar
Conference Program

Day 1: October 17, morning

8:00 – 8:45 Registration opens

8:45 – 9:00 Opening

9:00 – 10:00 Keynote: Masahiro Fujita
Unification of Synthesis and Verification in Topologically Constrained Logic Design

10:00 – 10:30 Coffee break

10:30 – 12:05 Session 1: Complexity, Decidability

J. Brunel, D. Kuperberg and D. Chemouil. On Finite Domains in First-Order Linear Temporal Logic

R. Brenguier and V. Forej. Decidability Results for Multi-objective Stochastic Games

A. Reynolds, R. Iosif, C. Serban and T. King. A Decision Procedure for Separation Logic in SMT

Philipp J. Meyer and M. Luttenberger. Solving Mean-Payoff Games on the GPU (tool paper)

12:05 – 13:20 Lunch
Day 1: October 17, afternoon

13:20 – 15:00 Session 2: Parallelism, Concurrency
Ó. Martín, A. Verdejo and N. Martí-Oliet. Synchronous products of rewrite systems

B. Finkbeiner, H. Seidl and C. Müller. Specifying and Verifying Secrecy in Workflows with Arbitrarily Many Agents

T. Nguyen Lam, B. Fischer, S. L. Torre and G. Parlato. Lazy Sequentialization for the Safety Verification of Unbounded Concurrent Programs

N. Benes, L. Brim, M. Demko, S. Pastva and D. Safránek. Parallel SMT-Based Parameter Synthesis with Application to Piecewise Multi-Affine Systems

15:00 – 15:30 Coffee break

15:30 – 17:00 Session 3: Counter Systems, Automata
R. Iosif and A. Sangnier. How hard is it to verify flat affine counter systems with the finite monoid property?

F. Avellaneda, S. D. Zilio and J. Raclet. Solving Language Equations using Flanked Automata


18:00 – 20:00 Reception: Room Hakuho (白鳯), 4th floor
Day 2: October 18, morning

9:00 – 10:00 Keynote: Tevfik Bultan
Side Channel Analysis Using a Model Counting Constraint Solver and Symbolic Execution

10:00 – 10:30 Coffee break

10:30 – 12:10 Session 4: Markov Models, Chains, and Decision Processes
A. Abate, M. Ceska and M. Kwiatkowska. Approximate Policy Iteration for Markov Decision Processes via Quantitative Adaptive Aggregations

T. Brazdil, A. Kucera and P. Novotný. Optimizing the Expected Mean Payoff in Energy Markov Decision Processes


N. Jansen, C. Dehnert, B. L. Kaminski, J. Katoen and L. Westhofen. Bounded Model Checking for Probabilistic Programs

12:10 – 13:20 Lunch
Day 2: October 18, afternoon

13:20 – 15:00 Session 5: Optimization, Heuristics, Partial-Order Reductions

C. Baier, H. de Meer, S. Klüppelholz, F. Niedermeier and S. Wunderlich. Greener Bits: Formal Analysis of Demand Response


P. Metzler, H. Saissi, P. Bokor, R. Hesse and N. Suri. Efficient Verification of Program Fragments: Eager POR

15:00 – 15:30 Coffee break

15:30 – 16:45 Session 6: Synthesis, Refinement

B. Finkbeiner and H. Torfah. Synthesizing Skeletons for Reactive Systems

S. Ben-David, M. Chechik and S. Uchitel. Observational Refinement and Merge for Disjunctive MTSs

X. Li and N. Kobayashi. Equivalence-Based Abstraction Refinement for muHORS Model Checking

18:00 – 21:00 Banquet: Room Heian-Higashi (平安東), 3rd floor
Day 3: October 19, morning

9:00 – 10:00 Keynote: Javier Esparza
From LTL to Limit-Deterministic Automata

10:00 – 10:30 Coffee break

10:30 – 12:10 Session 7: Solving Procedures, Model Checking

H. Roehm, J. Oehlerking, T. Heinz and M. Althoff. STL Model Checking of Continuous and Hybrid Systems

M. Marescotti, A. Hyvärinen and N. Sharygina. Clause Sharing and Partitioning for Cloud-Based SMT Solving

D. Deininger, R. Dimitrova and R. Majumdar. Symbolic Model Checking for Factored Probabilistic Models

12:10 – 12:20 presentation of ATVA 2017

12:20 – 13:30 Lunch
Day 3: October 19, afternoon

13:30 – 14:30 Industry session: Akira Mori, AIST
Cyber Physical Software Engineering

14:30 – 15:00 Coffee break

15:00 – 16:40 Session 8: Program Analysis

J. Hua and S. Khurshid. A Sketching-Based Approach for Debugging Using Test Cases

S. D. Oliveira, S. Bensalem and V. Prevosto. Polynomial invariants by linear algebra

R. Qiu, C. Pasareanu and S. Khurshid. Certified Symbolic Execution

P. Cadek, J. Strejcek and M. Trťík. Tighter Loop Bound Analysis

16:45 – 16:50 Closing
Day 4: October 20: Tutorial day  
Chiba University, Nishi-Chiba Campus

University Library, Building “I”, Content Studio Hikari

9:00 – 10:30 Tutorial 1: Masahiro Fujita
Synthesizing and completely testing hardware based on templates through small numbers of test patterns

10:30 – 11:00 Break

11:00 – 12:30 Tutorial 2: Tevfik Bultan
String Analysis for Vulnerability Detection and Repair

12:30 – 13:30 Lunch

13:30 – 15:00 Tutorial 3: Lenore Zuck
Uniform verification of parameterized systems

15:00 – 15:30 Break

15:30 – 17:00 Tutorial 4: Javier Esparza
Verification of Population Protocols
Getting around in Chiba and Tokyo

Japanese characters that are good to know

- 千葉  Chiba
- 東京  Tokyo
- 入口  entrance
- 出口  exit
- 一  one
- 二  two
- 三  three
- 肉  meat
- 魚  fish
- 鳥  chicken
- 牛  beef
- 豚  pork
- 女  woman
- 男  man

Eating out

Note: Soup-style dishes are served piping hot; please be careful with your first bite or sip.

- 居酒屋  Izakaya: Japanese-style pub serving many small dishes that are typically shared in a group
- 天ぷら  Tempura: deep-fried seafood and vegetables
- そば うどん  Soba, Udon: Noodles that are served in a savory broth; common side dishes include rice and tempura
- ラーメン  Ramen: Chinese-style noodle soups, typically served with sliced broiled pork
- トンカツ  Tonkatsu: Deep-fried pork cutlets
- 寿司  Sushi