VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kansliets noteringar Kod	Dnr
2014-13681-113265-110	2014-5489

2014
Project Research Grant

Area of science
Natural and Engineering Sciences

Announced grants
Research grants NT April 9, 2014

Total amount for which applied (kSEK)

2015	2016	2017	2018	2019
1602	1665	1766	1622	1727

APPLICANT

Name(Last name, First name)
Abdulla, Parosh

Date of birth
610618-2550

Gender
Male

Email address
parosh@it.uu.se

Academic title
Professor

Position
Professor

Phone
070-425 02 16

Doctoral degree awarded (yyyy-mm-dd)
1990-05-30

WORKING ADDRESS

University/corresponding, Department, Section/Unit, Address, etc.
Uppsala universitet
Institutionen för informationsteknologi
Datorteknik
Box 337
75105 Uppsala, Sweden

ADMINISTRATING ORGANISATION

Administrating Organisation
Uppsala universitet

DESCRIPTIVE DATA

Project title, Swedish (max 200 char)
Verifiering av Parallell Mjukvara

Project title, English (max 200 char)
Verification of Concurrent Software

Abstract (max 1500 char)
Our society is becoming increasingly dependent on complex computer systems. In communication, transportation, household appliances, industrial process control, health care, and many other areas, computers are now indispensable. New computer systems are parallel machines, requiring major changes in software development. Leveraging the full power of multicore processors demands new tools and new thinking from the software industry. Concurrent and parallel programs are particularly hard to write correctly, because of the vast number of possible interactions between concurrently executing activities. Algorithmic verification and model checking techniques guarantee correctness by covering all such interactions, but major advancements are needed to make them applicable to realistic concurrent programs.

The goal of the project is to develop and combine techniques from a wide range of different areas including infinite-state model checking, dynamic and static program analysis, efficient graph searching, and formal modeling. Concretely, we will define new formal models and algorithms for debugging and verifying programs running on weak memory models, develop efficient techniques to handle systems that run large numbers of parallel threads, and propose new methods to handle complicated features that are common in thread behaviors.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-13681-113265-110

Name of Applicant
Abdulla, Parosh

Date of birth
610618-2550

Abstract language
English

Keywords
concurrency, program verification, program analysis, model checking, testing

Review panel
NT-2

Project also includes other research area

Classification codes (SCB) in order of priority
10201, 10206,

Aspects

Continuation grant
Application concerns: New grant
Registration Number:

Application is also submitted to

similar to: **identical to:**

ANIMAL STUDIES

Animal studies
No animal experiments

OTHER CO-WORKER

Name(Last name, First name)
Krishnan, Narayan Kumar

University/corresponding, Department, Section/Unit, Addressetc.
Chennai Mathematical Institute

Date of birth
690223

Gender
Male

Academic title
Professor

Doctoral degree awarded (yyyy-mm-dd)
1997-05-05

Name(Last name, First name)
Cyriac, Aiswarya

University/corresponding, Department, Section/Unit, Addressetc.
Uppsala universitet
Institutionen för informationsteknologi

Date of birth
860904-7660

Gender
Female

Academic title
PhD

Doctoral degree awarded (yyyy-mm-dd)
2014-01-28

Name(Last name, First name)
Manjunatha, Praveen

University/corresponding, Department, Section/Unit, Addressetc.
Chennai Mathematical Institute

Date of birth
800526

Gender
Male

Academic title
PhD

Doctoral degree awarded (yyyy-mm-dd)
2011-11-16

Name(Last name, First name)
,

University/corresponding, Department, Section/Unit, Addressetc.

Date of birth

Gender

Academic title

Doctoral degree awarded (yyyy-mm-dd)

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-13681-113265-110

Name of Applicant
Abdulla, Parosh

Date of birth
610618-2550

ENCLOSED APPENDICES

A, B, B, B, B, C, C, C, C, N, d1, S

APPLIED FUNDING: THIS APPLICATION

Funding period (planned start and end date)
2015-01-01 -- 2019-12-31

Staff/ salaries (kSEK)

Main applicant	**% of full time in the project**	**2015**	**2016**	**2017**	**2018**	**2019**
Parosh Abdulla	30					
Other staff						
PostDoc	25	212	225	240	252	275
PhD Student	80	495	520	563	610	651
PhD student	80	495	520	563	610	651
Total, salaries (kSEK):		1202	1265	1366	1472	1577

Other project related costs (kSEK)	**2015**	**2016**	**2017**	**2018**	**2019**
Collaboration with India	250	250	250		
Premises (office rooms for hired staff	40	40	40	40	40
Travel Costs	80	80	80	80	80
Other costs	30	30	30	30	30
Total, other costs (kSEK):	400	400	400	150	150

Total amount for which applied (kSEK)

2015	2016	2017	2018	2019
1602	1665	1766	1622	1727

ALL FUNDING

Other VR-projects (granted and applied) by the applicant and co-workers, if applic. (kSEK)

Project title	**Applicant**	**Funded 2014**	**Funded 2015**	**Applied 2015**
Automata, Probabilities, and Games	Parosh Aziz Abdulla	1000		

Funds received by the applicant from other funding sources, incl ALF-grant (kSEK)

POPULAR SCIENCE DESCRIPTION

Popularscience heading and description (max 4500 char)

Parallella program som körs på flera processorer samtidigt eller på flera olika datorer samtidigt blir allt vanligare. Det kan handla om program där en enda processor inte är tillräcklig för att leverera den prestanda som krävs, och där man istället låter flera processorer samarbeta. Alternativt kan det handla om program vars uppgifter är naturligt

parallella, som operativsystem som kör flera applikationsprogram samtidigt eller server-program som hanterar flera klient-program samtidigt över ett nätverk.

Parallellismen tillför nya svårigheter för den som konstruerar programmet. Det är inte längre uppenbart i vilken ordning som olika operationer kommer att ske. Olika parallella delar av ett program kan också konkurrera med varandra om gemensamma och begränsade resurser. Den ökade svårigheten innebär att risken för fel i parallella program är stor. Sådana fel kan vara ödesdigra om programmet utför en säkerhetskritisk uppgift, som t.ex. att kontrollera ett kraftverk eller att kontrollera avståndet mellan två tåg på samma räls.

Den vanligaste metoden för att eliminera fel i program är testning. Programmet körs upprepade gånger i olika situationer och resultatet kontrolleras. Eftersom antalet möjliga situationer som ett program kan köras i oftast är stort eller oändligt så kan man normalt inte testa ett program fullständigt. Därför kan testning inte ge en fullständig garanti för att programmet är korrekt. Än värre är situationen när det gäller parallella program. Eftersom ordningen mellan olika operationer i ett parallellt program inte är förutbestämd kan ett parallellt program vid olika tillfällen ge olika resultat även om situationen är densamma. Det gör att fel i parallella program ofta är oförutsägbara och mycket svåra att hitta med hjälp av testning.

För program vars korrekthet är kritisk, är det nödvändigt att använda säkrare metoder. Sådana metoder kallas för "formella metoder". De syftar till att garantera ett programs korrekthet genom att producera ett matematiskt bevis för den.

Beteendet hos ett enskilt program kan beskrivas som ett system av tillstånd och övergångar. Tillstånden beskriver programmets nuvarande situation; vilka värden som olika parametrar för tillfället antar. Övergångarna beskriver på vilket sätt programmet kan byta tillstånd.

Ett parallellt program, som består av flera mindre delprogram, kan också beskrivas som ett system av tillstånd och övergångar. Vi låter varje tillstånd bestå av all delad information, samt en uppsättning deltillstånd; ett för varje delprogram. Det parallella programmet kan byta tillstånd genom att låta ett av delprogrammen byta sitt deltillstånd genom att följa en övergång i sitt delsystem. På så vis fångas det parallella beteendet att delprogrammen körs oberoende av varandra.

Har vi till exempel två delprogram som beskrivs av system med tillstånden A, B, C respektive D, E, så kan de kombineras till ett parallellt system med tillstånden AD, AE, BD, BE, CD och CE. Om A kan övergå i B i det första delsystemet, så kan AD övergå i BD, och AE kan övergå i BE i det parallella systemet.

Ett problem man stöter på med den metod som vi beskriver ovan är att antalet tillstånd för större program snabbt växer sig ohanterligt stort. Hur man minimerar antalet tillstånd som måste undersökas vid verifiering, liksom metoder för att verifiera system med ett oändligt antal tillstånd, är mycket aktiva forskningsområden.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-13681-113265-110

Name of applicant
Abdulla, Parosh

Date of birth
610618-2550

Title of research programme
Verification of Concurrent Software

Appendix A

Research programme

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Verification of Concurrent Software

Parosh Aziz Abdulla
Department of Information Technology
Uppsala University

Abstract

Leveraging concurrency effectively has become key to enhancing the performance of software, to the degree that concurrent programs have become crucial parts of industrial applications. At the same time, concurrency gives rise to enormously complicated implementations, making the task of producing error-free systems more and more difficult. Therefore, methods for correctness analysis and bug detection in concurrent systems have gained enormous importance, to the degree that developing such methods is generally considered as one of the grand challenges in computer science research.

The goal of this project is to design fundamental methods and tools that extend the state-of-the-art in the verification of concurrent programs. To that end, we will define formal models that explain the behavior of programs when run on modern (multi-core) architectures, develop efficient algorithms for debugging and verification, and carry out a substantial implementation effort to produce verification tools that we will run on real programs and architectures. We will rely on our experience and on the extended collaboration network we have established with leading research groups in the areas of formal methods, program verification, and model checking.

1 Background

Concurrent programs are notoriously difficult to get right. This is as true today as it was three decades ago. The difference is that, three decades ago, concurrent programs would be found only in the innards of operating systems, and these were built by specialists. Nowadays, concurrent programs are everywhere and they are often built by relatively inexperienced programmers. There are several reasons why the verification of concurrent programs is a such difficult task.

The first source of complexity is due to the large numbers of threads that are generated, and the complicated interaction patterns among these threads. This is a consequence of the desire to exploit the increasing level of concurrency available in modern multicore machines, and the fact that modern programming languages, such as SCALA, ERLANG, and GO contain powerful concurrency mechanisms that sometimes result in thousands or even millions of threads in typical programs written in these languages. In many cases, it can even be difficult to predict the number of threads that may participate in a given run of the system. For instance, in the specification of a mutual exclusion protocol, there is no bound on the number of processes that participate in a given session of the protocol. Also, in concurrency libraries, data structures may be accessed concurrently by an arbitrary number of threads. In fact, it is well-known that large numbers of threads lead quickly to state space explosion, which is the main bottleneck in the applicability of program verification in practice.

Another source of complexity is intricate interactions between the threads and the memory. Such interactions arise due to the *Out-of-Order Execution (OoOE)* paradigm that lets the scheduling of CPU instructions be governed by the availability of their operands rather than by the order in which they are issued [47]. OoOE leads to an efficient use of clock cycles and hence an improvement in program execution times. The gain in efficiency has meant that the OoOE technology is present in most modern processor architectures. In the context of *sequential* programming, the technique is transparent to the programmer since one can still work under the Sequential Consistency (SC) model [39] in which the program behaves according to the classical interleaving semantics. However, this is not true any more once we consider concurrent processes that share the memory. In fact, several algorithms that are designed

for the synchronization of concurrent processes, such as mutual exclusion and producer-consumer protocols, are not correct in the OoOE setting. The inadequacy of the interleaving semantics in the presence of OoOE has prompted researchers to introduce *weak (or relaxed) memory models* by allowing permutations between certain types of memory operations [17, 31, 18]. Weak memory models are used in all major CPU designs including Intel x86 [37, 46], SPARC [49], and PowerPC [36]. Since the weak memory semantics add more behavior to a program, the program may violate its specification even if it runs correctly under the SC semantics.

In addition to interactions among the threads themselves, and between them and the memory, the behavior of a concurrent program may further be complicated by features in the program code of the threads. Classical tools for automatic verification have been restricted either to finite-state programs, or models that are equipped with a "limited degree of infiniteness", typically a *single* unbounded data structure such as clocks [19], stacks [24], or lossy channels [14]. However, realistic models of software include several features that make them infinite in multiple dimensions. For instance, the software inside the server in a sensor network, or a component in a distributed system, may have unbounded data structures, dynamic memory allocations, timing constraints, and recursion, all at the same time. Therefore, there is an urgent need to extend automatic verification methods (such as model checking) by developing advanced verification algorithms and powerful abstraction mechanisms that allow applicability to systems with multiply infinite state spaces.

2 Goals

This proposal presents a five-year plan for a concentrated research effort to develop scalable techniques for efficient bug detection and verification of *parallel programs*, capable of handling the challenges described in the previous section. Our research will be based on developing and combining techniques from a wide range of different areas including infinite-state model checking, dynamic and static program analysis, efficient graph searching, and formal modelling. The algorithmic verification group in Uppsala has a substantial infrastructure and is conducting world leading research in the above areas. Through our vast experience and our wide collaboration network, we believe strongly that we will make a considerable push in the state-of-the-art. Concretely, we will develop efficient techniques to handle systems that run large numbers of parallel threads, define new formal models and algorithms for debugging and verifying programs running on weak memory models, and propose new methods to handle complicated features that are common in thread behaviors. To achieve our goals, we will conduct research in the following directions:

- **Formal Models and Semantics.** We will develop formal models that reflect the behaviors of programs running on modern multicore architectures. Such behaviors may deviate substantially from those under classical sequentially consistent machines, making existing definitions of program semantics inadequate. We will furthermore define new criteria for program correctness that are appropriate under the new semantics.

- **Powerful Verification Algorithms.** A central part of all software verification techniques is to reason efficiently about large/infinite state spaces. Such state spaces arise due to unbounded data structures inside the threads, unbounded numbers of simultaneously executing threads, and complicated interactions with the memory. Our goal is to extend the applicability of automatic verification by developing new efficient symbolic algorithms, defining powerful abstraction techniques, and designing SMT solving methods for new theories common in software systems.

- **Combining Dynamic and Static Analysis.** In order to develop methods which are both precise and scalable, we will combine the precision obtainable by dynamic analysis techniques (such as testing) with the coverage obtainable by static analysis techniques (such as model checking). Such combinations will be crucial for efficient bug detection since, on the one hand they allow more coverage of the state space than testing, and on the other hand they are more scalable than model checking.

- **Tool Support.** We will augment our research with a substantial effort to build tools for program verification and debugging, and carry out a large set of non-trivial case studies.

3 Planned Work

We give examples of a number of tasks that we plan to consider to fulfill the goals of the project. For each task, we will give a few key references that represent the state of the art, and also describe preliminary results that we have already achieved in our work.

3.1 Thread Interactions

We will consider the challenges that arise due to dynamic thread generation and interaction in concurrent software.

Testing is still the predominant technique for the detection of errors in software. Unfortunately, existing testing methods are inadequate to detect concurrency errors. Concurrent programs usually generate large sets of threads, with huge numbers of possible interleavings, and with complex interaction patterns. Unexpected interference among threads often results in Heisenbugs that are extremely difficult to reproduce and eliminate. Extensive efforts have been devoted to address this problem both in the design of programming languages, and in the development of testing technology. *Model checking* [27, 45] addresses the problem by systematically exploring the state space of a given program and verifying that each reachable state satisfies a given property. Applying model checking to realistic programs is problematic, however, since it requires to capture and store a large number of global states.

We will develop several efficient methods that combine the advantages of testing and model checking (scalability and precision). More precisely, by combining techniques such as stateless model checking and partial order techniques we will be able to carry out testing in a systemic manner that allows covering much more of the state space compared to classical techniques. Furthermore, by directing the searching of the state space to consider *only* a subset of the interleavings (those that are more likely to contain bugs), we will be able to substantially increase scalability compared to model checking. Finally, we will augment state space searching with dynamic *cut-off* detection that may report correctness through inspecting only small instances of the system. This yields a powerful technique that efficiently combines testing with program verification.

Bounded Search. We will develop techniques for efficient bug detection in concurrent programs by defining *search policies* that bound the set of program runs considered by the verification procedure, so that we only keep those interleavings that will most likely lead to error states. One useful policy that has recently been proposed for multi-threaded programs with shared variables, is that of *context-bounding* [44, 40]. The idea is to only consider runs performing at most some fixed number of context switches between processes. This provides a trade-off between computational complexity and verification coverage: on the one hand, context-bounded verification can be more efficient than unbounded verification; and on the other hand, many concurrency errors, such as data races and atomicity violations, are manifested in executions with few context switches [40]. In this project, we will define a uniform framework where we will consider different policies for bounding executions in the verification of concurrent programs. Since the size of the problem grows exponentially with the given bound, it is crucial to identify policies that are suitable for different classes of programs. One important contribution of the project will be to provide translation schema that transform the correctness of a concurrent program under a given policy, either to the correctness of a finite-state sequential program, or to the satisfiability of a quantifier-free Presburger formula. This opens the way to leveraging the full power of state-of-the-art finite-state model checkers, and SMT-solvers for obtaining efficient solutions to different bounded versions of reachability problem for concurrent programs. We have taken a first step in that direction by providing an algorithm that translates a bounded variant of the reachability problem for processes communicating through unbounded channels to Presburger formulas [3], resulting in the MPASS tool (`https://github.com/vigenere92/MPass`), and also an algorithm that translates a bounded variant of the reachability problem for concurrent recursive programs to the reachability problem for sequential programs [8].

Stateless Model Checking. *Stateless model checking* [35] is a technique for systematically testing complex real-world software, that avoids state space explosion by exploring the state space of the program without explicitly storing global states. A special run-time scheduler drives the program execution, making decisions on scheduling whenever such decisions may af-

fect the interaction between processes. Stateless model checking has been successfully implemented in tools, such as VeriSoft [34] and CHESS [41], and applied to the verification of realistic concurrent programs [35].

While stateless model checking is applicable to realistic programs, it still suffers from combinatorial explosion, as the number of possible interleavings grows exponentially with the length of program execution. *Partial Order Reduction* (POR) [48, 43, 28, 33] limits the number of explored interleavings, and provides full coverage of all behaviours that can occur in *any* interleaving, even though it explores only a representative subset. This is sufficient for checking most interesting safety properties, including race freedom, absence of global deadlocks, and absence of assertion violations [48, 43, 33]. The generation of the appropriate set of interleavings is based on information about possible future conflicts between threads. Early approaches analyzed such conflicts statically, leading to over-approximations and therefore limiting the achievable reduction. *Dynamic Partial Order Reduction* (DPOR) [32] improves the precision by recording actually occurring conflicts during the exploration and using this information to construct representative sets on-the-fly, "by need" [35, 32, 41].

The work of this project considers DPOR, and is based on the fact that the obtained reduction can vary significantly depending on the order in which processes are explored at each point of scheduling [38]. We will develop techniques in order to minimize the set of interleavings generated in the analysis. We have already presented such techniques that are possible to implement efficiently and that are applicable to ERLANG programs [1], in the CONQUERROR tool (`http://concuerror.com`). Furthermore, we intend to combine DPOR with analysis methods based on bounding criteria (see the above paragraph). More precisely, even for a given bounding criterion, the number of possible interleavings may grow rapidly as the number of processes increases in the executions of the program. An interesting challenge then is to find minimal sets of interleavings that are sufficient to fully analyze the behaviors of the program for a given bound.

We also intend to use the number of variables to further reduce the set of explored interleavings by DPOR. This is based on the observation that in practice, most concurrency bugs can be discovered by restricting the search to only a few number of variables [22]. The idea is to first apply partial order methods to the program while keeping a small subset of its variables; then refine the explored representative subset of interleavings by taking into account new variables of the program.

Cut-Offs. There is strong empirical evidence that concurrent programs often enjoy a *small model property* in the sense that analyzing only a small number of threads is sufficient to capture the reachability of bad configurations. On the one hand, bad configurations can often be characterized by minimal conditions that can be specified through a fixed number of *witness* processes. For instance, in a mutual exclusion protocol, a bad configuration contains *two* processes in their critical sections; and in a cache coherence protocol, a bad configuration contains *two* cache lines in their *exclusive* states. In both cases, having the two witnesses is sufficient to make the configuration bad (regardless of the actual size of the configuration). On the other hand, it is usually the case that such bad patterns (if present) appear already in small instances of the system. We will develop methods that exploits the small model property, and perform verification by only inspecting a small set of *fixed* instances of the system. We will use abstract interpretation techniques to develop frameworks that can be applied uniformly to combine testing with fully automatic verification algorithms for a wide range of systems including distributed algorithms, concurrent data structures, message passing programs, and weak memory models. For instance, for many classes of message passing programs, we believe that correctness can be established by analyzing the program only for small sizes of the buffers between the communicating processes. We have obtained early promising results, establishing small model properties for mutual exclusion protocols and cache coherence protocols [12], and for libraries in concurrent programs [13].

3.2 Weak Memory Models

Modern processor architectures use memory models in which the scheduling of CPU instructions is governed by the availability of their operands rather than by the order in which they are issued [47]. This allows an efficient use of clock cycles and hence an improvement in program execution times. However, this comes at a price. More precisely, the program may exhibit behaviors that are not intended by the programmer. The reason is that weak memory semantics add more behaviors to a program, often causing the program to deviate

substantially from its behavior under the standard *Sequentially Consistent (SC)* semantics (aka interleaving semantics). The standard way to eliminate the undesired behaviors is to insert memory *fence* instructions in the program code. A fence instruction, executed by a process, typically implies that no reordering is allowed between instructions issued before and after the fence instruction.

We consider one of the most important current problems in concurrent programming, namely that of finding sets of fences that guarantee program correctness without compromising its efficiency. Manual fence placement is time-consuming and error-prone due to the complex behaviors of multithreaded programs running on weak memory models. A crucial task then is to develop methods for *automatic* fence placement. This would allow the programmer to concentrate on the algorithmic aspects of the design independently from the underlying architecture, and to achieve rapid prototyping of new algorithms.

In this project, we will consider the "two-dimensional complexity" involved with automatic fence insertion. More precisely, we will deal with (i) The well-known *state space explosion* problem for concurrent programs, that occurs even under the SC semantics since the size of the state space grows exponentially with the number of processes (threads). (ii) The additional complexity due to intricate reorderings of program events, reflected in the fact that standard operational definitions for weak memory models [42, 46] often use *unbounded store-buffer* semantics, thus giving rise to an infinite state space even in the case where the original program is finite-state. Furthermore, we will consider an important aspect in the design of fence insertion algorithms, namely the trade-off between having efficient algorithms and the requirement that we should derive sets of fences that are as close to optimal as possible. We would like to avoid *under-fencing*, i.e., inserting too few fences since this would result in unsound program behaviors; and avoid *over-fencing*, i.e., inserting too many fences since this would result in a performance degradation as it would get us closer to the SC model.

We will develop methods and tools for automatic fence insertion in programs running under different memory models, that are both efficient for given programs with given numbers of threads, and that are scalable as the number of threads is increased. Concretely, we we will perform the following tasks in the project.

Semantics. We will develop the formal semantics of programs running on different memory models. We have already contributed [5, 4, 6] by defining new operational semantics for TSO (Total Store Ordering), which is one the most common models, and corresponds to that is adopted by Sun's SPARC multiprocessors [49]. However, there is still an urgent need for defining formal semantics for other common memory models. Examples include the ARM architecture which is currently found in hundreds of millions of smartphones and tablets, and is increasingly popular in multicore configurations, and IBM POWER multiprocessors which the Xbox 360 has already delivered to 70 million living rooms in a multicore configuration. We intend to define operational (buffer-based) semantics that are particularly suited to program verification. For each case, we will extract the computation models that arise, establish the semantics of programs running on the model, and investigate the decidability and complexity boundaries for the corresponding verification problems.

Stability. We will define different notions of program correctness. More precisely, we will define a hierarchy of 'stability" conditions that will measure, for each weak memory model, how stable the behavior of the program is when it is run under the model compared to when it is run under the SC semantics. Each stability condition corresponds to relaxing some of three (fixed) parameters: program order (the order in which instructions are executed within the same thread), read-from (the read operation from which a write operation fetches its value), and the store ordering (the order in which different write operations hit the shared memory). Identifying "good stability conditions" is crucial due to the important trade-off between the complexity of the fence insertion algorithm and the number of fences that will be inserted by the algorithm. For instance, under the classical data race freedom (DRF) condition, a program is declared incorrect in case it contains a trace with a data race. While this condition can be checked efficiently, it will cause severe over-fencing, since it amounts to the naive approach where we insert a fence instruction after every write instruction, or before every read instruction. This would mean that we get back to the SC model. In fact, many data races are in reality not harmful. For instance, lock-free data structures, transactional memories, and synchronization libraries sometimes explicitly employ data races although they rely on the SC semantics for correctness. On the other extreme,

we consider a strong correctness criterion in [5], where we check whether a given state of the program is reachable under the weak memory semantics but not under the SC semantics. Although, the method allows to derive an optimal set of fences, the correctness problem has a non-primitive recursive complexity, and hence the algorithm does not scale to large programs. We intend to define a hierarchy of stability criteria, and study their behaviors under different memory models. For each criterion, we will establish the complexity of the problem of checking whether a given program has a stable behavior. We will develop efficient algorithms and abstraction techniques for checking stability and integrate them in our MEMORAX tool (see below).

Verification. In order to carry out automatic fence insertion, we need to be able to verify *automatically* the correctness of the program for a given set of fences. This is necessary in order to be able to decide whether the current set of fences is sufficient, or whether additional fences are needed to ensure correctness. We intend to develop verification algorithms for checking stability conditions for programs running on different memory models. To that end, we need to design techniques that can handle infinite-state models, with unbounded data structures such as graphs, trees, and buffers, that arise in the modeling of these programs. In particular, we will provide code-to-code transformations in order to translate the verification of a program running on weak memory models to the verification of a new (synthesized) program that can be analyzed under the SC semantics. This allows the leveraging of existing tools for the latter, leading to very efficient fence insertion algorithms for the source program. We will characterize stability conditions that allow thread abstraction techniques, thus achieving compositional reasoning and significantly limiting the state space explosion problem. This will make the method scalable to programs with large numbers of threads. Finally, we will integrate our methods in the tool MEMORAX [6] (`https://github.com/memorax/memorax`) that we have developed for the verification of programs running on weak memory models. We will use MEMORAX as a platform to evaluate our framework on a wide range of benchmarks and real world examples.

3.3 Complicated Features

We will consider infinite-state models that arise in the verification of software systems. To that end, we will develop new algorithms and abstractions, and enhance the capabilities of SMT solvers.

Infinite-State Systems. One main limitation of most model checking methods is that they can handle only program models that are finite-state. However, finding faithful finite-state models is difficult since many important features are hard to encode in a finite way, e.g. timing constraints, buffers, stack contents, counter values, etc. This means that the abstract model will not be sufficiently accurate to give conclusive results. In fact, faithful models need to be infinite in *several* dimensions. For instance, for a multi-threaded program, we may have to reason about an unbounded number of threads each of which operating on variables over infinite domains. Furthermore, such threads may be communicating over unbounded communication media, and they may be operating under timed constraints. In contrast, almost all available verification algorithms are designed to deal either with finite-state models, or models with a *single* aspect of infiniteness, e.g. unbounded stacks in push-down automata [24], real-valued clocks in timed automata [19], unbounded queues in lossy channel systems [14], etc. We intend to work on new model checking algorithms which can be used on systems which are infinite in several dimensions. We have recently experimented with some classes of such models, e.g., timed Petri nets (which contain an unbounded number of real-values tokens) [15], timed pushdown automata [9] (where the stack carries real-valued clocks), timed lossy channels [2] (where the messages carry time stamps), and infinite-state energy games [16, 7] (where classical models such as Petri nets and push-down automata are extended by unbounded *energy* counters). The research in this track will also be relevant for dealing with thread interaction where infinite state spaces are induced by unbounded numbers of threads (Section 3.1), and for dealing with weak memory models where infinite state spaces are induced by unbounded buffers in the operational semantics (Section 3.2).

SMT Solving. One of the main reasons for the success of model checking is that such tools make use of the latest developments in SMT technology [26, 30, 25]

in order to reason about symbolic representations of different data types in programs. On the other hand, this dependence also means that model checking tools are inherently limited by the data types that can be handled by the underlying SMT solver. Most SMT solvers are currently limited to the theories of integers and arrays. Let us take as example a data type for which satisfying decision procedures have been missing, namely that of *Strings*. String data types are present in all conventional programming and scripting languages. In fact, it is impossible to capture the essence of many programs, for instance in database and web applications, without the ability to precisely represent and reason about string data types. The control flow of programs can depend on the words denoted by the string variables, on their lengths, or on the regular languages to which they belong. For example, a program allowing users to choose a username and a password may require the password to be of a minimal length, to be different from the username, and to be free from invalid characters. Reasoning about such constraints is also crucial when verifying that database and web applications are free from SQL injections and other security vulnerabilities. We intend to design decision procedures for new theories such as strings, trees, and graphs, that specifically target model checking applications. In particular, we will develop methods for integrating of the different theories, explore different Craig interpolation techniques, and explore the applicability of our framework to general classes of programs. We have recently designed the NORN tool (`http://user.it.uu.se/~jarst116/norn/`) for efficient solving of word equations with applications for the verification of string manipulating programs.

4 Past Achievements

The algorithmic verification group conducts world-leading research on algorithmic program verification and model checking. We have made fundamental contributions that address challenges introduced by program verification in general and the verification of concurrent software in particular. Below we present some recent highlights, including works that represent first steps in research directions mentioned in this proposal.

- All contemporary multicores implement relaxed memory models that further complicate the interaction between threads, and many groups worldwide are working to develop corresponding verification methods. We have solved the problem of finding a sound and complete method for verifying programs running on the well-known TSO memory model. This includes deep theoretical results on the operational semantics of the model [5], as well as the first sound and complete verification tool for programs with unbounded data domains [4, 6].

- Many parallel algorithms in applications such as run-time systems, bus and network protocols, cache coherence protocols, web services, sensor networks, etc., are designed for an unbounded number of threads, and their verification have remained a difficult research challenge for almost three decades. We have recently [12], achieved an important breakthrough by developing a "small model theorem" that reduces the verification of programs with *unboundedly many* threads to the verification of programs with a *finite* number of threads, thus allowing the use of existing tools for the latter. A particularly interesting application has been the verification of algorithms that concurrently access a dynamically allocated shared state. Such algorithms are widely used in libraries for concurrent programming. In [13] (ETAPS Best Paper Award), we described for the first time how to carry out fully automatic verification on such algorithms.

- The classical application area of model checking has been systems with *finite state spaces* such as hardware circuits. Our group has been internationally leading in the effort to extend the applicability of model checking to software systems with features, such as unbounded data domains, real-time, and asynchronous communication, that induce *infinite* state spaces. Our recent work [15, 9, 2, 7, 16] has opened a new area by presenting the first positive results for systems with "multiply infinite-state spaces", such as priced timed Petri nets, recursive programs with timed constraints, message passing programs with time stamps, or infinite-state energy games.

- We have developed several techniques for the verification of concurrent recursive programs. For instance, in [8], we have developed the first tool for the verification of safety and liveness properties for such programs where we allow an un-

bounded number of possible context-switches between threads. These techniques are based on systematic code-to-code translation from concurrent programs to sequential programs, and thus allow leveraging highly-developed sequential program analysis tools.

- By combining techniques from model checking, programming languages and a tool infrastructure developed in Uppsala around ERLANG, we have been able for the first time to achieve an optimal reduction in the number of interleaving sequences needed to exhaustively test concurrent programs and verify absence of concurrency-related errors in them [1]. This has also lead to a very efficient implementation of the CONCUERROR tool, a stateless model checker for ERLANG.

- We have made important contributions to a new research direction aiming at efficiently solving classical problems for finite automata such as language inclusion, minimization, and bisimulation [11, 10] (ETAPS Best Paper Award). The work shows how to combine known techniques such as simulation and subset construction to design new algorithms that vastly outperformed state-of-the-art methods. This research effort is of wide relevance since it solves problems on automata, which are fundamental in many areas of computer science, and play a crucial role in many applications, e.g., in the modeling of programs.

- We have recently carried out a substantial implementation effort resulting in the MEMORAX tool for the verification of programs running on weak memory models (`https://github.com/memorax/memorax`), the MPASS tool for the verification of message passing programs (`https://github.com/vigenere92/MPass`), the NORN tool for the verification of string manipulating programs (`http://user.it.uu.se/~jarst116/norn/`), and integrating partial order methods in the CONQUERROR tool (`http://concuerror.com`) for testing ERLANG programs.

Our work on verification has been widely recognized internationally. During the last eight years, we have published more than 80 papers, and have received best-paper-awards at top conferences in the area, including CIAA 2006 and CIAA 2008 (our algorithms for bisimulation minimization for tree automata), ETAPS 2010 (our algorithms for checking language inclusion on finite automata), and ETAPS 2013 (our work on shape analysis for concurrent data structures).

The algorithmic verification group is one the main participants in the VR Linnaeus center *The Uppsala Programming for Multicore Architectures Research Centre (UPMARC)* at Uppsala University. UPMARC is a long-term interdisciplinary research program in parallel computer systems. During the recent (January 2014) mid-term (5 years) evaluation of the center, the international evaluation committee classified our research as world-leading:

"The UPMARC verification group conducts world-leading research on algorithmic program verification and model checking. Since the start of UPMARC, fundamental results on this subject have been obtained, largely thanks to the contribution of one senior and one junior researcher, recently recruited. Some results on this subject earned the group the TACAS 2013 Best Paper Award."

5 Collaboration

In this project, we apply for extra funding to support collaboration between the algorithmic verification group in Uppsala and the Chennai Mathematical Institute (CMI), the team of K Narayan Kumar. Below, we describe the collaboration efforts of both groups with international partners, and current collaboration between the two groups.

Uppsala. We are cooperating with several world-leading groups in the area of the project. The groups with whom we have joint publications in the last eight years include: Ahmed Bouajjani (Paris 7), Yu-Fang Chen (Academia Sinica, Taiwan), Luca de Alfaro (University of California, Santa Cruz), Giorgio Delzanno (Genoa), Richard Mayr (Edinburgh), Joel Ouaknine (Oxford), and Tomas Vojnar (Brno), James Worrell (Oxford).

Chennai. K Narayan Kumar has collaborated extensively with Paul Gastin and Benedikt Bollig (at ENS Cachan) and Ahmed Bouajjani (LIAFA, Univ Paris 7)

for many years now. M Praveen was a post-doctoral fellow at LSV, ENS Cachan and LaBRI, Univ. Bordeaux, resulting in several joint publications.

Uppsala-Chennai. The algorithmic verification group at Uppsala University and group of K Narayan Kumar at the Chennai Mathematical Institute have collaborated in the recent years in addressing a number of problems in the verification of infinite state systems. K Narayan Kumar and Prakash Saivasan, a PhD student at CMI, visited Uppsala Univ. in 2012 and 2013 while M.F.Atig visited CMI in 2012 and 2014. The collaborative work between the two groups has successfully addressed problems in the analysis of multi-pushdown systems [20, 21] as well as energy games [7]. Aiswarya Cyriac, a member of the Algorithmic Program Verification group at Uppsala University, has worked with K Narayan Kumar and Paul Gastin (ENS Cachan) in developing general techniques for analysis of infinite-state systems [23, 29].

6 Strategic Importance

Our society is becoming increasingly dependent on complex computer systems. In communication, transportation, household appliances, industrial process control, health care, and many other areas, computers are now indispensable. New computer systems are *parallel machines*, requiring major changes in software development. Leveraging the full power of multicore processors demands new tools and new thinking from the software industry. Concurrent and parallel programs are particularly hard to write correctly, because of the vast number of possible interactions between concurrently executing activities. Algorithmic verification and model checking techniques guarantee correctness by covering all such interactions, but major advancements are needed to make them applicable to realistic concurrent programs.

Swedish industry has a long tradition of building large complex computer systems, such as telephone exchanges, cars, industrial robots, and power grids. More software components in such systems are becoming parallel. Software often accounts for the largest portion of the development cost for such systems. However, it is not the cost of the code-writing that dominates, but rather the cost of design, debugging, verification, and testing, in combination with maintenance, which typically accounts for more than 50% of the cost. In several discussions, representatives from major Swedish companies, including ABB and Ericsson, have stated that one of their main costs is the quality assurance and maintenance of their (often very complex) computer systems, and that current technologies to deal with these problems are not adequate. The work of the project concerns both laying strong mathematical foundations, and enabling the development of models and algorithmic techniques with a high potential of applications in the analysis of computer systems.

References

[1] P. A. Abdulla, S. Aronis, B. Jonsson, and K. F. Sagonas. Optimal dynamic partial order reduction. In *POPL*, pages 373–384. ACM, 2014.

[2] P. A. Abdulla, M. F. Atig, and J. Cederberg. Timed lossy channel systems. In *FSTTCS*, volume 18 of *LIPIcs*, pages 374–386, 2012.

[3] P. A. Abdulla, M. F. Atig, and J. Cederberg. Analysis of message passing programs using smt-solvers. In *ATVA*, volume 8172 of *LNCS*, pages 272–286, 2013.

[4] P. A. Abdulla, M. F. Atig, Y.-F. Chen, C. Leonardsson, and A. Rezine. Automatic fence insertion in integer programs via predicate abstraction. In *SAS*, volume 7460 of *LNCS*, pages 164–180. Springer, 2012.

[5] P. A. Abdulla, M. F. Atig, Y.-F. Chen, C. Leonardsson, and A. Rezine. Counter-example guided fence insertion under tso. In *TACAS*, pages 204–219, 2012.

[6] P. A. Abdulla, M. F. Atig, Y.-F. Chen, C. Leonardsson, and A. Rezine. Memorax, a precise and sound tool for automatic fence insertion under tso. In *TACAS*, volume 7795 of *LNCS*, pages 530–536. Springer, 2013.

[7] P. A. Abdulla, M. F. Atig, P. Hofman, R. Mayr, K. N. Kumar, and P. Totzke. Automatic fence insertion in integer programs via predicate abstraction. In *LICS*. ACM, 2014.

[8] P. A. Abdulla, M. F. Atig, O. Rezine, and J. Stenman. Budget-bounded model-checking pushdown systems, 2014. Accepted for publication in J. Formal Mehtods in System Design.

[9] P. A. Abdulla, M. F. Atig, and J. Stenman. Dense-timed pushdown automata. In *LICS*, pages 35–44. IEEE, 2012.

[10] P. A. Abdulla, Y.-F. Chen, L. Clemente, L. Holík, C.-D. Hong, R. Mayr, and T. Vojnar. Advanced ramsey-based büchi automata inclusion testing. In *CONCUR*, volume 6901 of *LNCS*, pages 187–202. Springer, 2011.

[11] P. A. Abdulla, Y.-F. Chen, L. Holík, R. Mayr, and T. Vojnar. When simulation meets antichains. In *TACAS*, volume 6015 of *LNCS*, pages 158–174. Springer, 2010.

[12] P. A. Abdulla, F. Haziza, and L. Holík. All for the price of few (parameterized verification through view abstraction). In *VMCAI*, volume 7737 of *LNCS*, pages 476–495, 2013.

[13] P. A. Abdulla, F. Haziza, L. Holík, B. Jonsson, and A. Rezine. An integrated specification and verification technique for highly concurrent data structures. In *TACAS*, volume 7795 of *LNCS*, pages 324–338, 2013.

[14] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. In *Proc. LICS' 93* 8^{th} *IEEE Int. Symp. on Logic in Computer Science*, pages 160–170, 1993.

[15] P. A. Abdulla and R. Mayr. Computing optimal coverability costs in priced timed petri nets. In *LICS*, pages 399–408. IEEE Computer Society, 2011.

[16] P. A. Abdulla, R. Mayr, A. Sangnier, and J. Sproston. Solving parity games on integer vectors. In *CONCUR*, volume 8052 of *LNCS*, pages 106–120. Springer, 2013.

[17] S. Adve and K. Gharachorloo. Shared memory consistency models: a tutorial. *Computer*, 29(12), 1996.

[18] S. Adve and M. D. Hill. Weak ordering - a new definition. In *ISCA*, 1990.

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

[20] M. Atig, A. Bouajjani, K. N. Kumar, and P. Saivasan. Linear-time model-checking for bounded-scope multithreaded programs. In S. Chakraborty and M. Mukund, editors, *ATVA*, volume 7561, pages 152–166, 2012.

[21] M. F. Atig, K. N. Kumar, and P. Saivasan. Adjacent ordered multi-pushdown systems. In M.-P. Béal and O. Carton, editors, *Developments in Language Theory*, volume 7907, pages 58–69. Springer, 2013.

[22] S. Bindal, S. Bansal, and A. Lal. Variable and thread bounding for systematic testing of multithreaded programs. In *ISSTA*, pages 145–155. ACM, 2013.

[23] B. Bollig, A. Cyriac, P. Gastin, and K. N. Kumar. Model checking languages of data words. In L. Birkedal, editor, *FoSSaCS*, volume 7213, pages 391–405. Springer, 2012.

[24] A. Bouajjani, J. Esparza, and O. Maler. Reachability Analysis of Pushdown Automata: Application to Model Checking. In *Proc. Intern. Conf. on Concurrency Theory (CONCUR'97)*. LNCS 1243, 1997.

[25] A. Brillout, D. Kroening, P. Rümmer, and T. Wahl. An interpolating sequent calculus for quantifier-free Presburger arithmetic. *Journal of Automated Reasoning*, 47:341–367, 2011.

[26] A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani. The MathSAT5 SMT solver. In *TACAS*, pages 93–107, 2013.

[27] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[28] E. M. Clarke, O. Grumberg, M. Minea, and D. Peled. State space reduction using partial order techniques. *STTT*, 2:279–287, 1999.

[29] A. Cyriac, P. Gastin, and K. N. Kumar. Mso decidability of multi-pushdown systems via split-width. In M. Koutny and I. Ulidowski, editors, *CONCUR*, volume 7454 of *LNCS*, pages 547–561. Springer, 2012.

[30] L. M. de Moura and N. Bjørner. Z3: An efficient smt solver. In *TACAS*, pages 337–340, 2008.

[31] M. Dubois, C. Scheurich, and F. A. Briggs. Memory access buffering in multiprocessors. In *ISCA*, 1986.

[32] C. Flanagan and P. Godefroid. Dynamic partial-order reduction for model checking software. In *POPL*, pages 110–121. ACM, 2005.

[33] P. Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem*. PhD thesis, University of Liège, 1996. Also, volume 1032 of LNCS, Springer.

[34] P. Godefroid. Model checking for programming languages using VeriSoft. In *POPL*, pages 174–186. ACM Press, 1997.

[35] P. Godefroid, B. Hammer, and L. Jagadeesan. Model checking without a model: An analysis of the heart-beat monitor of a telephone switch using VeriSoft. In *Proc. ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 124–133, 1998.

[36] IBM, editor. *Power ISA v.2.05*. 2007.

[37] I. Inc. Intel^{TM}64 and IA-32 Architectures Software Developer's Manuals.

[38] K. Kähkönen, O. Saarikivi, and K. Heljanko. Using unfoldings in automated testing of multithreaded programs. In *ASE*, pages 150–159. ACM, 2012.

[39] L. Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE Trans. Comp.*, C-28(9), 1979.

[40] M. Musuvathi and S. Qadeer. Iterative context bounding for systematic testing of multithreaded programs. In *PLDI*, pages 446–455. ACM, 2007.

[41] M. Musuvathi, S. Qadeer, T. Ball, G. Basler, P. Nainar, and I. Neamtiu. Finding and reproducing heisenbugs in concurrent programs. In *OSDI*, pages 267–280. USENIX Association, 2008.

[42] S. Owens, S. Sarkar, and P. Sewell. A better x86 memory model: x86-tso. In *TPHOL*, 2009.

[43] D. Peled. All from one, one for all, on model-checking using representatives. In *CAV*, volume 697 of *LNCS*, pages 409–423, 1993.

[44] S. Qadeer and J. Rehof. Context-bounded model checking of concurrent software. In *TACAS'05*. LNCS 3440, 2005.

[45] J. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *5th International Symposium on Programming, Turin*, volume 137 of *Lecture Notes in Computer Science*, pages 337–352. Springer Verlag, 1982.

[46] P. Sewell, S. Sarkar, S. Owens, F. Z. Nardelli, and M. O. Myreen. x86-tso: A rigorous and usable programmer's model for x86 multiprocessors. *CACM*, 53, 2010.

[47] J. E. Smith and A. R. Pleszkun. Implementation of precise interupts in pipelined processors. In *25 Years ISCA: Retrospectives and Reprints*, 1998.

[48] A. Valmari. Stubborn sets for reduced state space generation. In *Advances in Petri Nets*, volume 483 of *LNCS*, pages 491–515, 1990.

[49] D. Weaver and T. Germond, editors. *The SPARC Architecture Manual Version 9*. PTR Prentice Hall, 1994.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-13681-113265-110

Name of applicant
Abdulla, Parosh

Date of birth
610618-2550

Title of research programme
Verification of Concurrent Software

Appendix B

Curriculum vitae

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Curriculum Vitae – Parosh Aziz Abdulla

Personal Details

Born 1961. Married with two children.

Education

BSc in Electrical Engineering, University of Salahaddin, Arbil, Iraq, 1982. *PhD in Computer Science*, Uppsala University, 1990. *Docent* in Computer Systems, 1997.

Academic Positions

2000–present, Professor, Department of Information Technology, Uppsala University. 1992–2000, Associate Professor (universitetslektor), Department of Computer Systems, Uppsala University. 1990–1992, Lecturer, Department of Computer Systems, Uppsala University.

Visiting Positions

Visiting professor at the University of Paris 7, 2000,2005,2009. Visiting professor at École Normale Supérieure de Cachan, France, 2007. Visiting researcher at the Isaac Newton Institute for Mathematical Sciences, Logic and Algorithms, Cambridge, UK, 2006. Visiting professor at VERIMAG, the University of Grenoble, 1998,1999. Visiting professor at VERIMAG, the University of Grenoble, 1998.

Graduated PhD Students (as main advisor)

Mats Kindahl, graduated 1999. Aletta Nylén, graduated 2003 (on parental leave for one year). Julien d'Orso, graduated 2003. Pritha Mahata, graduated 2005. Johann Deneux, graduated 2006. Sven Sandberg, graduated 2007. Noomene Ben Henda, graduated 2008. Ahmed Rezine, graduated 2008. Lisa Kaati, graduated 2008.

Current PhD Students

Jonathan Cederberg (to graduate 2014), Frédéric Haziza (to graduate 2014), Othmane Rezine, Jari Stenman, and Yunyun Zhu.

Postdocs

Aiswarya Cyriac, postdoc, 2014-2015. Lukas Holik, postdoc, 2011–2012. Yu-Fang Chen, postdoc, 2009–2010. Ahmed Rezine, postdoc 2009–2011. Faouzi Atiq, postdoc, 2010–2011. Gerardo Schneider, postdoc, 2000–2002.

Distinctions and Awards

Best Paper at *ETAPS'2000*, the European Joint Conferences on Theory and Practice of Software. *Best Paper* at *ICALP'2001*, the 28th International Colloquium on Automata, Languages and Programming, *Best Paper* at *CIAA'2006*, the 11th International Conference on Implementation and Application of Automata. *Microsoft Award* for distinguished papers co-authored by a student, at *TACAS'2007*, the 13th International conference on Algorithms and Tools for the Construction and Analysis of Systems. *Best Paper* at *CIAA'2008*, the 13th International Conference on Implementation and Application of Automata. 2008. *Best Paper* at *WMC'2008*, the 9th Workshop on Membrane Computing. *Best Paper* at *ETAPS'2010*, the European Joint Conferences on Theory and Practice of Software. *Best Paper* at *ETAPS'2013*, the European Joint Conferences on Theory and Practice of Software. Receiver of the "Bjurzons prize for outstanding PhD theses" ("Bjurzons premium för förtjänstfull avhandling"), 1990.

Conference Organization, Committee Membership, and Invited Talks– Last 5 Years

Member of editorial board, Journal of Theoretical Computer Science. Guest editor, International Journal of Software Tools and Technology Transfer. Guest editor, Journal of Logical Methods in Computer Science. Steering committee member, ETAPS, 2010-2013. PC member, HIGLIGHT'2014. PC co-chair, RP'2013. PC co-chair, TACAS'2011. Unifying Invited Speaker, CONCUR'2011 and QEST'2011. PC member, ATVA'2011. PC member, RP'2011. PC member, INFINITY'2011. Invited Speaker, SOFSEM'2010. PC member, CONCUR'2006,CONCUR'2010. PC member, LICS'2001,LICS'2010, LICS'2013. PC member, TACAS'2010. PC member, GandALF'2010. PC member, CAV'2007,CAV'2009. Invited Speaker (Tutorial), QEST'2009. PC member, LATA'2009, LATA'2013. PC member, FoSSaCS'2009. PC member, QEST'2007,QEST'2009. Organizer and PC chair, MCC'2009. PC member, ICTAC'2009. Organizer, the UPMARC summer school on Multicore Computing, 2009. PC member, GAMES'2009. PC member, RP'2009,RP'2010,RP'2011. PC member, ICALP'2008. PC member, MEMICS'2008,MEMICS'2009,MEMICS'2010,MEMICS'2011. PC member, MFCS'2007, MFCS'2013. PC member, FORMATS'2006, FORMATS'2012, FORMATS'2014. PC member, INFINITY'2006,INFINITY'2009, INFINITY'2010, INFINITY'2011. Organizer, Dagstuhl seminar on Verification of Dynamically Evolving Graph Structures: Pointer Programs, Networks and Beyond, 2014. Organizer, Dagstuhl seminar on Software Verification: Infinite-State Model Checking and Static, Program Analysis, 2006. Invited Speaker, Constraints and Verification, Isaac Newton Institute Workshop, University of Cambridge, 2006.

Reviewing and Evaluation (Selected)

Evaluator for lectureship position at the Royal Institute of Technology (KTH), Stockholm, 2014. Evaluator for lectureship position at Umeå University, 2013. Research grant reviewer for The French National Research Agency (L'Agence nationale de la recherche), 2011. Research reviewer for the *The Leverhulme Trust*, UK, 2011. Review panel member of the Research Council for Natural Sciences and Engineering of the Academy of Finland, 2010. Reviewer for the European Research Council ERC Advanced Grant, 2010. Research grant reviewer for the Vienna Science and Technology Fund, 2010. Research grant reviewer for the Fundamental Research Programme at the Université Libre de Bruxelles, 2009. Research grant reviewer for the British Council Researcher Exchange Program, 2008. Research grant reviewer for The Netherlands Organisation for Scientific Research (NWO), 2006. Research grant reviewer for The French National Research Agency (L'Agence nationale de la recherche), 2005. Research grant reviewer for The Icelandic Centre for Research, 2005. Research grant reviewer for The United States-Israel Binational Science Foundation, 2003.

Curriculum Vitae

Aiswarya Cyriac

Personal Details

Born: 04 September 1986
Gender: Female
Nationality: Indian
Marital Status: Married

Education

- PhD in Computer Science, January 2014.
    - University:Ecole Normale Superieure de Cachan, France
    - Thesis Title: Verification of Communicating Recursive Programs via Split-width
    - Advisors: Benedikt Bollig and Paul Gastin
- Masters in Computer Science, 2010.
    - University (2nd year): Ecole Normale Superieure de Cachan, France
    - University (1st year): Institute of Mathematical Sciences, Chennai, India
    - Mention: High honors
- Bachelor of Technology in Computer Science, 2008.
    - University: National Institute of Technology, Calicut, India
    - Mention: First class with distinction

Current designation

Post-doctoral Researcher at Uppsala University since March 2014.

Distinctions

- Selected in the International Scholarship Program of ENS Cachan, 2009.
- Received Google India Women in Engineering Award, 2009.

CURRICULUM VITAE

Name: K. Narayan Kumar

Date of birth: 23 February, 1969

Address: Chennai Mathematical Institute,
H1, SIPCOT IT Park, Siruseri 603103, India.
http://www.cmi.ac.in/~kumar
Email: kumar@cmi.ac.in

Research interests:

- Automata, Logics and Verification
- Concurrency and distributed systems

Education:

- Ph.D. in Computer Science,
  Tata Institute of Fundamental Research, Univ. of Bombay, India, 1997.
  Thesis title: *Models of Asynchronous Processes*
  Advisor(s): R K Shyamasundar and Paritosh K Pandya
- M.Sc.(Tech.) Computer Science,
  Birla Institute of Technology and Science, Pilani, India, 1990.

Current designation:

- Professor (2007 –)
  Chennai Mathematical Institute, Chennai, India

Around 35% of my research has been done since 2007.

Experience

- Oct 1996 to June 2001
  Research Fellow
  Chennai Mathematical Institute, Chennai, India
- June 2001 to March 2007
  Reader/Associate Professor
  Chennai Mathematical Institute, Chennai, India
- Visiting Positions in State Univ of New York, Stony Brook (1997 to 2001)
  Visiting Professorship in Univ. of Paris 7 (2002)
  Regular visits/collaborations with LSV, ENS Cachan and LIAFA, Univ of Paris 7, France and Uppsala University, Sweden.

Summary of Research Work My recent work has been broadly in two areas. First one concerns analyzing concurrent communicating systems in the setting of real-time. The second one concerns the analysis of infinite state systems, in particular, the study of multi-pushdown systems which are formal models of multi-threaded programs. I am currently working on other variants of infinite state sytems (such as those arising from vector additions systems, counter systems and so on) and this area is likely to be the focus of my research in the near future.

In the past I have worked on process algebra formalisms for asynchronously communicating systems, using logic program transformations in program verification, the theory of message sequence charts as well as study of expressive power of temporal logics.

Recent collaborative research projects

1. Member, "INFORMEL", CNRS Laboratoire International Associe, 2012 –
2. Invitation to spend 3 months over 2012-2014 at the Department of Informatics, Uppsala University, Uppsala, Sweden.
3. Member, ARCUS Ile de France-Inde, a 3 year project involving Universities in Paris, CMI and IMSc, 2008–2011
4. Member, "Timed-DISCOVERI" – a three-year (2005–2008) Network Research project funded by CNRS-DST involving CMI, IMSc and IISc from India and LABRI, Univ of Bordeaux, LIAFA, Univ of Paris VII and LSV, ENS de Cachan from France.

Collaboration with Industries

1. Established, along with Madhavan Mukund, the CMI-TCS Ignite research initiative worth Rs. 1.5 Crores (2008-2010).
2. Collaboration with Infosys Labs, Infosys Ltd, Bangalore.

Programming Contests related activities

1. Coach and Academic Coordinator, Indian Computing Olympiad 2003–
2. Deputy Leader, Indian Team to the IOI, 2003–2011.
3. Leader, Indian Team to the IOI, 2012–
4. Coordinator of the ACM Intercollegiate programming contest participation at CMI. Coach, CMI teams at the ACM ICPC World Finals in 2008, 2012 and 2013.

Curriculum Vitae

M. Praveen

Affiliation

Assistant Professor
Chennai Mathematical Institute (CMI)
H1, SIPCOT IT Park, Siruseri
Kelambakkam 603103
India
Phone: +91 44 6748 0966
Fax: +91 44 2747 0225
E-mail: praveenm@cmi.ac.in, Home page: http://www.cmi.ac.in/~praveenm

Research Interests

Computational complexity of modelling and verifying concurrent infinite state systems, logic and parameterized complexity.

Education

- PhD (2011) in Theoretical Computer Science from The Institute of Mathematical Sciences, affiliated to Homi Bhabha National Institute, India. Advisor: Prof. Kamal Lodaya. Thesis: Parameterized complexity of some problems in concurrency and verification.
- MSc (2008) in Theoretical Computer Science from The Institute of Mathematical Sciences, affiliated to Homi Bhabha National Institute, India. Advisor: Prof. Kamal Lodaya. Thesis: Complexity of the reachability problem in subclasses of Petri nets.
- BE (2001) in Electronics and Communications Engineering from R.V.College of Engineering, affiliated to Bangalore University, India.

Present Position

Assistant professor at Chennai Mathematical Institute, from February 2014 onwards. Around 5% of my research has been done in this position.

Experience

- January 2013 to December 2013: Postdoctoral researcher at Laboratoire Bordelais de Recherche en Informatique, France. Funded by the French national research agency (ANR) non-thematic program project ReacHard.

- January 2012 to December 2012: ERCIM postdoctoral researcher at Inria Saclay - Île de France (with DAHU research team working at the Laboratoire Spécification et Vérification campus), ENS Cachan, France.
- October 2011 to December 2011: Research Intern at Microsoft Research, Bangalore, India.
- April 2002 to July 2006: Software engineer at Mindtree Consulting Pvt . Ltd., Bangalore, India.

Awards and Recognitions

- Excellent student paper award at the International Symposium on Parameterized and Exact Computation, 2010. Selected among the papers authored by PhD students submitted to the symposium. Awarded by the program committee of the symposium, for the quality of the paper. Year of award 2010.
- Marie Curie fellowship, co--funded by the European Research Consortium in Informatics and Mathematics (ERCIM). A one year fellowship awarded for carrying out postdoctoral research at a European research institute. Awarded by ERCIM for the year 2012. I used this fellowship for my postdoctoral work at Laboratoire Spécification et Vérification, ENS Cachan, France during January-December 2012.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-13681-113265-110

Name of applicant
Abdulla, Parosh

Date of birth
610618-2550

Title of research programme
Verification of Concurrent Software

Appendix C

List of publications

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Parosh Aziz Abdulla – Publications Since 2006

Citation numbers from Google Scholar 5 April 2014.

Refereed Journal Publications

1. (*) Parosh Aziz Abdulla, Mohamed Faouzi Atig, Othmane Rezine, and Jari Stenman. Budget-Bounded Model-Checking Pushdown Systems. *J. Formal Methods in System Design*, accepted.

2. Parosh Aziz Abdulla, Jonathan Cederberg, and Tomas Vojnar. Monotonic Abstraction for Programs with Multiply-Linked Structures. *Int. J. Found. Comput. Sci.*, 24(2): 187-210, 2013.
   Number of citations: 2.

3. Parosh Aziz Abdulla and Richard Mayr. Priced Timed Petri Nets.
   J. Logical Methods in Computer Science., 9(4): 1-51, 2013.

4. Regular Model Checking for LTL(MSO). *J. Software Tools for Technology Transfer*, 14(2): 223-241, 2012.

5. Parosh Aziz Abdulla, Giorgio Delzanno, and Laurent Van Begin. A classification of the expressive power of well-structured transition systems. *Information and Computation* 209(3): 248-279, 2011.
   Number of citations: 20.

6. Parosh Aziz Abdulla, Giorgio Delzanno, and Ahmed Rezine. Automatic Verification of Directory-Based Consistency Protocols with Graph Constraints. *Int. J. Found. Comput. Sci.* (4): 761-782, 2011.

7. Parosh Aziz Abdulla, Pavel Krcál, and Wang Yi. Sampled Semantics of Timed Automata. *J. Logical Methods in Computer Science* 6(3). 2010.
   Number of citations: 14.

8. Parosh Aziz Abdulla, Noomene Ben Henda, Giorgio Delzanno, and Ahmed Rezine. Monotonic Abstraction (On Efficient Verification of Parameterized Systems). The International Journal of *Foundations of Computer Science* 20(5): 779-801, 2009.
   Number of citations: 13.

9. Parosh Aziz Abdulla, Ahmed Bouajjani, Lukas Holik, Lisa Kaati, and Tomas Vojnar. Composed Bisimulation for Tree Automata. *Int. J. Foundations of Computer Science* 20(4): 685-700, 2009.
   Number of citations: 10.

10. Parosh Aziz Abdulla, Giorgio Delzanno, and Ahmed Rezine. Approximated Parameterized Verification of Infinite-State Processes with Global Conditions. *J. Formal Methods in System Design* 34(2): 126–156, 2009.
    Number of citations: 8.

11. Parosh Aziz Abdulla, Frédéric Haziza, and Mats Kindahl. Model Checking Race-Freeness. ACM SIGARCH *Computer Architecture News* 35(5): 72–79, 2009.
    Number of citations: 2.

12. Parosh Aziz Abdulla, Johann Deneux, Joël Ouaknine, Karin Quaas, and James Worrell. Universality Analysis for One-Clock Timed Automata. *J. Fundamenta Informaticae*, 89(4): 419–450, 2008.
Number of citations: 9.

13. Parosh Aziz Abdulla, Ahmed Bouajjani, and Julien d'Orso. Monotonic and Downward Closed Games. *Logic and Computation* 18(1): 153–169, 2008.
Number of citations: 12.

14. Parosh Aziz Abdulla, Pritha Mahata, and Richard Mayr. Dense-Timed Petri Nets: Checking Zenoness, Token liveness and Boundedness. The Journal of *Logical Methods in Computer Science* 3(1): 61 pages, 2007.
Number of citations: 22.

15. Parosh Aziz Abdulla, Noomene Ben Henda, and Richard Mayr. Decisive Markov Chains. The Journal of *Logical Methods in Computer Science* 3(4): 32 pages, 2007.
Number of citations: 8.

16. Parosh Aziz Abdulla, Johanna Högberg, and Lisa Kaati. Bisimulation Minimization of Tree Automata. The International Journal of *Foundations of Computer Science* 18(4): 699–713, 2007.
Number of citations: 8.

17. Parosh Aziz Abdulla, Johann Deneux, Pritha Mahata, and Aletta Nylén. Using Forward Reachability Analysis for Verification of Timed Petri Nets. The *Nordic Journal of Computing* 14: 1–42, 2007.
Number of citations: 5.

18. Parosh Aziz Abdulla, Axel Legay, Julien d'Orso, Ahmed Rezine. Tree Regular Model Checking: A Simulation-Based Approach. The Journal of *Logic and Algebraic Programming* 69(1-2): 93-121, 2006.
Number of citations: 26.

Refereed Conference Publications

19. (*) Parosh Aziz Abdulla, Mohamed Faouzi Atig, Piotr Hofman, Richard Mayr, K. Narayan Kumar, and Patrick Totzke. Automatic Fence Insertion in Integer Programs via Predicate Abstraction. Proc. *LICS'14*, 29th Annual IEEE Symposium on Logic in Computer Science, 2014.

20. Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jari Stenman. Computing Optimal Reachability Costs in Priced Dense-Timed Pushdown Automata. Proc. *LATA'14* , 8th International Conference on Language and Automata Theory and Applications, Madrid, Spain, 2014.

21. (*) Parosh Abdulla, Stavros Aronis, Bengt Jonsson, and Konstantinos Sagonas. Optimal Dynamic Partial Order Reduction. Proc. *POPL'14*, the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Diego, USA, 2014.

22. Parosh Aziz Abdulla, Lukás Holik, Bengt Jonsson, Ondrej Lengál, Cong Quy Trinh, and Tomás Vojnar. Verification of Heap Manipulating Programs with Ordered Data by Extended Forest Automata. Proc. *ATVA'13*, the 11th International Symposium on Automated Technology for Verification and Analysis, Hanoi, Vietnam, 2013.
Number of citations: 1.

Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jonathan Cederberg. Analysis of Message Passing Programs Using SMT-Solvers. Proc. *ATVA'13*, the 11th International Symposium on Automated Technology for Verification and Analysis, Hanoi, Vietnam, 2013.

23. Parosh Aziz Abdulla, Richard Mayr, Arnaud Sangnier, Jeremy Sproston. Solving Parity Games on Integer Vectors. Proc. *CONCUR'13*, 24th International Conference on Concurrency Theory, Buenos Aires, Argentina, 2013.
    Number of citations: 2.

24. Parosh Aziz Abdulla, Sandhya Dwarkadas, Ahmed Rezine, Arrvindh Shriraman, and Yunyun Zhu. Verifying Safety and Liveness for the FlexTM Hybrid Transactional Memory. Proc. *DATE'13*, Design, Automation and Test in Europe, Grenoble, France, 2013.

25. Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Othmane Rezine. Verification of Directed Acyclic Ad Hoc Networks. Proc. *FMOODS/FORTE'13*, Formal Techniques for Distributed Systems - Joint IFIP WG 6.1 International Conference, FMOODS/FORTE, Florence, Italy, 2013.
    Number of citations: 1.

26. Parosh Aziz Abdulla, Mohamed Faouzi Atig, Giorgio Delzanno and Andreas Podelski. Push-Down Automata with Gap-Order Constraints. Proc. *FSEN'13*, 5th IPM International Conference on Fundamentals of Software Engineering, Tehran, Iran, 2013.
    Number of citations: 1.

27. Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jari Stenman. Zenoness for Timed Pushdown Automata. Proc. *Infinity'13*, 14th International Workshop on Verification of Infinite-State Systems, 2013.

28. Parosh Aziz Abdulla, Lorenzo Clemente, Richard Mayr, and Sven Sandberg. Stochastic Parity Games on Lossy Channel Systems. Proc. *QEST'13*, 10th International Conference on Quantitative Evaluation of Systems,Buenos Aires, Argentina, 2013.
    Number of citations: 1.

29. (*) Parosh Aziz Abdulla, Frdric Haziza, Lukas Holik, Bengt Jonsson, and Ahmed Rezine. An Integrated Specification and Verification Technique for Highly Concurrent Data Structures. Proc. *TACAS'13*, 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Rome, Italy, 2013.
    Number of citations: 4.

30. Parosh Aziz Abdulla, Mohamed Faouzi Atig, Yu-Fang Chen, Carl Leonardsson, and Ahmed Rezine. Memorax, a Precise and Sound Tool for Automatic Fence Insertion under TSO. Proc. *TACAS'13*, 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Rome, Italy, 2013.
    Number of citations: 5.

31. Parosh Aziz Abdulla, Frdric Haziza, Lukas Holik. All for the Price of Few (Parameterized Verification through View Abstraction) Proc. *VMCAI'13*, 14th International Conference on Verification, Model Checking, and Abstract Interpretation, Rome, Italy, 2013.

32. Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jonathan Cederberg. Timed Lossy Channel Systems. Proc. *FSTTCS'12*, IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, Hyderabad, India, 2012.
    Number of citations: 6.

33. Parosh Aziz Abdulla, Mohamed Faouzi Atig, Yu-Fang Chen, Carl Leonardsson, and Ahmed Rezine. Automatic Fence Insertion in Integer Programs via Predicate Abstraction. Proc. *SAS'12*, 19th International Symposium on Static Analysis, 2012.
Number of citations: 6.

34. Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jari Stenman. The Minimal Cost Reachability Problem in Priced Timed Pushdown Systems. Proc. *LATA'12* ,6th International Conference on Language and Automata Theory and Applications, A Coruña, Spain, 2012.
Number of citations: 9.

35. Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jari Stenman. Dense-Timed Pushdown Automata. Proc. *LICS'12*, 27th Annual IEEE Symposium on Logic in Computer Science, 2012.
Number of citations: 13.

36. Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jari Stenman. Adding Time to Pushdown Automata. Proc. *QFM'12*, Quantities in Formal Methods, 2012.
Number of citations: 1.

37. Parosh Aziz Abdulla and Richard Mayr. Petri Nets with Time and Cost. Proc. *Infinity'12*, 14th International Workshop on Verification of Infinite-State Systems, 2012.

38. Parosh Aziz Abdulla, Mohamed Faouzi Atig, Othmane Rezine, and Jari Stenman. Multi-Pushdown Systems with Budgets. Proc. *FMCAD'12*, Formal Methods in Computer-Aided Design, Cambridge, UK, 2012.

39. (*) Parosh Aziz Abdulla, Mohamed Faouzi Atig, Yu-Fang Chen, Carl Leonardsson, and Ahmed Rezine. Counter-Example Guided Fence Insertion under TSO. Proc. *TACAS'12*, 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Tallinn, Estonia, 2012.
Number of citations: 19.

40. Parosh Aziz Abdulla and Richard Mayr. Computing Optimal Coverability Costs in Priced Timed Petri Nets. Proc. *LICS'11*, the 26th IEEE International Symposium on Logic in Computer Science, Ontario, Canada, 2011.
Number of citations: 8.

41. Parosh Aziz Abdulla, Yu-Fang Chen, Lorenzo Clemente, Lukas Holik, Chih-Duo Hong, Richard Mayr, and Tomas Vojnar. Advanced Ramsey-Based Büchi Automata Inclusion Testing. Proc. *CONCUR 2011*, 22nd International Conference on Concurrency Theory, Aachen, Germany, 2011.
Number of citations: 12.

42. Parosh Aziz Abdulla, Giorgio Delzanno, Othmane Rezine, Arnaud Sangnier, and Riccardo Traverso. On the Verification of Timed Ad Hoc Networks. Proc. *FORMATS'11*, 9th International Conference on Formal Modeling and Analysis of Timed Systems, Aalborg, Denmark, 2011.
Number of citations: 8.

43. Parosh Aziz Abdulla, Jonathan Cederberg, and Tomas Vojnar. Monotonic Abstraction for Programs with Multiply-Linked Structures. Proc. *RP'11*, Reachability Problems - 5th International Workshop, RP 2011, Genoa, Italy, 2011.

44. Parosh Aziz Abdulla, Yu-Fang Chen, Lorenzo Clemente, Lukas Holik, Chih-Duo Hong, Richard Mayr, and Tomas Vojnar. Simulation Subsumption in Ramsey-based Büchi Automata Universality and Inclusion Testing. Proc. *CAV'10*, the 22nd International Conference on Computer Aided

Verification, Edinburgh, Scotland, Volume 6174 of *Lecture Notes in Computer Science*, pages 132–147, 2010.
Number of citations: 13.

45. Parosh Aziz Abdulla, Yu-Fang Chen, Giorgio Delzanno, Frédéric Haziza, Chih-Duo Hong, and Ahmed Rezine. Constrained Monotonic Abstraction: A CEGAR for Parameterized Verification. Proc. *CONCUR'10*, the 21th International Conference on Concurrency Theory, Paris, France, Volume 6269 of *Lecture Notes in Computer Science*, pages 86-101, 2010.
Number of citations: 11.

46. Parosh Aziz Abdulla, Yu-Fang Chen, Lukas Holik, Richard Mayr, and Tomas Vojnar. When Simulation Meets Antichains. Proc. *TACAS'10*, the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Paphos, Cyprus, Volume 6015 of *Lecture Notes in Computer Science*, pages 93–108, 2010.
Number of citations: 44.

47. Parosh Aziz Abdulla, Jonathan Cederberg, and Lisa Kaati. Analyzing the Security in the GSM Radio Network Using Attack Jungles. Proc. *ISoLA'10*, the 4th International Symposium on Leveraging Applications, Crete, Greece, Volume 6415 of *Lecture Notes in Computer Science*, pages 60–74, 2010.
Number of citations: 4.

48. Parosh Aziz Abdulla, Yu-Fang Chen, Lukas Holik, and Tomas Vojnar. Mediating for Reduction (on Minimizing Alternating Bchi Automata). Proc. *FSTTCS'09, the 29th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, Kanpur, India,* Leibniz International Proceedings in Informatics*, 2009.*
Number of citations: 6.

49. Parosh Aziz Abdulla, Muhsin Atto, Jonathan Cederberg, and Ran Ji. Automated Analysis of Data-Dependent Programs with Dynamic Memory. Proc. ATVA'09*, the 7th International Symposium on Automated Technology for Verification and Analysis, Macao SAR, China, Volume 5799 of* Lecture Notes in Computer Science*, pages 348–363, 2009.*
Number of citations: 5.

50. Parosh Aziz Abdulla and Richard Mayr. Minimal Cost Reachability/Coverability in Priced Timed Petri Nets. Proc. FoSSaCS'09*, the 12th International Conference on Foundations of Software Science and Computation Structures, York, UK, Volume 5504 of* Lecture Notes in Computer Science*, pages 348–363, 2009.*
Number of citations: 24.

51. Parosh Aziz Abdulla, Giorgio Delzanno, and Laurent Van Begin. A Language-Based Comparison of Extensions of Petri Nets with and without Whole-Place Operations. Proc. LATA'09*, the 3rd International Conference on Language and Automata Theory and Applications, Tarragona, Spain, Volume 5457 of* Lecture Notes in Computer Science*, pages 71–82, 2009.*
Number of citations: 8.

52. Parosh Aziz Abdulla, Giorgio Delzanno, and Ahmed Rezine. Approximated Context-sensitive Analysis for Parameterized Verification. Proc. FORTE'09*, the 29th IFIP International Conference on Formal Techniques for Networked and Distributed Systems, Lisbon, Portugal, Volume 5504 of* Lecture Notes in Computer Science*, pages 348–368, 2009.*
Number of citations: 9.

53. *Parosh Aziz Abdulla, Giorgio Delzanno, and Ahmed Rezine. Automatic Verification of Directory-based Consistency Protocols. Proc.* RP'09*, the 2nd Colloquium on Reachability Problems, Paris, France, 2009.*
    Number of citations: 3.

54. *Parosh Aziz Abdulla, Pavel Krcal, and Wang Yi. R-Automata. Proc.* CONCUR'08*, the 19th International Conference on Concurrency Theory, Toronto, Canada, Volume 5201 of* Lecture Notes in Computer Science*, pages 67–81, 2008.*
    Number of citations: 14.

55. *Parosh Aziz Abdulla, Ahmed Bouajjani, Lukas Holik, Lisa kaati, and Tomas Vojnar. Composed Bisimulation for Tree Automata. Proc.* CIAA'08*, the 13th International Conference on Implementation and Application of Automata, San Fransisco, USA, Volume 5148 of* Lecture Notes in Computer Science*, pages 212–222, 2008.*
    Number of citations: 10.

56. *Parosh Aziz Abdulla, Ahmed Bouajjani, Jonathan Cederberg, Frédéric Haziza, and Ahmed Rezine. Monotonic Abstraction for Programs with Dynamic Memory Heaps. Proc.* CAV'08*, the 20th International Conference on Computer Aided Verification, Princeton, USA, Volume 5123 of* Lecture Notes in Computer Science*, pages 341–354, 2008.*
    Number of citations: 32.

57. *Parosh Aziz Abdulla, Noomene Ben Henda, Giorgio Delzanno, Frédéric Haziza, and Ahmed Rezine. Parameterized Tree Systems. Proc.* FORTE'08*, the 28th International Conference on Formal Techniques for Networked and Distributed Systems, Tokyo, Japan, Volume 5048 of* Lecture Notes in Computer Science*, pages 69–83, 2008.*
    Number of citations: 9.

58. *Parosh Aziz Abdulla, Ahmed Bouajjani, Lukas Holik, Lisa kaati, and Tomas Vojnar. Computing Simulations over Tree Automata: Efficient Techniques for Reducing Tree Automata. Proc.* TACAS'08*, the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Budapest, Hungary, Volume 4963 of* Lecture Notes in Computer Science*, pages 93–108, 2008.*
    Number of citations: 25.

59. *Parosh Aziz Abdulla, Noomene Ben Henda, Luca de Alfaro, Richard Mayr, and Sven Sandberg. Stochastic Games with Lossy Channels. Proc.* FoSSaCS'08*, the 11th International Conference on the Foundations of Software Science and Computation Structures, Budapest, Hungary, Volume 4962 of* Lecture Notes in Computer Science*, pages 35–49, 2008.*
    Number of citations: 10.

60. *Parosh Aziz Abdulla, Noomene Ben Henda, Giorgio Delzanno, and Ahmed Rezine. Handling Parameterized Systems with Non-Atomic Global Conditions. Proc.* VMCAI'08*, the 9th International Conference on Verification, Model Checking and Abstract Interpretation, San Francisco, USA, Volume 4905 of* Lecture Notes in Computer Science*, pages 22–36, 2008.*
    Number of citations: 23.

61. *Parosh Aziz Abdulla, Giorgio Delzanno, and Laurent Van Begin. On the Qualitative Analysis of Conformon P Systems. Proc.* WMC'08*, the 9th Workshop on Membrane Computing, Edinburgh, UK, Volume 5391 of* Lecture Notes in Computer Science*, pages 78–94, 2008.*

62. *Parosh Aziz Abdulla, Giorgio Delzanno, and Ahmed Rezine. Monotonic Abstraction in Action: Automatic Verification of Distributed Mutex Algorithms. Proc.* ICTAC'08*, the 5th International Colloquium on Theoretical Aspects of Computing, Istanbul, Turkey, Volume 5160 of* Lecture Notes in Computer Science*, pages 50–65, 2008.*

63. *Parosh Aziz Abdulla, Pavel Krcal, and Wang Yi. Universality of R-automata with Value Copying. Proc.* INFINITY'08*, the 10th International Workshop on Verification of Infinite-State Systems, Toronto, Canada, 2008.*

64. *Parosh Aziz Abdulla, Lukas Holik, Lisa kaati, and Tomas Vojnar. A Uniform (Bi-)Simulation-Based Framework for Reducing Tree Automata. Proc.* MEMICS'08*, the 4th Annual Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, Brno, Czech Republic, 2008.*
Number of citations: 5.

65. *Parosh Aziz Abdulla, Giorgio Delzanno, and Laurent Van Begin. Comparing the Expressive Power of Well-structured Transition Systems. Proc.* CSL'07*, the 16th EACSL Annual Conference on Computer Science and Logic, Lausanne, Switzerland, Volume 4646 of* Lecture Notes in Computer Science*, pages 99–114, 2007.*
Number of citations: 15.

66. *Parosh Aziz Abdulla, Giorgio Delzanno, and Ahmed Rezine. Parameterized Verification of Infinite-state Processes with Global Conditions. Proc.* CAV'07*, the 19th International Conference on Computer Aided Verification, Berlin, Germany, Volume 4590 of* Lecture Notes in Computer Science*, pages 145–157, 2007.*
Number of citations: 59.

67. *Parosh Aziz Abdulla, Noomene Ben Henda, Giorgio Delzanno, and Ahmed Rezine. Regular Model Checking without Transducers. Proc.* TACAS'07*, the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Braga, Portugal, Volume 4424 of* Lecture Notes in Computer Science*, pages 602–617, 2007.*
Number of citations: 65.

68. *Parosh Aziz Abdulla, Pavel Krcal, and Wang Yi. Sampled Universality of Timed Automata. Proc.* FoSSaCS'07*, the 10th International Conference on the Foundations of Software Science and Computation Structures, Braga, Portugal, Volume 4423 of* Lecture Notes in Computer Science*, pages 2–16, 2007.*
Number of citations: 7.

69. *Parosh Aziz Abdulla, Joel Ouaknine, Karin Quaas, and James Worrell. Zone-Based Universality Analysis for Single-Clock Timed Automata. Proc.* FSEN'07*, the IPM International Symposium on Fundamentals of Software Engineering, Tehran, Iran, Volume 4767 of* Lecture Notes in Computer Science*, pages 98–112, 2007.*
Number of citations: 6.

70. *Parosh Aziz Abdulla, Johanna Högberg, and Lisa Kaati. Bisimulation Minimization of tree automata. Proc.* CIAA'06*, the 11th International Conference on Implementation and Application of Automata, Taipei, Taiwan, Volume 4094 of* Lecture Notes in Computer Science*, pages 173–185, 2006.*
Number of citations: 7.

71. *Parosh Aziz Abdulla, Noomene Ben Henda, Richard Mayr, and Sven Sandberg. Limiting Behavior of Markov Chains with Eager Attractors. Proc.* QEST'06*, the 3rd International Conference on Quantitative Evaluation of SysTems, University of California, Riverside, IEEE Computer Society, pages 253–264, 2006.*
`Number of citations: 5.`

72. *Parosh Aziz Abdulla, Noomene Ben Henda, Richard Mayr, and Sven Sandberg. Eager Markov Chains. Proc.* ATVA'06*, the 4th Symposium on Automated Technology for Verification and Analysis, Beijing, China, Volume 4218 of* Lecture Notes in Computer Science*, pages 24–38, 2006.*
`Number of citations: 5.`

73. *Parosh Aziz Abdulla, Bengt Jonsson, Ahmed Rezine, and Mayank Saksena. Proving Liveness by Backwards Reachability. Proc.* CONCUR'06*, The 17th International Conference on Concurrency Theory, Bonn, Germany, Volume 4137 of* Lecture Notes in Computer Science*, pages 31–42, 2006.*
`Number of citations: 14.`

74. *Parosh Aziz Abdulla, Johann Deneux, Lisa Kaati, and Marcus Nilsson. Minimization of Nondeterministic Automata with Large Alphabets. Proc.* CIAA'06*, The 10th International Conference on Implementation and Application of Automata, Sophia Antipolis, France Volume 3845 of* Lecture Notes in Computer Science*, pages 95–109, 2006.*
`Number of citations: 6.`

75. *Parosh Aziz Abdulla and Giorgio Delzanno. Constrained Multiset Rewriting. Proc.* AVIS'06*, the 5th International Workshop on Automated Verification of Infinite-State Systems, Vienna, Austria, 2006.*
`Number of citations: 27.`

Books and Book Chapters

76. Parosh Aziz Abdulla, Prasad Sistla, and Murali Talupur. Parameterized Systems. In the Handbook of Model Checking. To appear in 2014.

77. Parosh Aziz Abdulla and K. Rustan M. leino. Tools for software verification - Introduction to the special section from the seventeenth international conference on tools and algorithms for the construction and analysis of systems. *J. Software Tools for Technology Transfer*, 15(2): 85-88, 2013.

78. Parosh Aziz Abdulla and Giorgio Delzanno. Editor. Proceedings of *RP'13*, the 6th Colloquium on Reachability Problems, Uppsala, Sweden, , Volume 8169 of *Lecture Notes in Computer Science*, 2013.

79. Parosh Aziz Abdulla. Regular Model Checking. Editor of the special issue of STTT. *J. Software Tools for Technology Transfer*, 14(2): 109-118, 2012.

80. Parosh Aziz Abdulla and K. Rustan M. Leino. Editor. Proceedings of TACAS 2011, the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Volume 6605 of *Lecture Notes in Computer Science*, pages 50–65, 2011.

Invited Contributions

81. Parosh Aziz Abdulla. From Transition Systems to Markov Chains and Back. Unifying invited talk for *CONCUR'11*, the 22nd International Conference on Concurrency Theory, and *QEST'11*, the 8th International Conference on Quantitative Evaluation of SysTems.

82. Parosh Aziz Abdulla and K. Rustan M. Leino . Selected Papers from TACAS 2011. Guest editor of the journal of *Logical Methods in Computer Science*.

83. Parosh Aziz Abdulla. Forcing monotonicity in parameterized verification: From Parameterized Verification to Heap Manipulation. Proc. *SOFSEM'10*, the 36th International Conference on Current Trends in Theory and Practice of Computer Science, Špindleruv Mlýn, Czech Republic, 2010.
Number of citations: 5.

84. Parosh Aziz Abdulla. Well- and Better-Quasi Ordered Transition Systems. The Journal of *Bulletin of Symbolic Logic*. 2010.
Number of citations: 11.

85. Parosh Aziz Abdulla. Infinite-State Verification: From Transition Systems to Markov Chains. Proc. *QEST'09*, the 6th International Conference on Quantitative Evaluation of SysTems, Budapest, Hungary, IEEE Computer Society, pages 197–212, 2009.

86. Parosh Aziz Abdulla. Parameterized Verification via Monotonic Abstraction, *MEMICS'08*, the 4th Annual Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, Brno, Czech Republic, 2008.

87. Parosh Aziz Abdulla, Giorgio Delzanno, and Ahmed Rezine. Monotonic Abstraction in Parameterized Verification. Proc. *RP'08*, the 2nd Workshop on Reachability Problems, Liverpool, UK, 2008

88. Parosh Aziz Abdulla. Shape Analysis via Monotonic Abstraction. *Beyond the Finite: New Challenges in Verification and Semistructured Data*, Dagstuhl Seminar 08171, 2008.

Five Most Cited Publications

- Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Tsay Yih-Kuen. General Decidability Theorems for Infinite-State Systems. Proc. *LICS'96*, the 11th IEEE International Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, IEEE Computer Society, pages 313–321, 1996.
Number of citations: 354.

- Parosh Aziz Abdulla and Bengt Jonsson. Verifying Programs with Unreliable Channels. Proc. *LICS'93*, the 8th IEEE International Symposium on Logic in Computer Science, Montréal, Canada, IEEE Computer Society, pages 57–63, 1993.
Number of citations: 334.

- Parosh Aziz Abdulla, Per Bjesse, and Niklas Eén. Symbolic Reachability Analysis Based on SAT-Solvers. Proc. *TACAS'00*, the 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Berlin, Germany, Volume 1785 of *Lecture Notes in Computer Science*, pages 395–410, 2000.
Number of citations: 216.

- Parosh Aziz Abdulla, Karlis Čerāns, Bengt Jonsson, and Tsay Yih-Kuen. Algorithmic Analysis of Programs with Well Quasi-ordered Domains. *Information and Computation* 160: 109–127, 2000.
  Number of citations: 158.

- Parosh Aziz Abdulla, Ahmed Bouajjani and Bengt Jonsson. On-the-Fly Analysis of Systems with Unbounded, Lossy FIFO Channels. Proc. *CAV'98*, the 10th International Conference on Computer Aided Verification, Vancouver, BC, Canada, Volume 1427 of *Lecture Notes in Computer Science*, pages 305–318, 1998.
  Number of citations: 157.

Publication List

Aiswarya Cyriac

Citation numbers from Google Scholar 8 April 2014

Peer-reviewed original articles

1. Benedikt Bollig, Aiswarya Cyriac, Paul Gastin, and Marc Zeitoun. Temporal logics for con- current recursive programs: Satisfiability and model checking. In Journal of Applied Logic, To appear, 2014.

   Number of citations: 0

Peer-reviewed conference contributions

2. Benedikt Bollig, Aiswarya Cyriac, Loıc Helouet, Ahmet Kara, and Thomas Schwentick. Dynamic communicating automata and branching high-level MSCs. In Proceedings of the 7th Language and Automata Theory and Applications (LATA'13), volume 7810 of Lecture Notes in Computer Science. Springer, 2013.

   Number of citations: 0

3. Aiswarya Cyriac, Paul Gastin, and K. Narayan Kumar. MSO decidability of multi-pushdown systems via split-width. In Maciej Koutny and Irek Ulidowski, editors, Proceedings of the 23rd International Conference on Concurrency Theory (CONCUR'12), volume 7454 of Lecture Notes in Computer Science, pages 547–561, Newcastle, UK, September 2012. Springer

   Number of citations: 12

4. Benedikt Bollig, Aiswarya Cyriac, Paul Gastin, and K. Narayan Kumar. Model checking languages of data words. In Lars Birkedal, editor, Proceedings of the 15th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'12), volume 7213 of Lecture Notes in Computer Science, pages 391–405, Tallinn, Estonia, March 2012. Springer

   Number of citations: 9

5. Benedikt Bollig, Aiswarya Cyriac, Paul Gastin, and Marc Zeitoun. Temporal logics for con- current recursive programs: Satisfiability and model checking. In Filip Murlak and Piotr Sankowski, editors, Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science (MFCS'11), volume 6907 of Lecture Notes in Computer Science, pages 132–144, Warsaw, Poland, August 2011. Springer

   Number of citations: 8

Publications List (Last 8 years and 5 highly cited)

(with citation numbers from Google Scholar)

K Narayan Kumar

For a complete list see `http://scholar.google.co.in/citations?user=tFCSOd0AAAAJ`.

Peer-reviewed Original Articles

1. Abhik Roychoudhury, K. Narayan Kumar, C.R. Ramakrishnan and I.V. Ramakrishnan: "A Unfold/Fold transformation framework for Definite Logic Programs", ACM Transactions on Programming Language Systems 26(3): 464-509 (2004).
   Number of citations: 28

2. Jesper Henrikson, Madhavan Mukund, K. Narayan Kumar, Milind Sohoni and P.S. Thiagarajan: "A Theory of Regular MSC Languages", *Information and Computation* 202(1): 1-38 (2005).
   Number of citations: 95

3. S. Akshay, B. Bollig, P. Gastin, M. Mukund and K. Narayan Kumar: Distributed Timed Automata with Independently Evolving Clocks. Fundam. Inform. 130(4): 377-407 (2014).

Peer-reviewed Original Conference Contributions

1. J.G. Henriksen, M. Mukund, K. Narayan Kumar and P.S. Thiagarajan: "On Message Sequence Graphs and Finitely Generated Regular MSC Languages", *Proc. International Colloquium on Automata, Languages and Programming (ICALP) 2000*, Springer Lecture Notes in Computer Science, **1854**, (2000) 675–686.
   Number of citations: 86

2. M. Mukund, K. Narayan Kumar, M. Sohoni: "Synthesizing distributed finite-state systems from MSCs", *Proc. 11th International Conference on Concurrency Theory (CONCUR) 2000*, Springer Lecture Notes in Computer Science, **1877**, (2000) 521–535.
   Number of citations: 67

3. J.G. Henriksen, M. Mukund, K. Narayan Kumar and P.S. Thiagarajan: "Regular Collections of Message Sequence Charts", *Proc. Mathematical Foundations of Computer Science (MFCS) 2000*, Springer Lecture Notes in Computer Science, **1893**, (2000) 405–414.
   Number of citations: 64

4. Puneet Bhateja, Paul Gastin, Madhavan Mukund, K. Narayan Kumar: "Local Testing of Message Sequence Charts Is Difficult", Proceedings of the 16th International Symposium on the *Foundations of Computing Theory* (FCT 2007), Springer LNCS **4639** (2007) 76-87.
   Number of citations: 8

5. S. Akshay, Madhavan Mukund, K. Narayan Kumar: "Checking Coverage for Infinite Collections of Timed Scenarios", Proceedings of the 18th *International Conference on Concurrency* (CONCUR 2007), Springer LNCS **4703** (2007) 181-196.
Number of citations: 12

6. S. Akshay, B. Bollig, P. Gastin, Madhavan Mukund and K. Narayan Kumar: "Distributed Timed Automata with Independently Evolving Clocks", Proceedings of the 19th *International Conference on Concurrency* (CONCUR 2008), Springer LNCS **5201** (2008) 82-97.
Number of citations: 22

7. Joy Chakraborty, Deepak D'Souza, K. Narayan Kumar: Analysing Message Sequence Graph Specifications. ISoLA (1) 2010, Springer LNCS **6415** (2010) 549-563
Number of citations: 8

8. S. Akshay, Paul Gastin, Madhavan Mukund, K. Narayan Kumar: Model checking time-constrained scenario-based specifications. Proceedings of the 30th International Conference on *Foundations of Software Technology and Theoretical Computer Science* (FSTTCS 2010), LIPIcs 8 Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010) 204-215.
Number of citations: 7

9. Benedikt Bollig, Aiswarya Cyriac, Paul Gastin, K. Narayan Kumar: Model Checking Languages of Data Words. Proceedings of the 15th International Conference on *Foundations of Software Science and Computational Structures* (FoSSaCS 2012):Springer LNCS **7213** (2012) 391-405.
Number of citations: 9

*9. Aiswarya Cyriac, Paul Gastina and K. Narayan Kumar: MSO Decidability of Multi-Pushdown Systems via Split-Width. Proceedings of the 23rd International Conference on *Concurrency Theory* (CONCUR 2012):Springer LNCS **7454** (2012) 547-561
Number of citations: 12

*10. M. F. Atig, A. Bouajjani, K. Narayan Kumar and P. Saivasan: Linear-Time Model-Checking for Multithreaded Programs under Scope-Bounding. Proceedings of the 10th International Symposium on *Automated Technology for Verification and Analysis* (ATVA 2012):Springer LNCS **7561** (2012) 152-166
Number of citations: 8

*11. M. F. Atig, K. Narayan Kumar and P. Saivasan: Adjacent Ordered Multipushdown Systems. Proceedings of the 17th International Conference on *Developments in Language Theory* (DLT 2012):Springer LNCS **7907** (2013) 58-69
Number of citations: 2

*12. Parosh Abdulla, M. F. Atig, P. Hofman, R. Mayr, K. Narayan Kumar and P. Totzke: Infinite State Energy Games. To Appear in the Joint Proceedings of the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) and 23rd EACSL Annual Conference on Computer Science Logic (CSL) 2014.

Articles in Books

1. Madhavan Mukund, K. Narayan Kumar, Y. Shaofa and P.S. Thiagarajan: "Anchored Concatenation of MSCs", in *Formal Models, Languages and Applications*, Volume 66, Series in Machine Perception and Artificial Intelligence, World Scientific, 2006.

2. P. Gastin, Madhavan Mukund and K. Narayan Kumar: "Reachability and Boundedness in Time-Constrained MSC Graphs", in *Perspectives in Concurrency Theory*, Universties Press, 2008.

3. K Narayan Kumar: The Theory of MSC Languages, in *Modern Applications of Automata Theory*, IISc Research Monographs Series, Vol 2, World Scientific, Singapore (2012) 289-324

Edited Volumes

1. Ravi Kannan, K. Narayan Kumar (Eds.): 29th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2009), December 15-17, 2009, IIT Kanpur, India, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik LIPIcs **4** 2009.

Manuscripts

*1 M. F. Atig, A. Bouajjani, K. Narayan Kumar and P. Saivasan: Model checking Branching-Time Properties of Multi-Pushdown Systems is Hard. *CoRR abs/1205.6928* (2012)
Number of citations: 3

*2 A. Cyriac, P. Gastin, K. Narayan Kumar: Verifying Communicating Multi-pushdown Systems. (http://hal.archives-ouvertes.fr/hal-00943690) (2014)

*3 M. F. Atig, A. Bouajjani, K. Narayan Kumar and P. Saivasan: On Bounded Reachability Analysis of Shared Memory Systems. (Submitted) (2014)

4. S. Akshay, P. Gastin, M. Mukund and K. Narayan Kumar: Checking Conformace for time-contrained scenario-based Specifications. (Submitted) (2014)

Publication List

M. Praveen

Database used for citation data: Google Scholar.

Peer-reviewed original articles

1. M. Praveen. Does treewidth help in modal satisfiability? *ACM Transactions on Computational Logic*, 14(3):1--18, 2013. Number of citations: 0.

2. M. Praveen. Small vertex cover makes Petri net coverability and boundedness easier. *Algorithmica*, 65(4):713--753, 2013. Number of citations: 0.

Peer-reviewed conference contributions

1. * Jérôme Leroux, M. Praveen, and Grégoire Sutre. Hyper-Ackermannian bounds for pushdown vector addition systems. In *Proceedings of the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (to appear)*, 2014. Number of citations: 0.

2. * Gilles Geeraerts, Alexander Heußner, M. Praveen, and Jean-Francois Raskin. ω-Petri nets. In José Manuel Colom and Jörg Desel, editors, *Proceedings of the 34th international conference on application and theory of Petri nets and concurrency*, volume 7927 of *Lecture Notes in Computer Science*, pages 49--69. Springer Berlin Heidelberg, 2013. Full version here. Number of citations: 5.

3. * Stéphane Demri, Diego Figueira, and M. Praveen. Reasoning about data repetitions with counter systems. In Orna Kupferman, editor, *Proceedings of the 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS 2013, pages 33--42. IEEE Computer Society, 2013. Number of citations: 0.

4. Jérôme Leroux, M. Praveen, and Grégoire Sutre. A relational trace logic for vector addition systems with application to context-freeness. In Pedro R. D'Argenio and Hernán Melgratti, editors, *Proceedings of the 24th International Conference on Concurrency Theory*, volume 8052 of *Lecture Notes in Computer Science*, pages 137--151. Springer-Verlag Berlin Heidelberg, 2013. Full version here. Number of citations: 0.

5. Rémi Bonnet, Alain Finkel, and M. Praveen. Extending the Rackoff technique to Affine nets. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 18 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 301--312. Schloss Dagstuhl--Leibniz-Zentrum fuer Informatik, 2012. Full version here. Number of citations: 0.

6. Kamal Lodaya and M. Praveen. Parameterized complexity results for 1-safe Petri nets. In Joost-Pieter Katoen and Barbara König, editors, *Proceedings of the 22nd International Conference on Concurrency Theory*, volume 6901 of *Lecture Notes in Computer Science*,

pages 358--372. Springer Berlin Heidelberg, 2011. Full version here. Number of citations: 3.

7. M. Praveen. Does treewidth help in modal satisfiability? In Petr Hliněný and Antonín Kučera, editors, *Proceedings of the 35th International Symposium on Mathematical Foundations of Computer Science*, volume 6281 of *Lecture Notes in Computer Science*, pages 580--591. Springer Berlin Heidelberg, 2010. Full version here. Number of citations: 4.

8. * M. Praveen. Small vertex cover makes Petri net coverability and boundedness easier. In Venkatesh Raman and Saket Saurabh, editors, *Proceedings of the International Symposium on Parameterized and Exact Computation*, volume 6478 of *Lecture Notes in Computer Science*, pages 216--227. Springer Berlin Heidelberg, 2010. Got the excellent student paper award of this symposium. Full version here. Number of citations: 5.

9. * Kamal Lodaya and M. Praveen. Modelchecking counting properties of 1-safe nets with buffers in paraPSPACE. In Ravi Kannan and K. Narayan Kumar, editors, *Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 4 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 347--358. Schloss Dagstuhl--Leibniz-Zentrum fuer Informatik, 2009. Full version here. Number of citations: 6.

10. Kamal Lodaya and M. Praveen. Analyzing reachability for some Petri nets with fast growing markings. In Vesa Halava and Igor Potapov, editors, *Proceedings of the 2nd workshop on reachability problems*, volume 223 of *Electronic Notes in Theoretical Computer Science*, pages 215--237. Elsevier, 2008. Full version here. Number of citations: 4.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-13681-113265-110

Name of applicant
Abdulla, Parosh

Date of birth
610618-2550

Title of research programme

Appendix N

Budget and research resources

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Budget and Research Resources

The requested budget is for:

- 80% of the time of 2 new PhD students, their remaining 20% coming from teaching.
- 25% of the time of 1 PostDoc researcher, her remaining 75% coming from faculty research funding and teaching.
- We are also applying for separate funding to support scientific collaboration between the group and the *Chennai Mathematical Institute*. The funding will cover travel costs, equipment, and 25% of the time of 1 PhD student who will work on common project topics.
- Other costs are for the local infrastructure on Department level: administrative and technical support, computer network, telephones, copying, etc.

The research of the algorithmic verification group is partially funded by Uppsala University, *UPMARC: Uppsala Programming for Multicore Architectures Research Center* which is a *Linnaeus Grant by VR* at Uppsala University, from 2008 to 2017, and *Automata, Probabilities, and Games*, an individual VR project ending in 2014. The research within UPMARC concentrates on issues related to multicore architectures and is complementary to the topics of this proposal. The research of Parosh Abdulla is supported by faculty funding (80%).

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod

Name of applicant

Date of birth

Title of research programme

Appendix d1

Collaboration with Indian researchers

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Planned Visits

1.1 India To Sweden

	Visiting Scientist	Period and Duration of Visit	Purpose of Visit
1st Year	M Praveen	May-June 2015, 30 Days	Research Discussions
	K Narayan Kumar	June-July 2015, 30 Days	Research Discussions
2nd Year	M Praveen	May-June 2016, 30 Days	Research Discussions
	K Narayan Kumar	June-July 2016, 30 Days	Research Discussions
3rd Year	M Praveen	May-June 2017, 30 Days	Research Discussions
	K Narayan Kumar	June-July 2017, 30 Days	Research Discussions

1.2 Sweden to India

	Visiting Scientist	Period and Duration of Visit	Purpose of Visit
1st Year	Parosh Abdulla	November-December 2015, 30 days	Research Discussions
	Aiswarya Cyriac	November-December 2015, 30 days	Research Discussions
2nd Year	Parosh Abdulla	November-December 2016, 30 days	Research Discussions
	Aiswarya Cyriac	November-December 2016, 30 days	Research Discussions
3rd Year	Parosh Abdulla	November-December 2017, 30 days	Research Discussions
	Aiswarya Cyriac	November-December 2017, 30 days	Research Discussions

Requested Funds

2.3 Swedish Applicant (Costs in SEK)

Year	Mobility Expenses for Visits from Sweden to India (Airfare + Insurance)	Mobility Expenses for visits from India to Sweden (Accommodation + Per Diem Etc.)	Man-power (PhD Stud / Post-Doc)	Minor Equipments/ Accessories	Chemicals/ Consumables	Total
1st Year	20.000	60.000	150.000	20.000	0	250.000
2nd Year	20.000	60.000	150.000	20.000	0	250.000
3rd Year	20.000	60.000	150.000	20.000	0	250.000
Total	60.000	180.000	450.000	60.000	0	750.000

2.4 Indian Applicant (Costs in Rupees)

Year	Mobility Expenses for Visits from India to Sweden (Airfare + Insurance)	Mobility Expenses for visits from Sweden to India (Accommodation + Per Diem Etc.)	Man-power (JRF/ SRF/ Post-Doc)	Minor Equipments/ Accessories	Chemicals/ Consumables	Total
1st Year	1,60,000[1]	3,90,000[2]	0	2,00,000[4]	0	7,50,000
2nd Year	1,60,000[1]	3,90,000[2]	3,90,000[3]	1,40,000[5]	0	10,80,000
3rd Year	1,60,000[1]	3,90,000[2]	3,90,000[3]	0	0	9,40,000
Total	4,80,000	11,70,000	7,80,000	3,40,000	0	27,70,000

Notes:

1. Two air tickets plus overseas travel insurance.
2. Two visits of 30 days each at Rs. 4000 (accommodation) + Rs. 2500 (expenses) per day.
3. One Post-doctoral Fellowship (25000pm + 30% HRA).
4. Two Macbook Air Laptops including apple care 2 year warranty, taxes.
   A Sample quote is enclosed. The cost here includes taxes and rounded off to account for increase in price over time.
5. One iMac including 2 year warranty, taxes.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-13681-113265-110

Dnr
2014 -

Name of applicant
Abdulla, Parosh

Date of birth
610618-2550

Reg date
2014-04-09 12:17:50

Project title
Verification of Concurrent Software

NT

Research grants NT April 9, 2014
Project Research Grant
*Övrigt generellt
Uppsala universitet

Appendix S

Signatures

NT-2

2015	2016	2017	2018	2019	
1602	1665	1766	1622	1727	2015-01-01 -- 2019-12-31

Applicant | **Date**

Head of department at host University | **Clarifi cation of signature** | **Telephone**

A signature on the application is required not only from the applicant but also from the authorised representative of the administrating organisation/institution or equivalent (normally the head of the department or establishment where the research is to be conducted). The signature confirms that the department can accommodate the proposed research, position or equipment; that the costing in the application is approved for the department's part, that any proposed experimentation on human or animal subjects has been reported, and that the applicant has reported any secondary occupations and commercial ties (s)he may have, and nothing inconsistent with good research practice has thereby emerged. The applicant must have discussed these conditions with the representative of the administrating organisation/institution or equivalent before the latter approves and signs the application.

NOTE the exceptions!
Please note that a different procedure applies for the following types of grants:

- Infrastructure- Operation Grants
- Infrastructure- Grants for Expensive Equipment
- Infrastructure- Grants for Large Databases
- Framework Grant SIMSAM
- International Recruitment

For these types of grants the vice-chancellor, instead of the head of department, must sign Appendix S. Thereby (s)he confirms the commitments of the administrative organization. The details are specified in the Call for Proposals, published at www.vr.se.

Vetenskapsrådets noteringar
Kod

2014-13681-113265-110

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se