Self-Stabilization

22.10 - 27.10.2000

organized by

Prof. Dr. A. Arora (Ohio State University, Columbus, OH, USA),
Prof. Dr. J. Beauquier (Universitè Paris Sud, Orsay, France),
Prof. Dr. S. Dolev (Ben Guroin University, Beer-Sheva, Israel),
Prof. Dr. T. Herman (University of Iowa, Iowa City, IA, USA),
Prof. Dr. W. P. de Roever (Universität Kiel, Kiel, Germany)

Distributed system research is at an interesting juncture. Many basic problems have been identified and intensively studied, often using traditional models of distributed computing and networks. However the nature of applications of distributed systems is changing, and new technological developments are challenging conventional system design. These trends encourage new approaches to problems of distributed control, fault recovery, and adaptive behavior.
Self-stabilization is now acknowledged as an important theme in distributed computing, with papers regularly appearing in prominent conferences. The importance derives from the fact that self-stabilization makes initialization independent from desired convergence in system computations. As a consequence of this fact, protocols and algorithms that are self-stabilizing can automatically adapt to dynamic environments and can recover from faults that put the system in arbitrary states. Many examples of algorithms use only local information to stabilize, and thus scale well to large distributed systems.
This report contains abstracts of presentations given at the second Dagstuhl Seminar on Self-Stabilization. Specialists in the area of stabilization research see new opportunities to apply stabilization to other areas of computer science and engineering, and this was a theme of the seminar. Results presented at the seminar included new directions: relating control theory ideas to self-stabilization; application to higher-level systems research and middleware reliability; application to data structures; group communication that is self-stabilizing; and stabilization in the model of mobile agent computing. In addition to the presentations described by the abstracts, there were several informal talks and open sessions with lively discussions of future directions. Some traditional issues provoked conversations on many topics: model conversions; atomicity and fairness implementations; impossibility results; heuristics and frameworks for proving the convergence of self-stabilizing algorithms; and experiments with new technical definitions of self-stabilization that could benefit more applications.

Once again, the pleasant atmosphere of Dagstuhl was an important prerequisite for the success of this seminar. We would like to thank all who contributed to it. Particularly, the Scientific Directorate for their encouragement to organize the seminar, the Schloß Dagstuhl Office for financial support and the excellent local arrangements, and the National Science Foundation for providing travel support to US participants.

The organizers

A. Arora
J. Beauquier
S. Dolev
T. Herman
W. P. de Roever

Contents

1 Stability of Long-lived Consensus

Shlomi Dolev
Ben Gurion University of the Negev, Beer Sheva, Israel

This work introduces the notion of stability for a long-lived consensus system. This notion reflects how sensitive to changes the decisions of the system are, from one invocation of the consensus algorithm to the next, with respect to input changes. Stable long-lived consensus systems are proposed, and tight lower bounds on the achievable stability are proved, for several different scenarios. The scenarios include systems that keep memory from one invocation of consensus to the next versus memoryless systems; systems that take their decisions based on the number of different inputs but not on the source identities of those inputs versus non-symmetric systems. These results intend to study essential aspects of stability, and hence are independent of specific models of distributed computing. Applications to particular asynchronous and synchronous systems are described.

Joint work with Sergio Rajsbaum.

2 Communication Adaptive Self-Stabilizing Group Communication

Shlomi Dolev and Elad Schiller
Ben-Gurion University of the Negev, Beer-Sheva, Israel

This paper presents the first (randomized) algorithm for implementing self-stabilizing group communication services in an asynchronous system. Our algorithm converges fast to a legal behavior and is communication adaptive. Namely, the communication volume is high when the system recovers from the occurrence of faults and is low once a legal state is reached. The communication adaptability is achieved by a new technique that combines transient fault detectors.

3 Time and Space Optimality of Distributed Depth-First Token Circulation Algorithms

Franck Petit
LaRIA, University of Picardie Jules Verne, Amiens, France

We address the depth-first token circulation problem (DFTC) in this work. DFTC is to implement a token circulation scheme where the token is passed from one processor to another in the depth-first order such that every processor gets the token at least once in every token circulation cycle.
We first consider rooted tree networks where every processor knows which of its neighbors leads to a particular processor called the root. On such trees, we propose a state optimal DFTC algorithm—each processor p requires $D_p + 1$ states (proven in [1]), where D_p is the degree of p. Next, we propose a second algorithm, also for trees, but where no processor knows which of its neighbor leads to the root. This algorithm is also optimal in terms of the number of states per processor ($D_p + 2$ states [2]). Furthermore, both algorithms are *snap-stabilizing* [1]. A *snap-stabilizing protocol* guarantees that the system always maintains the desirable behavior. In other words, a snap-stabilizing algorithm is also a self-stabilizing algorithm which stabilizes in 0 steps. Obviously, any snap-stabilizing protocol is optimal in terms of the stabilization time. Therefore, they are state and time optimal, even without considering the property of self-stabilization.

References

[1] A. Bui, A. K. Datta, F. Petit, and V. Villain. State-optimal snap-stabilizing PIF in tree networks. In Proceedings of the Fourth Workshop on Self-Stabilizing Systems. IEEE Computer Society Press, 78–85, 1999.

[2] F. Petit and V. Villain. Optimality and self-stabilization in rooted networks. Parallel Processing Letters, 10(1), 3–14, 2000.

Joint work with Vincent Villain, LaRIA, University de Picardie Jules Verne, Amiens, France.
Please refer to `http://www.laria.u-picardie.fr/~petit/publi/dimacs_dftc.ps.gz`

4 Self-Stabilization Aspects of Rollback Recovery

Augusto Ciuffoletti
Universita' di Pisa, Pisa, Italy

I introduce endo-stabilization as a tool to approach the design of distributed algorithms that take action even in the absence of complete information.
I present a case study related to the problem of backward recovery, as composed of three sub-problems: checkpointing, rollback, and disposal of obsolete checkpoints.
A simple session-level algorithm is introduced to gather information about the state of the system, in the form of a timestamp that advances consistently on each unit in the system.
I show that this information is sufficient to control the checkpointing protocol, but is not sufficient to control rollback and disposal.
In the case of rollback, we can apply to the endo-stabilization concept to restore consistency using a lazy algorithm.
The disposal operation appears to be intractable using endo-stabilization, since the intermediate behavior of the system can be unsafe.

5 On the Benefits of Modeling Self-Stabilizing Algorithms as Variable Structure Feedback Systems

Oliver Theel
Darmstadt University of Technology, Darmstadt, Germany

The self-stabilizing property of certain distributed algorithms exhibits interesting analogies to stable feedback systems used in various engineering domains, like electrical or mechanical engineering. Informally, a feedback system is stable, if after a certain finite period of time, the system reaches and remains in a pre-defined state. Contrary to the self-stabilization research domain, which is a rather new area of research in computer science, control theory in the engineering domain has a century-old background and offers a broad theoretical foundation with powerful criteria for reasoning about the stability of feedback systems. In this talk, we show how to model a distributed algorithm implemented an a set of processes executing guarded commands as an instance of a discrete-time variable structure system model. Based on this modeling, we prove convergence (and closure) of the sample algorithm by means of Ljapunov's "Second Method." The talk closes with remarks on how Ljapunov Theory can furthermore be used to derive upper and lower bounds for the convergence speed as well as how to automatically identify variant functions for a certain subclass of self-stabilizing algorithms.

6 A Theory of Composition

I.S.W.B. Prasetya and S.D. Swierstra
Utrecht University, Utrecht, The Netherlands

A theory of composition is a theory that allows us to decompose a property of a system into local properties of its component. Because local properties can be verified locally, this can save us some proof steps. Another advantage is that a component is guaranteed to behave correctly in system without having to fix its partner components. This optimizes the component's reusability. The composition of progress properties is however believed to be problematical. There have been many approaches, but for one reason or another they are unsatisfactory. Our approach is based on the idea of temporary interference and non-interference. It is a simple idea, and it works very well. The whole theory is based on UNITY and has already been mechanically verified. Two more specialized composition theories derived from the main theory is in the field of self stabilization and fault tolerance.
Note: Best reference to the work is available in the form of a paper available at request (`wishnu@cs.uu.nl`). The paper. "Factorizing Fault Tolerance" is scheduled to appear in the coming Journal of Theoretical Computer Science, special edition on Fault Tolerance.

7 Stabilization in Device Networks

Anish Arora
Ohio State University, Columbus, OH, USA

Dependability of applications involving access and control of diverse devices over potentially different networks may be achieved via system stabilization. This talk describes Aladdin, a stabilizing device networking system developed for the home environment. Abstractions such as Soft-State Storage and Model-Based Detection/Correction are outlined that enable stabilization in Aladdin. Experiences in the difficulty of validating its stabilization and in measuring its overall dependability are presented, to motivate a new direction for research in stabilization. Lastly a Dagstuhl Manifesto is laid out to challenge researchers to address pragmatic issues in stabilization.

8 A Space Optimal, Deterministic Self-Stabilizing Leader Election Algorithm for Unidirectional Rings

Faith E. Fich
University of Toronto, Toronto, Canada

A deterministic, self-stabilizing, leader election algorithm for unidirectional rings is presented that uses $10n$ states per processor. This improves the number of states per processor of previously known algorithms and matches, to within a small constant factor, a lower bound of n.
The approach first develops an algorithm that stabilizes for a restricted class of schedules, the alternating schedules, where between successive steps of each processor there is exactly one step of each of its neighbors. Then this algorithm is combined with the deterministic token algorithm to ensure stabilization for all schedules.

This work is joint with Colette Johnen.

9 Self-Stabilizing Minimum Spanning Tree Construction on Message-Passing Networks

Lisa Higham
University of Calgary, Calgary, Canada

Self-stabilizing algorithms for constructing a spanning tree of an arbitrary network have been studied for many models of distributed networks including those that communicate via registers (either composite or read/write atomic) and those that employ message-passing. In contrast, much less has been done for the corresponding minimum spanning tree problem. In particular, we present the first self-stabilizing algorithm for minimum spanning tree for deterministic, synchronous message-passing networks of known size, and discuss possible extensions to asynchronous ones (with time-outs) and to ones of unknown size.

10 Universal Dynamic Synchronous Self-Stabilization

Paolo Boldi and Sebastiano Vigna
Università di Milano, Milano, Italy

We prove the existence of a "universal" synchronous self-stabilizing protocol, that is, a protocol that allows a distributed system to stabilize to a desired nonreactive behavior (as long as a protocol stabilizing to that behavior exists). Previous proposals required drastic increases in asymmetry and knowledge to work, whereas our protocol does not use any additional knowledge, and does not require more symmetry-breaking conditions than available; thus, it is also stabilizing with respect to dynamic changes in the topology. We prove an optimal quiescence time $n + D$ for a synchronous network of n processors and diameter D; the protocol can be made finite state with a negligible loss in quiescence time. Moreover, an optimal $D + 1$ protocol is given for the case of unique identifiers. Finally, we highlight the intimate connection between self-stabilizing and anonymous computations showing how to turn any anonymous algorithm into a self-stabilizing protocol.

11 Defining Redundancy: In Search of the Holy Grail

Felix Gärtner
TU Darmstadt, Darmstadt, Germany

Redundancy is a key concept in fault-tolerance, however it still lacks a formal definition. In this work we explore the notions of redundancy which are inherent in the fault-tolerance theory of Arora and Kulkarni. In this theory, every fault-tolerance mechanism can be constructed from two simple components: detectors and correctors. Detectors are necessary and sufficient to maintain safety specifications, while correctors are necessary and sufficient to eventually satisfy safety specifications. We investigate the assumptions of this theory and discover that fault-tolerance components contain either non-reachable states or non-reachable transitions. This gives rise to definitions of redundancy in space and redundancy in time. This is very much unfinished work, not all theorems have been proven yet, and the definitions are somewhat orthogonal to the understanding of the terms in the literature. Because of this and several other reasons, this work might never be finished. Thus, it resembles – at least to the authors – a search for the holy grail.

12 Compositional Reasoning about (Shared-Variable) Concurrency

Willem-Paul de Roever
Christian-Albrechts-Universität zu Kiel, Kiel, Germany

- The notion of compositionality is explained from the viewpoint of the verify-while-develop paradigm.
- The advantages of compositional reasoning about concurrent programs are discussed.
- The Assume-Guarantee method for specifying and reasoning about concurrent (and distributed) programs is introduced.

- The interpretation of this paradigm is investigated for shared-variable concurrency: Should one use reactive sequences or so-called Aczel traces? We give a reactive sequences interpretation which leads to an unsound parallel composition rule. The Aczel-traces interpretation is proved to be sound and complete.

13 Questions about Snap-Stabilization

Vincent Villain
LaRIA (Laboratoire de Recherche en Informatique d'Amiens), UPJV (Universite de Picardie Jules Verne), Amiens, France

Snap-stabilization has been introduced in 1999 by Bui, Datta, Petit, and Villain. A snap-stabilizing protocol guarantees that, starting from an arbitrary system configuration, the protocol always behaves according to its specification. So, a snap-stabilizing protocol is a time optimal self-stabilizing protocol (stabilizes in 0 rounds). We present snap-stabilizing wave algorithms on particular topologies (rings, chains, and trees). We show that any single initiator application using such protocols guarantees that the first computation starting after the occurrence of a fault will give the expected result. We then present some open questions.

- Is it possible to design snap-stabilizing wave algorithms on general graphs?
- More generally, can we characterize the class of problems which admit a snap-stabilizing protocol?

14 Self-Stabilizing Local Mutual Exclusion and Daemon Refinement

Ajoy K. Datta
University of Nevada, Las Vegas, NV, USA

Refining self-stabilizing algorithms which use tighter scheduling constraints (weaker daemon) into corresponding algorithms for weaker or no scheduling constraints (stronger daemon), while preserving the stabilization property, is useful and challenging. Designing transformation techniques for these refinements has been the subject of serious investigations in recent years. This paper proposes a transformation technique to achieve the above task. The core of the transformer is a self-stabilizing local mutual exclusion algorithm. The local mutual exclusion problem is to grant a process the privilege to enter the critical section if and only if none of the neighbors of the process has the privilege. The contribution of this paper is twofold. First, we present a bounded-memory self-stabilizing local mutual exclusion algorithm for arbitrary network, assuming any arbitrary daemon. After stabilization, this algorithm maintains a bound on the service time (the delay between two successive executions of the critical section by a particular process). This bound is $(n\times(n-1))/2$ where n is the network size. Second, we use the local mutual exclusion algorithm to design two scheduler transformers which convert the algorithms working under a weaker daemon to ones which work under the distributed, arbitrary (or unfair) daemon, both transformers preserving the self-stabilizing property. The first transformer refines algorithms written under the central daemon, while the second transformer refines algorithms designed for the k-fair $(k \geq (n-1))$ daemon.

Co-authors: Joffroy Beauquier, Maria Gradinariu, Frederic Magniette

15 A Universal Self-Stabilizing Mutual Exclusion Algorithm

Hirotsugu Kakugawa and Masafumi Yamashita
Kyushu University, Fukuoka, Japan

A distributed algorithm is said to be self-stabilizing if it converges to a correct state from any initial state. In this talk, we characterize the class of networks on which there is a self-stabilizing mutual exclusion algorithm, and present a universal self-stabilizing mutual exclusion algorithm for the class in the sense that the algorithm works as a self-stabilizing mutual exclusion algorithm on each of the networks in the class.

16 Composite Available and Stabilizing Data Structures

Ted Herman
University of Iowa, Iowa City, IA, USA

A data structure is called available if the effect of any operation on the structure is consistent with its response even if internal variables of the structure have arbitrarily invalid values. For example, if an insert(x) operation's response is ack, then the operation inserted x into the structure; if the response is full, then x was not inserted into the structure. A data structure is called stabilizing if, for any arbitrary (and possibly illegal) initial state, any sequence of sufficiently many operations brings the data structure to a legal state. Availability also constrains the time complexity of operations: even in an illegal state, the response time for an operation is bounded by the operation time for some legal state; typically this bound is the worst-case operation time for a full data structure. After the structure has stabilized to a legal state, all operations have normal running times and responses.
This work constructs an available stabilizing data structure made from two constituents, a heap and a search tree. These constituents are themselves available and stabilizing data structures described in previous papers. Each item of the composite data structure is a pair (key,value), which allows items to be removed by either minimum value (via the heap) or by key (via the search tree) in logarithmic time. This is the first research to address the problem of constructing larger data structures from smaller ones that have desired availability and stabilization properties. The research has negative and positive results: it is impossible to construct a heap-search-tree composite without relaxing requirements on at least one of the data structure's operations; and on the positive side, with a suitable relaxation of one operation requirement, an available and stabilizing composite data structure is obtained using a mark-and-sweep technique.

17 A Framework for Constructing Self-Stabilizing Systems

Luc Onana Alima
Royal Institute of Technology, Kista, Sweden

The ability of a self-stabilizing system to tolerate any number and any type of transient faults makes the concept of self-stabilization very attractive in distributed systems. However, designing and proving correct a self-stabilizing system can be a hard task. We have to deal with the proof of convergence for each design. To alleviate this complication, we propose a simple and powerful framework that can be used to construct self-stabilizing systems in a systematic manner. By simple we mean a tool that is easy to understand and thus to use, and by powerful we mean a tool that can be used to generate a great variety of self-stabilizing systems.
The proposed framework is based on:

1. The assumption that the network's topology is a rooted tree.

2. A scheme for generating infinite sequence of waves in a self-stabilizing manner on a rooted tree system;

3. Two concepts we introduce: pyramidal and pseudo-pyramidal functions. We regard a problem to be solved on a distributed system as a function that takes a multi-set of input values (one per node) and returns a multi-set of output values (one per node). Then we distinguish two important classes of functions:

   (a) The class of pyramidal functions, which contains any function that can be computed on a rooted tree in two phases as follows: from the leaves towards the root, each node performs the same calculation, which is based on the values received from its children (if any) and its own input value. At the root, the final result (the output value at each node) is computed by performing the same calculation and possibly an additional calculation. Then, the root starts the second phase that consists in broadcasting the output value which is the same for each node of the system.

   (b) The class of pseudo-pyramidal functions, which contains any function that can be computed on a rooted tree in two phases as in the case of pyramidal functions except that during the second phase, each node, depending on the specific value received from its parent (if any), computes its own output and produces a specific value for each of its children.

4. Two theorems that we call Tree Embedding Theorems. The minor theorem gives a generic algorithm for computing any pyramidal function in a self-stabilizing manner. The major theorem gives a generic algorithm to compute any pseudo-pyramidal function in a self-stabilizing manner.

We demonstrate the viability of our framework by constructing self-stabilizing systems for a significant number of tasks including: counting/topology updated, distributed reset, distributed selection, determining an upper-bound of the system diameter, ranking, distributed sorting, load-balancing, max/min-heap maintenance.
The main benefits of the proposed framework are: (i) it releases the designer from the proof of convergence. He/She only needs to design correct non-self-stabilizing solutions (that is, finding pyramidal or pseudo-pyramidal functions). (ii) The stabilization time of the resulting systems is $O(h)$ rounds, where h is the height of the tree. However, for some functions, the memory required may be $O(n)$, where n is the system size.

18 Stabilizing Agents

Sukumar Ghosh
University of Iowa, Iowa City, IA, U.S.A.

This paper illustrates the use of mobile agents in the stabilization of distributed systems. The goal is to build stabilizing systems on the Internet, where the component processes are not under the control of a single administration. Two examples are presented to illustrate the idea: the first is that of mutual exclusion on a unidirectional ring, and the second deals with the construction of a DFS spanning tree.

19 The Convergence Theorem

Mohamed G. Gouda and Chin-Tser Huang
The University of Texas at Austin, Austin, TX, U.S.A.

To prove the convergence of a system, according to the theorem of convergence, it is sufficient to exhibit a ranking function whose value decreases by each action execution in that system (until the system convergence is completed). Unfortunately, the theorem of convergence does not hint on how to construct the required ranking function. To solve this problem, we present an interesting variation of

the convergence theorem. In this variation, the required ranking function can be constructed based on a partitioning of the system actions, or based on some decomposition of the system actions followed by a partitioning of the decomposed actions.

20 Formal Framework to Prove Probabilistic Self-Stabilizing Algorithms

Colette Johnen
Université Paris-Sud, Orsay, France

We give a strong motivation for the design of a formal framework to analyze probabilistic self-stabilizing algorithms.
An important issue is to make a clear distinction between what is non-deterministic (the scheduler behavior) and what is probabilistic (the output of the transition).
Also, our model can analyze a self-stabilizing algorithm under k-bounded, fair, weakly fair, or unfair schedulers.
We propose a framework based on scheduler strategies: a probabilistic self-stabilizing algorithm designer has to analyze all the scheduler strategies. If on any strategy of the scheduler, "most of the computations converge" then the algorithm is self-stabilizing under the studied scheduler. We give a formal definition of a strategy, and we have also presented a probabilistic space on top of the strategy structure that is suitable to analyze an algorithm.
We finally present an useful formal tool to prove the converge. If there is a pattern in any strategy such that "on any computation of the strategy, a legitimate state can be reached in least than n steps with a probability greater than E" then the probability of the computations set that converge is 1, on any strategy.
This model is formally presented in the research report no. 1225 of LRI "Randomized Self-Stabilizing and Space Optimal Leader Election under Arbitrary Scheduler on Rings," Joffroy Beauquier, Maria Gardinariu, Colette Johnen.