06121 Abstracts Collection
Atomicity: A Unifying Concept in Computer Science
— Dagstuhl Seminar —

Clifford B. Jones[1], David Lomet[2], Alexander Romanovsky[3] and Gerhard Weikum[4]

[1] University of Newcastle, GB
Cliff.Jones@ncl.ac.uk
[2] Microsoft Research - Redmond, US
lomet@microsoft.com
[3] University of Newcastle, GB
alexander.romanovsky@ncl.ac.uk
[4] MPI für Informatik, DE
weikum@mpi-sb.mpg.de

Abstract. From 19.03.06 to 24.03.06, the Dagstuhl Seminar 06121 "Atomicity: A Unifying Concept in Computer Science" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Formal methods, dependability, fault tolerance, atomic actions, databases, advanced transactional models, system structuring

06121 Executive Summary – Atomicity: A Unifying Concept in Computer Science

This seminar was based on and continued the interaction of different computer-science communities that was begun in an earlier Dagstuhl seminar in April 2004. Both seminars have aimed at a deeper understanding of the fundamental concept of atomic actions and their roles in system design, execution, modeling, and correctness reasoning, and at fostering collaboration, synergies, and a unified perspective across largely separated research communities. Each of the two seminar brought together about 30 researchers and industrial practitioners from the four areas of database and transaction processing systems, fault tolerance and dependable systems, formal methods, and to smaller extent, hardware architecture and programming languages. The interpretations and roles of

the atomicity concept(s) vary substantially across these communities. For example, the emphasis in database systems is on algorithms and implementation techniques for atomic transactions, whereas in dependable systems and formal methods atomicity is viewed as an intentionally imposed or postulated property of system components to simplify designs and increase dependability. Nevertheless, all four communities share the hope that it will eventually be possible to unify the different scientific viewpoints into more coherent foundations, system development principles, design methodologies, and usage guidelines.

Joint work of: Jones, Clifford B.; Lomet, David; Romanovsky, Alexander; Weikum, Gerhard

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2006/835

06121 Report: Break Out Session on Guaranteed Execution

Calton Pu (Georgia Institute of Technology, USA)

The break out session discussed guaranteed properties during program execution. Using a workflow example application, we discussed several research topics that form part of the guaranteed properties, including declarative specifications, generation of workflow program, generation of invariant guards, automated failure analysis, automated repair, and automated reconfiguration of workflow.

Keywords: Guaranteed properties, declarative specifications, generation of workflow program, generation of invariant guards, automated failure analysis, automated repair, automated reconfiguration

Joint work of: Pu, Calton; Johnson, Jim; de Lemos, Rogerio; Reuter, Andreas; Taylor, David; Zakiuddin, Irfan

Development of the Simpson's 4-slots algorithm

Jean-Raymond Abrial (ETH Zürich, CH)

The complete formal development of Simpson's 4-slots algorithms is presented by means of a number of refinements.

In an abstract initial model, the two participants in this algorithm, a writer and a reader, are defined by means of the properties of their traces.Their actions on these traces are performed at once.

The various refinements then consist in introducing more and more concurrency between the participants by cutting these initial actions into smaller atomic actions. The development has been entirely proved.

BPQL - A Query Language for Business Processes

Catriel Beeri (The Hebrew University of Jerusalem, IL)

This talk presents BPQL, a novel graphical query language for querying Business Processes, implemented as a set of cooperating web services. BPQL is based on an intuitive model of business processes, an abstraction of the emerging BPEL (Business Process Execution Language) standard. It allows users to query business processes visually, in a manner analogous to how such processes are typically specified, and can be employed in a distributed setting, where process components may be provided by distinct providers(peers).

The talk describes the query language as well as its underlying formal model. Special emphasis is given to the following subjects: (a) The analogy between the specification and querying of business processes, and (b) the use of graph grammars to represent potentially infinite query results by a finite and concise representation. The current implementation, using Active XML and Web services is briefly described.

Joint work of: Beeri, Catriel; Eyal, Anat; Kamenkovich, Simon; Milo, Tova

SOS is Good For You

Joey W. Coleman (University of Newcastle, GB)

A Structural Operational Semantics (SOS) model of a restricted language is presented along with the context in which it was developed. The language includes concurrency and allows for a very high degree of interference between parallel portions of a program in the language. In the larger context, this moel is used to extend a logical frame in a manner similar to that of Nipkow and Melham. With the Rely/Guarantee framework developed by Jones, this allows program developments to be directly justified in the combined logical frame.

Keywords: Structural operational semantics, rely/guarantee reasoning

Spheres of Control II

Charles T. Davies (San Luis Obispo, USA)

This talk describes transaction processing at the application level and assumes that the operating system or human is something on the other side of a wall that has a portal through which requests and responses may be propagated. It does not mention any specific programming language since that would be below the application functional level. The transactions are viewed as the application unit of process atomicity. The main topics covered are ad hoc transactions, audit: the act of verification, transaction creation and generation of transaction data.

Allocating Isolation Levels to Transactions

Alan Fekete (The University of Sydney, AU)

Serializability is a key property for executions of OLTP systems; without this, integrity constraints on the data can be violated due to concurrent activity.

Serializability can be guaranteed regardless of application logic, by using a serializable concurrency control mechanism such as strict two-phase locking (S2PL); however the reduction in concurrency from this is often too great, and so a DBMS offers the DBA the opportunity to use different concurrency control mechanisms for some transactions, if it is safe to do so.

However, little theory has existed to decide when it is safe! In this paper, we discuss the problem of taking a collection of transactions, and allocating each to run at an appropriate isolation level (and thus use a particular concurrency control mechanism), while still ensuring that every execution will be conflict serializable. When each transaction can use either S2PL, or snapshot isolation, we characterize exactly the acceptable allocations, and provide a simple graph-based algorithm which determines the weakest acceptable allocation.

Full Paper:
http://doi.acm.org/10.1145/1065167.1065193

See also: Proc ACM Conference on Principles of Database Systems, 2005, pp 206-215

Relaxed Currency Serializability for Mid-Tier Caching and Replication

Alan Fekete (The University of Sydney, AU)

Many applications, such as e-commerce, routinely use copies of data that are not in sync with the database due to heuristic caching strategies used to enhance performance. We study concurrency control for a transactional model that allows update transactions to read out-of-date copies. Each read operation carries a *freshness constraint* that specifies how fresh a copy must be in order to be read. We offer a definition of correctness for this model and present algorithms to ensure several of the most interesting freshness constraints. We outline a serializability-theoretic correctness proof and present the results of a detailed performance study.

Keywords: Lazy propagation, freshness constraint, drift constraint, limited divergence

Joint work of: Berstein, Philip; Fekete, Alan; Guo, Hongfei; Ramakrishnan, Raghu; Tarma, Pradeep

See also: Proc ACM Sigmod 2006, to appear

Atomic Exception Handling

Christof Fetzer (TU Dresden, D)

Exception and error handling is often buggy. To support this statement, I will describe some measurements regarding the atomicity of existing exception handling code. A simple atomic block that rolls back internal and external state on exception might be a simple and intuitive abstraction for exception handling. However, realistically programmers will most likely want to have some control over what is rolled back (internal and/or external state) and how to deal with external actions. This will require some less elegant and potentially more difficult mechanisms.

Keywords: Exception handling

The Case for Handling Inconsistency Caused by Errors

Wilhelm Hasselbring (Universität Oldenburg, D)

Errors are a fact of life. In requirements engineering, for instance, it is well accepted that we have to live with inconsistent specifications; we need measures to handle inconsistency caused by errors.

Database textbooks generally explain that integrity constraints should be satisfied at all times because they capture the set of all legal databases. Nevertheless, data inconsistency is a phenomenon that often occurs in practice. The most common reason for inconsistency is the need to integrate heterogeneous, independent data sources: different databases that are consistent by themselves can contain inconsistent information about the same real-world object. The conflicts are revealed only when the data is brought together in an integrated database. In such situations, it is of practical importance to know how to deal with violations of integrity constraints. In general, there is no single best way to restore consistency, leaving us with a multitude of possible repairs. Domain-specific approaches are required: in hospital information systems, for instance, data is often added, but only seldom changed. Delays in inserting data may cause incomplete, but not contradictory data. Fault-tolerance and self-healing/self-stabilizing systems address the problem of handling errors, i.e. inconsistent states. EAI patterns emphasize asynchronous updates. In Software Engineering, software architectures with redundancy for enabling fault tolerance are designed. Programming languages provide mechanisms for exception handling.

In this presentation, I'll discuss various issues of handling inconsistency, and some related topics that are investigated in our graduate school TrustSoft (http://trustsoft.uni-oldenburg.de).

Keywords: Inconsistency, Error handling, Trustworthy Software Systems

References:

[1] Hasselbring, W., Reussner, R.: Toward trustworthy software systems. IEEE Computer 39 (2006) 91-92

Atomicity in a timed world

Holger Hermanns (Universität des Saarlandes, D)

This talk discusses atomicity in the context of process algebra and operational semantics. After some general remarks, I plan to focus on the question how atomicity is – or is not – affected by real-time and by probabilistic phenomena.

Overview of Selected Research Topics at MSR Cambridge

Tony Hoare (Microsoft Research UK, GB)

Some Concurrency research at Cambridge England – by Tony Hoare

Byron Cook and Josh Berdine on termination detection.

Loops that go on forever are a menace, especially in kernel mode. A tool is under development to detect the risk so that the programmer can avert it. It has been applied successfully to several Windows device drivers, and may find wider application in this area.

Marc Shapiro and Viktor Vafeiadis on Rely/guarantee proofs.

The spread of multi-core architecture provides a strong motive for ingenious concurrent algorithms that exploit interleaving of individual store accesses from multiple threads, without using locks for concurrency control. Many such algorithms have been published by researchers, and are subject to formal verification. Rely and guarantee conditions play a role in concurrent programming proofs, similar to that of preconditions and postconditions in sequential programming.

Tim Harris and Simon Peyton Jones on Software Transactional Memory.

STM provides a high level language programmer (e.g., Java or Haskell) with a facility to declare that a block of code is a conditional critical region. The whole region will be executed atomically as if it were a single store access. It is translated into a transaction that runs optimistically without a lock, and either commits atomically or aborts if interference has been detected. This facility greatly facilitates the design of concurrent algorithms and their proofs, while maintaining most of the efficiency of fine-grained lock-free execution.

Georges Gonthier on the reader/writer lock in Windows Kernel.

The existing reader/writer lock in Windows kernel has been highly optimised for efficiency. It has been modelled abstractly in Leslie Lamport's logic TLA+, and successfully model-checked. An analysis of fairness properties has lead to suggestions for improving responsiveness.

Andrew Kennedy on the CLR Class Loader.

There are complex dynamic loading rules for Classes in languages such as C♮# and Java. Because of heavy disc loading, these must be implemented concurrently. Control of concurrency involves hundreds of locks; to avoid deadlock, these are governed by a strict order of acquisition.

Peter O'Hearn and Matthew Parkinson on Separation Logic.
Separation logic is a convenient way of writing assertions that implicitly claim ownership of the objects on which a thread is operating. A tool (Smallfoot) is under development to offer automatic proof of the necessary verification conditions.

Scoping Mechanism for Mobile Agent Systems

Alexei Iliasov (University of Newcastle, GB)

The presentation describes the scoping mechanism implemented by CAMA (Context-Aware Mobile Agents) .The CAMA framework is intended for developing mobile applications built using the agent paradigm. The framework provides a powerful set of abstractions along with a supporting middleware and adaptation layer that allow developers to address the main characteristics of mobile application: openness, fault tolerance, asynchronous and anonymous communication, as well as device and code mobility. CAMA ensures disciplined system development by enabling recursive system structuring using scope, agent, role, and location abstractions. It also provides fault tolerance through exception handling and agent coordination in an open dynamic environment. The framework is developed for constructing large mobile agent applications and provides an effective highly-scalable middleware.

Both the CAMA middleware and the adaptation layer are available as open source software.

Keywords: Mobile agents, Linda, Coordination

Joint work of: Iliasov, Alexei; Romanovsky, Alexander

AO Challenge - Implementing the ACID properties for Transactional Objects

Jörg Kienzle (McGill University - Montreal, CA)

This talk presents a challenge case study to the aspect-oriented community: ensuring the ACID properties (atomicity, consistency, isolation and durability) for transactional objects. We define a set of ten base aspects, each one providing a well-defined reusable functionality. The base aspects are simple, yet have complex dependencies among each other. We then show how these base aspects can be configured and composed in different ways to implement different concurrency control and recovery strategies. This composition is delicate: some aspects conflict with each other, others have to be reconfigured dynamically at run-time. We believe that this case study can serve as a benchmark for aspect-oriented software development, in particular for evaluating the expressivity of aspect-oriented programming languages, the performance of aspect-oriented programming environments, and the suitability of aspect-oriented modeling notations.

Recovery from *Bad* User Transactions

David Lomet (Microsoft Research, USA)

User written transaction code is responsible for the *C* in ACID transactions, i.e., taking the database from one consistent state to the next. However, user transactions can be flawed and lead to inconsistent (or invalid) states. Database systems usually correct invalid data using *point in time* recovery, a costly process that installs a backup and rolls it forward. The result is long outages and the *de-commit* of many valid transactions, which must then be re-submitted, frequently manually. We have implemented in our transaction-time database system a technique in which only data tainted by a flawed transaction and transactions dependent upon its updates are *removed*. This process identifies and quarantines tainted data despite the complication of determining transactions dependent on data written by the flawed transaction. A further property of our implementation is that no backup needs to be installed for this because the prior transaction-time states provide an online backup.

Joint work of: Lomet, David; Vagena, Zografoula; Barga, Roger

See also: 2006 SIGMOD Conference, Chicago, IL (June, 2006)

A Derivation of RDCSS

Stephan Merz (INRIA Lorraine & LORIA - Nancy, F)

Lock-free data structures have been proposed for implementing data structures that are concurrently accessed by multiple threads. They promise to avoid contention at mutexes that ensure single-thread access to the data structure, without having to manage fine-grained locks, which are error-prone. Several authors have suggested that the design of lock-free data structure is simplified by a multi-word compare-and-swap operation (CASN). In a paper at DISC 2002, Harris, Fraser, and Pratt have suggested an efficient implementation of CASN on top of single-word compare-and-swap (CAS1). In order to verify this algorithm, we specify a linearizable memory in TLA+, and prove the implementation to be a correct refinement of this high-level specification, refining atomicity. Automatic first-order SMT (satisfiability modulo theories) solvers appear to be particularly suited to the verification conditions generated by this case study. We hope to be able to completely automate this proof and to generalize this approach to similar problems.

Joint work of: Fejoz, Loic; Merz, Stephan

Atomicity and the Commutativity of Actions

Louise Moser (Univ. California - St. Barbara, USA)

Atomic transactions are one of the great successes of modern computing. They have allowed the construction of large complex applications of high reliability and have eased the programming of the applications. Considerable ingenuity has been applied to maintaining the atomicity of transactions as systems scale in size and concurrency, with sophisticated strategies for locking, ordering and optimism.

Unfortunately, atomicity becomes more difficult and more expensive as systems become more distributed. Recent attempts to extend transactions into distributed environments without incurring such costs, such as compensating transactions, have adversely affected the atomicity of distributed activities.

In many circumstances, much of the value of atomicity derives not from atomicity itself but from the separation of activities that is facilitated by atomicity. We propose a strategy for the separation of activities based on commutativity of actions rather than on atomicity of transactions. The operations of an activity on a data item (record) are preceded by a special action that imposes constraints on subsequent actions of that activity on that data item. The constraints suffice to ensure that those subsequent actions commute with the actions of other concurrent activities that operate on the same data item.

A corresponding concluding action follows the activity's operations on the data item. The preceding action and the concluding action can be strictly local to the data item. This strategy based on commutativity of actions is applicable to many common and useful activities.

Joint work of: Melliar-Smith, P. Michael; Moser, Louise

Beyond Harris and Fraser: Nested Transaction for Java

J. Eliot B. Moss (Univ. of Massachusetts - Amherst, USA)

Describes possible closed and open nesting constructs for Java, with discussion of hoped-for semantics and sketch of possible directions at formalizing the semantics. It is mostly intended to provoke discussion, refinement of concepts, and formalization.

Keywords: Open nesting, closed nesting, nested transactions, atomic actions, Java

Shades of Atomicity in Object-Oriented Programming

Arnd Poetzsch-Heffter (TU Kaiserslautern, D)

Modern object-oriented programming languages support multi-threading and provide locking mechanisms for synchronization. However, the underlying concurrency models are very complicated, mainly to enable compiler optimizations and to exploit parallel computer architectures. In order to understand and receive an expected behavior of a concurrent program, the application programmer has to ensure correct synchronization which is not easy to prove. Several higher-level language concepts are currently investigated to simplify multi-threaded object-oriented programming, in particular transaction mechanisms, atomic methods, and asynchronous methods.

Our goal is to combine such techniques with ownership structures for the heap to get more fine grained atomicity. Ownership techniques hierarchically structure the heap into encapsulated regions of objects that we call compound objects. We proposed two possible semantics for concurrent methods calls on compound objects. Weak compound object atomicity guarantees that multi-threading does not violate the consistency of compound objects, that is, their implementation invariants. It does not provide any serialization guarantees for calls going to other compound objects. In particular, if a thread T working in compound object C makes a call to another compound D, it might see a different state of C on return from D. With compound object atomicity, the second, stronger notion, a method call to C performs all actions within C atomically, but the actions of external calls need not be atomic.

Keywords: Concurrency, object-oriented programming, atomicity, ownership

A (Transaction-Inspired) Defence against TOCTTOU Attacks: The EDGI Approach

Calton Pu (Georgia Institute of Technology, USA)

We describe the TOCTTOU (Time-of-Check-To-Time-Of-Use) problem, characterized by a pair of file object access (check and use) by a victim process and simultaneously an attacker access to the same file object in-between the check/use steps. An abstract model of the TOCTTOU problem (called STEM) enumerates all the potential file system call pairs (called exploitable TOCTTOU pairs) that form the check/use steps. When applied to POSIX and Linux, STEM shows that Linux contains more than 200 ex-ploitable TOCTTOU pairs and POSIX more than 400. A detection framework and tools for Linux found previously unreported TOCTTOU vulnerabilities in often-used system utilities such as vi and emacs. A defense mechanism based on STEM, called EDGI, works by preventing file object creation and removal by other attacker processes during process execution. We discuss the influence of atomicity concept in the EDGI approach as an example of useful guaranteed execution properties that are weaker than atomicity.

Keywords: TOCTTOU, file system vulnerabilities

Joint work of: Pu, Calton; Wei, Jinpeng

Database interfaces for replication support

Luis Rodrigues (University of Lisboa, P)

Database replication is a key technology to offer highly available services. Two main architectural options have been followed in the past to implement database replication:

- The "in-core" approach consists in augmenting the database kernel with support for replication. This typically involves the implementation of specialized replication strategies that are hard to maintain as the database kernel evolves.

- The "middleware" approach implements the replication protocols at the middleware layers, using the underlying database as a black box. This often requires the middleware to reimplement several of the functionalities already implemented at the database level.

The GORDA project is a EU funded research project that intents to foster database replication as a means to address the challenges of trust, integration, performance, and cost in current database systems underlying the information society. As a step in this direction, it aims at defining standard interfaces, to be exported by database vendors, that allow replication protocols to be logically decoupled from the database kernel, regardless how they are bundled in the final product.

This talk will motivate the GORDA API currently under development and illustrate its use for different replication strategies.

Joint work of: Rodrigues, Luis; Pereira, José; Correia, Alfrânio; Carvalho, Nuno; Guedes, Susana; Oliveira, Rui

Dependable Systems Perspective (Atomic Manifesto) - a very brief introduction

Alexander Romanovsky (University of Newcastle, GB)

This brief talk introduces the issues related to dependability and atomicity as they were summarised in the Atomic Manifesto. The intention is to describe to the partially new audience the vision of the Atomicity I people. Dependability is understood as the reliance that can be justifiably placed on the system. The main means of achieving dependability are rigorous design, fault tolerance, verification and validation and system evaluation. It is clear that all the Seminar attendees work on dependability (probably without realising it) because the users are solely interested in system dependability. In the context of dependability atomicity has two main meanings: it ensures error confinement as well as the

atomic system results or the atomic views on system evolution. To design fault tolerant applications we need to use atomic actions as structuring units hiding and encapsulating system state and behaviour and by doing these confining errors. To ensuring consistent views of several distributed processes on system evolution/changes (including failures of various assumed types) the dependability community has been working on developing atomic commitment and atomic multicast protocols, which represent particular cases of the general consensus problem. It is proven that this problem does not have general solutions for asynchronous systems with single process failure. Component based development of dependable systems is greatly simplified if the components are atomic in a sense that they do not have side effects. Atomic component composition can rely on both component interfaces and specifications of component interactions.

Keywords: Dependability, atomicity, atomic actions, consensus problem, error confinement, component composition

Exception Handling and Atomicity

Alexander Romanovsky (University of Newcastle, GB)

This talk was prepared before the seminar but was not given. It briefly introduces exception handling as the most general means for achieving application-specific recovery and system structuring. Exception handling means are employed during application development: architecting, modelling, design, implementation. Exception handling can be successfully used when backward recovery is either not applicable or prohibitively runtime expensive (control systems, web services, virtual organisations, e-science/grid, mobile systems, and many more). Forward Not Backward! Exception handling scopes need to be atomic to ensure that all errors are confined and that only well-defined known outcomes are reported. These outcomes are ALL (normal outcome), NOTHING (abort exception Ea) or WELL-DEFINED SOMETHING (more exceptions E1, E2, etc.). The structuring units of system design are atomic actions (atomic operations, conversations, CA actions, ACID transactions, etc.). Exception handling being an application level mechanism can deal in a consistent way with weaker forms of atomicity: information smuggled in action, information smuggled out, runaway participants of atomic actions, belated action participants, external components which cannot guarantee ACID access, etc.

Keywords: Exception handling, atomic actions, exceptional outcomes, system structuring

Compensation handling

Alexander Romanovsky (University of Newcastle, GB)

How to help application programmers to write fault tolerant systems using exception handling, abort and compensation? We are offering programmers too many techniques for recovery. Error recovery is too complex and is known to be error prone. The mess needs cleaning up. We need to understand if these mechanisms work at different levels, deal with different faults, incur different cost, etc. And, more generally, how to use them together and what are the right combinations. It is clear that not everything can be or needs to be aborted or compensated for. We do not live in such a world, compensation is obviously not enough. Exception handling is the most general technique but in many situations we need to use compensation and/or abort. The choice of the right mixture of abort, compensation and exception handling is application specific.

Keywords: Application fault tolerance, exception handling, compensation, abort

Isolation Levels in Federated Environments

Ralf Schenkel (MPI für Informatik - Saarbrücken, D)

Atomicity and isolation of transactions are key requirements of advanced applications in federated systems consisting of distributed and heterogeneous components. While all existing federated systems support atomicity using the two-phase commit protocol, they lack support for federated concurrency control. Many possible solutions have been proposed in the literature, but they failed to make impact on real systems because they completely ignored the widely used concept of isolation levels, which offer optimization options to applications at the cost of less rigorous control over data consistency.

We discuss how global serializability can be guaranteed even in the presence of weaker isolation levels at the component systems. For the commonly used isolation level Snapshot Isolation, we present several algorithms for federated concurrency control. For situations in which weaker consistency is sufficient, we show how retricted isolation levels can be supported for global transactions.

Keywords: Isolation levels, snapshot isolation, federated databases

The HO model and dogmas in distributed fault tolerant computing

André Schiper (EPFL - Lausanne, CH)

We start by pointing out two dogmas that have appeared over the years in distributed fault tolerant computing. Then we show their drawbacks in the context of agreement problems (e.g., consensus), and we present a new computional model, the HO model, that frees us from these dogmas and unifies all benign faults. The advantage of the HO model is illustrated on consensus.

Keywords: Consensus, system model, fault models

Joint work of: Charron-Bost, Bernadette; Schiper, André

Towards Exploiting Meta-Programming for Web Services

Gottfried Vossen (Universität Münster, D)

The management of procedural data has regained interest in recent years, due to an increased exploitation of database concepts in novel applications such as the Web. We give a quick survey of several languages for "meta-programming" databases, i.e., query languages for databases containing queries. In particular, we discuss Meta-SQL, which assumes that stored queries are represented in XML, and that XSLT is used as a manipulation sublanguage. Since just a few features need to be added to SQL to turn it into a full meta-query language, we try to exploit the approach for specifying, executing, and (ultimately) composing Web services. We show our initial attempts in this direction by simulating relational transducers in "Service SQL."

Keywords: Meta-programming, Web services

From (Security) Protocol to Enterprise Network Infrastructure (Security) Analysis

Christoph Weidenbach (MPI für Informatik - Saarbrücken, D)

Atomicity violations in (security) protocols often result in attack opportunities. We propose first-order logic theorem proving as a paradigm where such vulnerabilities can be automatically detected.

Furthermore, the underlying first-order modelling can be extended from single protocol analysis to cope with an enterprise network infrastructure including routing, firewalling, and higher level services.

Formal Development of Distributed Transactions for Replicated Databases using Event B

Divakar Yadav (Univ. of Southampton, GB)

System availability is improved by the replication of data objects in a distributed database system. However, during updates, the complexity of keeping replicas identical arises due to failures of transactions. Event B is a formal technique which provides a framework for developing mathematical models of distributed systems by rigorous description of the problem, gradually introducing solutions in refinement steps, and verification of solutions by discharge of proof obligations.

In this talk, I present a formal development of a distributed system using Event B that ensures atomic commitment of distributed transactions consisting of communicating transaction components at participating sites. This formal approach carries the development of the system from an initial abstract specification of transactional updates on a one copy database to a detailed design containing replicated databases in refinement. Through refinement we verify that the design of the replicated database confirms to the one copy database abstraction.

Keywords: Formal Method, Transactions, Replicated Database, Event B

Joint work of: Butler, Michael; Yadav, Divakar

Architectural Reconfiguration using Coordinated Atomic Actions

Rogerio de Lemos (University of Kent, GB)

The provision of system services despite the presence of faults is known as fault tolerance. One of its associated activities is fault handling, which aims to prevent the reactivation of already located faults. System reconfiguration, one of the steps of fault handling, is a complex cooperative activity involving several participants, thus should be developed in a structured fashion. This position paper describes how coordinated atomic actions (CA actions) and exception handling can be applied to the architectural reconfiguration of systems.

Keywords: Fault tolerance, fault handling, dynamic reconfiguration, software architecture