Dagstuhl Seminar 01451 on Exploration of Large State Spaces

November 04 - November 09, 2001

Organizers by
Tom L. Dean (Brown University)
Bernhard Nebel (Freiburg University)
Moshe Y. Vardi (Rice University)

Last Revision April 12, 2002

1 Introduction

1.1 Description

The goal of this workshop is to bring together researchers working on state-exploration methods in artifical intelligence (AI) and in automated verification (AV). The idea of organizing such a workshop came during the DIMACS/IRCS Tutorial and Workshop on Logic and Cognitive Science, which was held in April 1999 at the University of Philadelphia. We realized there that state-exploration is a major issue in computer-aided verification and in artificial intelligence.

1.2 Automated Verification

Automated verification provides a new approach to validating the correct behavior of software and hardware designs. In traditional design validation, "confidence" in the design is the result of running a large number of test cases through the design. Automated verification, in contrast, uses mathematical techniques to check the entire state space of the design for conformance to some specified behavior. Thus, while simulation is open-ended and fraught with uncertainty, formal verification is definitive and eliminates uncertainty. Over the last few years, automated verification tools, such as model checkers, have shown their ability to provide thorough analysis of reasonably complex designs. Companies such as AT&T, Cadence, Fujitsu, HP, IBM, Intel, Motorola, NEC, SGI, Siemens, and Sun are using model checkers increasingly on their own designs to ensure outstanding product quality. Unfortunately, model checking suffers from a fundamental problem known as state-explosion: the ability to handle only systems with limited-size state spaces. This explosion arises mainly because the transition system analyzed describes the global behavior of the system. In a system comprised of multiple components, the global state space is a cross product of the individual component state spaces. Even a system containing only small components can therefore yield a large global state-space. For a modern hardware and software, the global state space is prohibitively large. This poses a serious challenge to extant model checkers. Combating state-explosion is therefore a complex computational problem of critical importance.

1.3 Markov Decision Processes

Work on Markov decision processes (MDPs) dates back to the 1950's and there is a large literature stemming primarily from the fields of operations research and adaptive control. In the 1980's there was renewed interest in MDPs coming from work in automated planning in artificial intelligence and the work on binary decision diagrams from the verification community. Dean and Kanazawa [1989] and Tatman and Shachter [1990] introduced the idea of structured Markov processes and suggested how they might be used for representing and solving planning and control problems with very large state and action spaces. In subsequent years, a great deal of progress was made exploring structured versions of earlier, unstructured algorithms from the operations research and adaptive control communities.

True MDPs, i.e., problems in which the current state of the system is completely observable to the decision maker, are rare in practice and hence the partially observable variant (POMDP) is of great importance. Recently there has been a resurgence of interest in POMDPs, partially spurred by the success of Cassandra, Kaelbling and Littman, and a host of new algorithms have been developed, including variational methods which open up the possibility of solving a wide range of problems. Variational statistical methods can in some cases reduce the need for state exploration using a combination of sampling techniques and reformulation in terms of a (continuous) parameterized space of actions.

More recently researchers have begun using the basic technology for model checking including binary decision diagrams (BDDs), various algebraic extensions, and quantified variants that make use of modal logics of time to tackle a wider class of MDPs and their partially observable counterparts. In all cases, the need for exploring very large state and action spaces is dealt with by identifying and then exploiting structure in the combinatorial space of states and actions.

1.4 AI Planning

Planning is a sub-field of AI which is concerned with the generation of a rational course of action given a declarative specification of the environment, the goals, and the possible actions. In the case of "classical planning," one usually makes the simplifying assumptions that everything (that is relevant) is observable, that all actions are deterministic, and that the only change in the world is caused by actions of the agent. While this seems like a trivialization of the MDP problem, the problem is still computationally very hard because of the large state space one has to explore. The last five years brought considerable progress on the algorithmic side. In particular, the planning-graph approach by Blum and Furst [1995], which outperformed any existing planning system then, led to a flurry of further developments, such as planners that reduce the planning problem to a sequence of propositional satisfiability problems [Kautz and Selman 1996] or nonmonotonic reasoning problems [Dimopoulos at al 1997]. Type inference algorithms, the derivations of invariants and other techniques were used to prune the search space and to speed up planning, and OBDDs were used to code sets of states in a compact way. All these developments led to an impressive performance of AI planning systems as demonstrated at the first international planning competition in 1997 (at AIPS'97). Just to give an example, Kautz and Selman's BLACKBOX planner produced a 100-action plan in a world with approx. 10^{16} states in a few minutes. This has to be compared with the state of the art in the beginning of the nineties, when a 10-action plan needed a few hours to be generated. Currently, the main focus of research is in extending the existing techniques to handle more expressive planning formalism (e.g., including resources), in improving the core search algorithms, and in exploiting all types of knowledge that can be derived from the problem specification in order to prune the search space.

1.5 Benefits

We believe that the workshop will help to increase the awareness of the researchers working in one field of the problems and methods in the others and thus to increase the interaction and collaboration of the two fields, and the transfer of methodologies from one field to another.

2 Final Program

Monday, Nov. 5, 2001

07:30-08:45	breakfast
08:45-09:00	Welcome and Introductions
09:00-10:00	David Dill (Stanford University) Verification Tutorial: Explicit Search
10:00-11:00	Fabio Somenzi (University of Colorado) Verification Tutorial: Implicit Search
11:00-11:30	coffee break
11:30-12:15	Wolfgang Thomas (RWTH AAchen) Algorithmic Results in Infinite Automata Theory
12:15-14:00	lunch break
14:00-14:40	Prasad Sistla (University of Illinois, Chicago) Symmetry Reductions
14:40-15:20	David Basin (Freiburg University) Exploration of Large State Spaces using Monadic Logics, Automata, BDDs, and SAT Solvers
15:20-16:00	Javier Esparza (Edinburgh University) An Unfolding Approach to exploring State Spaces of Concurrent Systems
16:00-16:30	coffee break
16:30-17:15	Pierre Wolper (University of Liege) Exploring Infinite State Spaces with Finite Automata
17:15-18:00	Orna Kupferman (Hebrew University) Exploring Infinite State Spaces with Tree Automata
18:00	dinner

Tuesday, Nov. 6, 2001

07:30-09:00	breakfast
09:00-10:00	Fahiem Bacchus (University of Toronto) AI Planning Tutorial: Part I
10:00-10:15	coffee break
10:15-11:15	Fahiem Bacchus (University of Toronto) AI Planning Tutorial: Part II
11:15-11:30	coffee break
11:30-12:15	Brian C. Williams (MIT) A Hierarchical Approach to Model-based Reactive Planning in Large State Spaces
12:15-14:00	lunch break
14:00-14:40	Maria Fox (University of Durham) Using Symmetry Elimination to assist in Search Control in Planning
14:40-15:20	Ulrich Scholz (TU Darmstadt) Reducing Planning Problems by Path Analysis
15:20-16:00	Jörg Hoffmann (Freiburg University) Utilizing Problem Structure in Local Search: The Planning Benchmarks as a Case Study
16:00-16:30	coffee break
16:30-17:15	Enrico Guinchiglia (University of Genoa) SAT-Based Conformant Planning
17:15-18:00	Alessandro Cimatti (IRST, Trento) Planning under Limited Observability as Search in the Space of BDDs
18:00	dinner

Wednesday, Nov. 7, 2001

07:30-09:00	breakfast
09:00-10:00	Robert Givan (Purdue University) + Ron Parr (Duke University) MDP Tutorial: Part I and MDP Tutorial: Part II
10:00-10:15	coffee break
10:15-11:15	Ron Parr (Duke University) + Robert Givan (Purdue University) MDP Tutorial: Part III and MDP Tutorial: Part IV
11:15-11:30	coffee break
11:30-12:15	Jürgen Schmidhuber (IDSIA) Optimal Search
12:15-14:00	lunch break
14:00-18:00	free afternoon
18:00	dinner

Thursday, Nov. 8, 2001

07:30-09:00	breakfast
09:00-09:45	Charles Pecheur (NASA) Symbolic Model Checking of Domain Models for Autonomous Spacecrafts
09:45-10:30	Christian Stangier (Univ. Trier) Hierachical Image Computation and Dynamic Conjunction Scheduling in Symbolic Model Checking
10:30-11:00	coffee break
11:00-11:45	Malte Helmert (University Freiburg) Decidability and Undecidability Results for Planning with Numerical State Variables
11:45-12:30	Jana Köhler (IBM) What is Large? Searching State Space under Real-Time Requirements
12:30-14:00	lunch break
14:00-14:40	Robert Givan (Purdue University) Sampling Techniques for Large MDP Problems
14:40-15:20	Ron Parr (Duke University) Projection Methods
15:20-16:00	Tom Dean (Brown University) Minimizing Approximate Models versus Approximating Minimal Models
16:00-16:30	coffee break
16:30-17:15	Thomas Wilke (Univ. Kiel) Reducing the State Space of Buchi Automata using Simulation Relations
17:15-18:00	Colin Stirling (Edinburgh University) Decidability of Language Equivalence for DPDA
18:00	dinner
20:00-21:30	Panel: Fahiem Baachus, David Dill, and Robert Givan What Did We Learn?

Friday, Nov. 9, 2001

07:30-09:00	breakfast
09:00-09:45	Stefan Leue (Freiburg University) Explicit-State Search
09:45-10:30	Stefan Edelkamp (Freiburg University) Common Exploration Techniques in Search, Planning and Model Checking
10:30-11:00	coffee break
11:00-11:45	Richard Korf (UCLA) Space-Efficient Search
11:45-12:30	Moshe Y. Vardi (Rice University) Benefits of Bounded Model Checking in an Industrial Setting
12:15-14:00	lunch break

3 Abstract of Presentations

3.1 Verification Tutorial: Explicit Search

David Dill (Stanford University)

Automatic formal verification of finite-state concurrent and reactive systems depends on characterizing the reachable states of the system. There are two general approaches to this problem: explicit methods store the states individually (e.g., in a hash table), while symbolic methods use data structures such as Binary Decision Diagrams to store an implicit representation of the state space.

This talk is an overview of explicit methods. It begins with a discussion of how they are used to verify properties and the basic search algorithms. Then we describe two state reduction techniques, which reduce the number of states that need to be searcheed: symmetry reduction and partial order reduction. Finally, several other methods of reducing space and time requirements are discussed: hash compaction, storing states on disk, and parallel search.

3.2 Verification Tutorial: Implicit Search

Fabio Somenzi (University of Colorado)

In this tutorial, we review the techniques that contribute to the efficient implementation of symbolic model checkers. We start with a review of the properties of binary decision diagrams (BDDs) that are of interest. We then briefly review the temporal logic CTL*, which is used to describe properties, and summarize the algorithm for CTL* model checking, showing that it requires the ability to compute fixpoints of monotonic functions over finite lattices.

These functions evaluate the set of predecessors and successors of a set of vertices of a graph. Therefore, we examine the computation of images and pre-images for graphs that are symbolically represented by a set of implicitly conjoined predicates. In particular, we discuss quantification/conjunction scheduling, and the combination of conjunction and splitting.

We then consider the efficient computation of fixpoints. Symbolic algorithms usually resort to breadth-first exploration of the the state graph. However, this often leads to large intermediate BDDs. Several techniques that mix breadth-first and depth-first exploration can be used to ameliorate the problem. We then discuss the use of approximations in a refinement/based approach to model checking.

The detection of cycles satisfying fairness constraints is another important task of a model checker, for which two families of algorithms can be employed: SCC-hull and SCC enumeration algorithms. We review the main features of both. We also discuss the notion of strength of an automaton, and how it can be used to simplify cycle detection.

Finally, we look at various ways in which don't care conditions can be used to simplify model checking. These include the simplification of the transition relation with respect to the reachable states, the so-called ”forward model checking” algorithm, and the detection of dependencies among state variables.

3.3 Algorithmic Results in Infinite Automata Theory

Wolfgang Thomas (RWTH AAchen)

Infinite automata are of interest not only in the verification of systems with infinite state spaces, but also as a natural (and so far underdeveloped) framework for the study of formal languages. In this survey talk, we discuss some basic types of infinite automata, which are based on the so-called prefix-recognizable, synchronized rational, and rational transition graphs, respectively. We present characterizations of these transition graphs (due to Muller/Schupp and to Caucal and students), mention results on their power to recognize languages, and discuss the status of central algorithmic problems (like reachability of given states, or decidability of the first-order theory). A paper titled ”A short introduction to infinite automata” is avalaible under `http://www-i7.informatik.rwth-aachen.de/~thomas/publications.html`

3.4 Symmetry Reductions

Prasad Sistla (University of Illinois, Chicago)

The talk presented three different methods, based on symmetry reductions, in containing the state explosion problem in model checking. The first method considers the symmetries in the program as well the formula. In this method we first construct a quotient structure, corresponding to the reachable part of the global state graph, and then check the satisfaction of the formula in the quotient structure using the traditional model checking algorithms. This method is primarily useful in checking safety properties.

The second method considers only the symmetries in the program and is based on the construction of Annotated Quotient Structure (AQS). The AQS is like the quotient structure excepting that the edges carry additional information. This additional information is used for checking correctness under fairness. This method allows checking of both safety and liveness properties. The third method employs Guarded Quotient Structures (GQS). This method can be employed for employing symmetry reductions in systems that are almost symmetric.

The talk also presented a Symmetry based Model Checker, called SMC. It is an on-the-fly model checker that employs symmetry reductions and checks for correctness under a variety of fairness conditions. SMC permits the user to invoke two different types of symmetry reductions— process symmetry and state symmetry. It also allows the user to specify the type of fairness that needs to be invoked— weak fairness, strong fairness. It also allows the user to specify different types of on-the-fly options that need to be invoked.

3.5 Exploration of Large State Spaces using Monadic Logics, Automata, BDDs, and SAT Solvers

David Basin (Freiburg University)

I survey recent work on using monadic logic to represent verification and search problems. I show how, when one choose the right monadic logic there are different possibilities for problem solving. In particular, automata theoretic techniques (realized using BDDs) can be used to construct a representation of the entire solution space or, alternatively, satisfiability procedures can be used to find individual solutions. Paper on topic:
`http://www.informatik.uni-freiburg.de/~basin/pubs/cavBMC00.ps.Z`
(Joint work with Abdelwaheb Ayari, University of Freiburg)

3.6 An Unfolding Approach to exploring State Spaces of Concurrent Systems

Javier Esparza (Edinburgh University)

The automatic verification of finite state systems suffers from the explosion of states caused by the many possible permutations of concurrent events. Unfoldings are a verification technique that avoids this explosion by disregarding the order of concurrent events. It belongs to the group of so-called partial-order methods for model checking, which also contains Godefroid's sleep sets, Peled's ample sets, Valmari's stubborn sets, and others.

The unfolding technique can be applied to any concurrency model in which (a) global states are represented as tuples of local states (plus possibly values of variables, contents of channels, etc ...) and (b) transitions may be executed by parts of the system. This induces a notion of independence: two transitions are independent if the parts of the systems executing them are independent.

The unfolding technique replaces the construction of the state space of the system by the construction of its unfolding. If the system consists of one automaton, the unfolding is the familiar unwinding of the automaton into an infinite tree. If the system consists of a tuple of communicating automata, then the unfolding can be seen as a tuple of communicating trees. The unfolding is infinite for most systems of interest, but the construction stops when enough information has been obtained. "Enough information" depends on the kind of property to be checked. Using the automata-theoretic approach to model-checking, the verification of all (next-free) LTL properties can be reduced to the following three problems: Can a given transition be executed at least once? Can a given transition be executed infinitely often? Are there livelocks, i.e., executions with an infinite tail containing only invisible transitions?

In the talk, I sketched a solution to the first of these three problems. I then provided some data about performance, and concluded by comparing the technique with BDD-approaches and with other partial-order techniques.

The talk is based on two papers, co-authored with Keijo heljanko, available at

1. `http://www.tcs.hut.fi/~kepa/publications/`

2. A (somewhat outdated) introduction to the unfolding technqiue can be found at
   `http://www7.in.tum.de/gruppen/theorie/pom/`

3.7 Exploring Infinite State Spaces with Finite Automata

Pierre Wolper (University of Liege)

After introducing the type of state-space search used in verification, this talk focuses on the symbolic exploration of infinite state-spaces. It describes the two ingredients needed for this type of state-space search: an adequate symbolic representation and an acceleration technique. Two acceleration techniques are briefly described: widening and the use of meta-transitions corresponding to the repeated execution of loops. Thereafter the talk turns to symbolic representations and shows, through a series of examples, that finite-automata based representations are a widely usable and powerful class. The examples used include programs manipulating integers, reals and unbounded message queues.

For references, see
`http://www.montefiore.ulg.ac.be/˜pw/papers/papers.html`

3.8 Exploring Infinite State Spaces with Tree Automata

Orna Kupferman (Hebrew University)

We develop an automata-theoretic framework for reasoning about infinite-state sequential systems. Our framework is based on the observation that states of such systems, which carry a finite but unbounded amount of information, can be viewed as nodes in an infinite tree, and transitions between states can be simulated by finite-state automata. Checking that the system satisfies a temporal property can then be done by an alternating two-way tree automaton that navigates through the tree. As has been the case with finite-state systems, the automata-theoretic framework is quite versatile. We demonstrate it by solving several versions of the model-checking problem for μ-calculus specifications and prefix-recognizable systems, and by solving the realizability and synthesis problems for μ-calculus specifications with respect to prefix-recognizable environments.
`http://www.cs.huji.ac.il/˜orna/publications/cav00.ps.Z`

3.9 AI Planning Tutorial

Fahiem Bacchus (University of Toronto)

The Overview covered the Standard Techniques used in AI planning to represent state transitions and planning problems. It also discussed the most popular techniques for searching for plans.

3.10 A Hierarchical Approach to Model-based Reastive Planning in Large State Spaces

Brian C. Williams (MIT)

A new generation of sensor rich, massively distributed systems is emerging that include deep space explorers, ubiquitous computing environments, and sensor webs for monitoring the earth ecosystem. Programming these systems involves reasoning through complex system interactions along lengthy paths between the sensors, control processors and control actuators. The resulting code lacks modularity, and is fraught with error. Model-based programming supports modularity by enabling engineers to program reactive systems by simply articulating high-level control strategies and by plugging together commonsense models of hardware and software modules. Model-based executives use these models on the fly to coordinate between modules, according to the control strategy.

3.11 Using Symmetry Elimination to assist in Search Control in Planning

Maria Fox (University of Durham)

Many planning problems contain symmetries which lead to redundant search. Breaking symmetries during search can yield orders of magnitude improvement in performance, even exponential improvement when problems are highly symmetric. If the domain modeller supplies some of the symmetries present in a problem a planner can exploit these with only minor modification to its search procedure. This presentation describes how symmetries expressed as permutations (of the values and variables describing a planning problem) can be applied to find the symmetric alternatives to failed value choices during the search, thereby allowing these alternatives to be pruned without the necessity to expend any effort on exploring them. A close connection can be drawn with related work on symmetry-breaking in the CSP community. A strength of the work described in this presentation is the potential for the dynamic discovery of symmetries, not present in the initial state of the problem, during the search process. For reference see:
`http://www.dur.ac.uk/maria.fox/symmetry.ps.gz`

3.12 Reducingg Planning Problems by Path Analysis

Ulrich Scholz (TU Darmstadt)

If the explicit representation of a transition graph is too large to be searched directly, it can be helpful to reduce its implicit representation prior to the search. In this talk, we presented path analysis which can perform such a reduction. It is a preprocessing technique motivated by work on planning.

Path analysis is based on a decomposition of the transition graph into - usually dependent - finite state machines. If such a decomposition exists, the global solution to the search problem is a combination of a solution to each of the FSMs.

With a fixpoint computation, path analysis identifies a set of transitions of the FSMs, which can be relevant for a global solution. Transitions that are not part of this fixpoint are irrelevant for a solution to the global search problem and can be eliminated form the transition graph.

The talk mentions ways to guarantee optimality of the reduction, to minimize the fixpoint, and to overcome the high complexity of searching for a fixpoint.

3.13 Utilizing Problem Structure in Local Search: The Planning Benchmarks as a Case Study

Jörg Hoffman (Freiburg University)

Recently, the planning community has, in the context of the 2nd international planning systems competition alogside AIPS-2000, seen a dramatic improvement in terms of the size of problems that state-of-the-art planners can handle. One planner participating in this development is the FF system. This planning system is based on a simple local search approach. The talk briefly describes the main algorithmic techniques used in FF. Afterwards, the excellent runtime behavior of the system on many planning benchmarks is explained in terms of a detailed investigation of the local search topology of these benchmarks: as it turns out, the majority of the benchmark domains do not contain any local minima under an idealized version of FF's heuristic function, and the maximal distance to exits on benches is often limited by a constant.

1. A detailed paper on the FF system is published in JAIR and can be downloaded from `http://www.informatik.uni-freiburg.de/˜hoffmann/papers/jair01.ps.gz`

2. An empirical paper on the local search topology of the benchmarks is published in the proceedings of IJCAI-2001 and can be downloaded from `http://www.informatik.uni-freiburg.de/˜hoffmann/papers/ijcai01.ps.gz`

3.14 SAT-Based Conformant Planning

Enrico Giunchiglia (University of Genoa)

Planning as satisfiability is an efficient technique for classical planning. This technique has been recently extended to conformant planning, that is, to planning domains having incomplete information about the initial state and/or the effects of actions. In this talk we present the basic ideas and procedures for complete and optimal conformant planning as satisfiability. Then, we discuss the limitations of the basic approach and present some domain independent optimizations meant to mitigate the outlined problems. A comparative experimental analysis shows that the resulting procedure is competitive with other state-of-the-art conformant planners.

3.15 Planning under Limited Observability as Search in the Space of BDDs

Alessandro Cimatti (IRST, Trento)

Conformant planning is the problem of finding a sequence of actions that will achieve the goal for all possible initial states and for all possible outcomes of non-deterministic actions.

In this talk, an efficient approach to conformant planning is presented, in the framework of Planning via Symbolic Model Checking. The problem of conformant planning is recast as a problem of search in the space of beliefs. The search space is compactly representeed and efficiently explored by means of Symbolic Model Checking techniques based on Binary Decision Diagrams.

Different planning algorithms are presented. The approach in (1) is based on a breadth-first search, while the approach in (2) is based on goal-directed, heuristic search techniques, with selection functions that take into account the degree of knowledge. The different algorithms are implemented in the MBP planner. The experimental evaluations confirm the efficiency of the approach.

REFERENCES

1. A. Cimatti and M. Roveri *Conformant Planning via Symbolic Model Checking* Journal of Artificial Intelligence Research, 13:305-338, 2000. `http://www.cs.washington.edu/research/jair/volume13/cimatti00a.ps`
   Also IRST Technical Report 0006-04, Istituto Trentino di Cultura, June 2000.

2. P. Bertoli, A. Cimatti, and M. Roveri *Heuristic Search + Symbolic Model Checking = Efficient Conformant Planning* In Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI 2001), Seattle, USA, August 2001. IRST Technical Report 0105-07, Istituto Trentino di Cultura, May 2001.

3.16 MDP Tutorial

Robert Givan (Pardue University) + Ron Parr (Duke University)

This tutorial reviews the basic definitions and techniques in the area of Markov decision processes (MDPs). We discuss discounted, average, and finite-horizon total reward objective functions, and their associated value functions. We give Bellman equations defining the optimal value function for the discounted case and present the standard algorithms for solving MDPs: value iteration, policy iteration and linear programming. We present extensions to the basic model for very large factored state spaces, and discussed techniques for solution of large state space problems. First, we describe techniques for constructing smaller equivalent problems by factored statespace aggregation (”model minimization/reduction”) and related techniques for carrying out standard dynamic programming solutions in factored form (”structured dynamic programming”). Finally, we present the topic of value function approximation, by which the value function for a large MDP is approximated using a parametric representation. Convergence results and error bounds for several popular methods are dicussed.

3.17 Optimal Search

Jürgen Schimdhuber(IDSIA)

I gave an overview over Kolmogorov complexity and Kt-complexity and asymptotically optimal search algorithms, such as Levin's universal search(1973) and my postdoc Hutter's fastest and shortest algorithm for all well-defined problems (2001). Then I tried to encourage the model checking and planning and MDP communities to apply Adaptive Levin Search (Machine Learning, 1997) to their search problems: Put all your prior knowledge into a probabilistic (typically non-universal) programming language whose instructions include your favorite search algorithms. In Phase $i = 1, 2, 3, ...$ run all programs p such that $t(p)/P(p) < 2^i$, until a solution is found, where P(p) is p's probability, and t(p) its runtime. This is an optimal way of allocating time to the computation of solution candidates, given the prior knowledge in P. You can start with small search problems and adapt P such that the probability of successful programs increases. (I also mentioned metalearning self-modifying policies for reinforcement learning in partially observable environments.)

Links:

1. J. Schmidhuber, J. Zhao, and M. Wiering. Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement. Machine Learning 28:105-130, 1997.
   `ftp://ftp.idsia.ch/pub/juergen/bias.ps.gz`
2. Universal Learning Algorithms:
   `http://www.idsia.ch/˜juergen/unilearn.html`
3. Self-modifying policies:
   `http://www.idsia.ch/˜juergen/ssabook/ssabook.html`
4. Additional papers:
   `http://www.idsia.ch/˜juergen/onlinepub.html`

3.18 Symbolic Model Checking of Domain Models for Autonomous Spacecrafts

Charles Pecheur (NASA)

NASA is investing a lot into model-based reasoning, as a way to build "intelligent" autonomous software that require less human supervision. The validation of such systems is a critical issue: they are built to perform complex tasks over extended periods and under a wide range of external conditions, so the testing task is daunting.

This talk discusses our current research in applying symbolic model checking techniques to verify domain models for NASA's Livingstone model-based diagnosis system. We have developed a translator that converts thos model into the syntax of the SMV model checker (Carnegie Mellon). Thanks to its BDD-based symbolic analysis, SMV allows exhaustive coverage of full Livingstone models with very large state spaces ($10^5 5$). A peculiar aspect is that due to their loose internal coupling, Livingstone models feature a huge but very shallow state space. We briefly discuss some technical issues and experimental results. We conclude with some innovative, unexplored ideas on the use of symbolic analysis to verify whether a model is ”diagnosable”, i.e. whether its observable variables are sufficient to discriminate between nominal behaviour and critical faults.

3.19 Hierarchical Image Computation and Dynamic Conjuntion Scheduling in Symbolic Model Checking

Christian Stangier (University of Trier)

When traversing large state spaces by using OBDDs (e.g. in symbolic model checking), one often faces the problem that the transition relation of the system under consideration is too large to be represented as a monolithic OBDD. Even a representable transition relation might in this form not be suited for efficient image computation. Then, a partitioned transition relation has to be used. This talk presents methods to obtain a partitioned transition relation and how to schedule the conjunctions to obtain the image. Also, a new approach to partitioning is presented. This approach creates a hierarchical partitioning based on structural information abtained from the system. Based on this hierarchical partitioning a dynamic conjunction scheduling algorithm is presented. The effectiveness of this approach has been proven by benchmarking experiments.

3.20 Decidability and Undecidability Results for Planning with Numerical State Variables

Malte Helmert (Freiburg University)

These days, propositional planning can be considered a quite well-understood problem. Good algorithms are known that will solve a wealth of very different and sometimes challenging planning tasks, and theoretical computational properties of both general STRIPS-style planning and the best-known benchmark problems have been established.

However, propositional planning has a major drawback: The formalism is too weak to allow for the easy encoding of many genuinely interesting planning problems, specifically those involving numbers. A recent effort to enhance the PDDL planning language to cope with (among other additions) numerical state variables, to be used at the third international planning competition, has increased interest in these issues.

In this contribution, we analyze ”STRIPS with numbers” from a theoretical point of view. Specifically, we show that the introduction of numerical state variables makes the planning problem undecidable in the general case and many restrictions thereof and identify special cases for which we can provide decidability results. A paper on this topic can be found on
`http://www.informatik.uni-freiburg.de/˜helmert/publications.html`

3.21 What is Large? Searching State Spaces under Real-Time Requirements

Jana Koehler (IBM)

Offering individually tailored services to customers becomes in interesting area for thepractical application of search techniques. The talk presented an example from the elevator industry, where a systematic depth-first search, branch-and-bound algorithm is used to compute optimal travel routes in an elevator system. These travel routes satisfy various constraints imposed by the customer, for example they guarantee that passengers can only reach floors they are granted access to, they separate passengers that were not allowed to meet in the elevator cabin, or they enable express service to distinguished passengers. The search space size of these problems ranges in the $10^{10} - 10^{15}$ states. A novel combination of techniques from AI planning and constraint satisfaction guarantees that optimal solutions are found in less than 100 ms. Embedding the search core into a multi-agent architecture yields a high-performing, self-adaptive, and modular control software.

3.22 Sampling Techniques for for Large MDP Problems

Robert Givan (Pardue University)

We evaluate and extend recent work in sampling-based control for large stochastic planning problems using problems selected from the domain of intelligent packet-network control. We first survey two recently proposed methods for applying sampling to such large (PO)MDP problems—the sampled look-ahead tree proposed by Kearns et al. and the policy rollout method proposed by Bertsekas and Castanon. Our first new technique is inspired by Ginsberg's Monte Carlo card selection algorithm for computer bridge, and our second new technique generalizes the policy rollout technique to ”roll out” a finite set of base policies simultaneously, rather than a single base policy as originally proposed. We evaluate all four techniques against a suite of MDP problems derived from telecommunications network control.

3.23 Projection Methods

Ron Parr (Duke University)

In this talk I provide an overview of methods we have developed for approximating value functions for Markov decision processes (MDPs) using projection. While other approaches to value function approximation typically use sampling, our methods directly incorporate the influence of every state in the state space by using an implicit, factored representation of the MDP transition model and reward function. This approach also lets us achieve meaningful a posteriori error bounds on our approximate value functions.

This is joint work with Carlos Guestrin (Stanford University) and Daphne Koller (Stanford University).

3.24 Minimizing Approximate Models versus Approximating Minimal Models

Tom Dean (Brown University)

Several current methods for solving very large state space, factored Markov decision processes involve searching (implicitly or otherwise) in the space of reduced models that aggregate states according to similar dynamics and expected rewards. These methods proceed by splitting blocks in a state-space partition to improve some local measure of performance. We consider the advantages and disadvantages of such top-down refinement methods as they pertain to the size and generality of the models explored in the course of searching. We then consider a method that, at first blush, is not guided by splitting blocks (refining partitions) of dissimilar states but rather by splitting blocks to improve some approximate global measure of optimality. The method works by constructing an approximate model from the current partition and then evaluating the optimal solution for this approximate model. We analyze the relationship between optimal solutions to these approximate models and optimal solutions to the original model and describe some preliminary experimental results.

3.25 Reducing the State Space Buchi Automata using Simulation Relations

Thomas Wilke (University of Kiel)

The purpose of this talk was to explain how structural similarities of transition systems can be exploited to reduce the size of their state spaces. More specifically, it was first explained - using a game-theoretic approach - (1) what it means for two states of a nondeterministic finite automaton to be bisimilar and that they simulate each other, (2) how this can be checked efficiently, and (3) how state space reduction by quotienting with respect to bisimulation and simulation equivalence, respectively, works. The second part of the talk was devoted to the question how fairness constraints, which arise when nondeterministic automata are used to recognize languages of infinite words, can be taken care of. In particular, it was explained what delayed simulation is about and why it is interesting in the context of Bchi automata. For details, consult the paper available at
`http://cm.bell-labs.com/cm/cs/who/kousha/fair_icalp.ps.gz`

3.26 Decidability of Language Equivalence for DPDA

Colin Stirling (Edinburg University)

Despite intensive work throughout the late 1960s and 1970s, the equivalence problem for determinisitic pushown automata, DPDA, whether language equivalence is decidable for deterministic context-free languages, remained unsolved until 1997 when Sénizergues announced a positive solution. It seems that the notation for configurations of DPDA, although simple, is not rich enough to sustain a proof. Deeper algebraic structure needs to be exposed. The full proof by Sénizergues, in journal form, appeared in TCS early in 2001.

I produced a different proof of decidability using ideas from concurrency and language theory that is essentially a simplification of Sénizergues's proof that also appeared in TCS in 2001. The first step is to view the problem as a bisimulation equivalence problem for a process calculus whose expressions generate infinite state transition graphs. The process calculus is built from determinising strict grammars.

The proof of decidability is still very complex because showing termination uses a technique for "decomposition" that is based on unifiers and auxiliary recursive nonterminals. Sénizuergues uses a more intricate mechanism. This means that the syntax of the starting process calculus has to be extended with auxiliary symbols. It also introduces nondeterminism into tableaux with the consequence that the decision procedure is two semi-decision procedures, and therefore there is not an upper bound on complexity. This is also the case with proofs of decidability in restricted cases, such as for DPDA without ε-transitions that was first shown in 1980.

In the talk I describe a simpler decision procedure that is determinisitic and that avoids the decomposition mechanism for termination. One consequence is that the syntax of the starting process calculus is not extended. Another consequence is a primitive recursive upper bound on the complexity of the procedure. For a preliminary full paper see
`http://www.dcs.ed.ac.uk/home/cps/newdpda.ps`

3.27 Explicit-State Search

Stefan Leue (Freiburg University)

Traditional explicit-state model checkers perform state-space searches that ignore information available on the state space and on the property specification structure. I show how the intorduction of heuristic search algorithms can greatly reduce the length of error trails in communication protocol verification for safety properties, and how modest improvements can be achieved when verifying liveness properties. For some problems, heuristic search can render erstwhile unsolvable problems solvable. I finally show that heuristic seach techniques can co-exist with partial-order reduction.

3.28 Common Exploration Techniques in Search, Planning and Model Checking

Stefan Edelkamp (Freiburg University)

The talk elaborates on a common ground for search, planning and (error detection in) model checking: the state space, spanned by a set of operators, an initial state and a set of goal states.

The techniques presented in the talk include symbolic pattern databases for both explicit and symbolic search engines as well as scheduling approaches to estabilish optimal concurrent solutions in real-time domains.

All applied algorithms are directed, i.e., they accelerate the search with respect to the set of goal states.

We rise the question of how much transfer of technology, like partial order reduction, symbolic representation, abstraction, state compaction etc., has been and can be achieved.

3.29 Space-Efficient Search

Richard Korf (UCLA)

Many search algorithms are memory-limited and exhaust the available memory in a matter of minutes. Disks generally don't help due to their long latencies for random access. We discuss several techniques for dealing with this problem. If the problem space is approximately a tree, depth-first search runs in space that is only linear in the depth of search. Furthermore, depth-first iterative deepening simulates breadth-first search with an overhead of only B/(B-1), where B is the branching factor of the tree. In a problem with many paths to the same node, however, depth-first search explores every path to a given node. For example, in a 2-dimensional grid, the number of nodes at a radius of R is only $O(R^2)$, while the number of paths is $O(3^R)$. If such a problem space has a great deal of structure, a finite-state machine can often be learned which will reduce the number of duplicate nodes generated by a depth-first search. Even if the obvious problem space for a problem results in multiple paths to the same node, there may be an alternative problem space for the same problem which is tree structured. Finally, we discuss a technique called frontier search, which stores only the frontier nodes in the search, and not the interior nodes. For example, in a 2-dimensional grid graph, the number of interior nodes is quadratic in the search radius, while the number of frontier nodes is only linear, resulting in a dramatic space reduction. The actual solution path can be reconstructed by a divide-and-conquer technique.

3.30 Benefits of Bounded Model Checking in an Industrial Setting

Moshe Y. Vardi (Rice University)

The usefulness of Bounded Model Checking (BMC) based on propositional satisfiability (SAT) methods for bug hunting has already been proven in several recent work. In this paper, we present two industrial strength systems performing BMC for both verification and falsification. The first is *Thunder*, which performs BMC on top of a new satisfiability solver, *SIMO*. The second is *Forecast*, which performs BMC on top of BDD package. SIMO is based on the Davis Logemann Loveland procedure (DLL) and features the most recent search methods. It enjoys *static* and *dynamic* branching heuristics, advanced *back-jumping* and *learning* techniques. SIMO also includes new heuristics that are specially tuned for BMC problem domain. With Thunder we have achieved impressive capacity and productivity for BMC. Real designs, taken from Intel's $Pentium© 4$, with over 1000 models variables were validated using the default tool settings and without manual tuning. In Forecast, we present several alternatives for adapting BDD-based model checking for BMC. We have conduced comparison of

Thunder and Forecast on a large set of real and complex designs and on almost all of them Thunder has demonstrated clear win over Forecast in two important aspects: capacity and productivity. For details, consult the paper available at
`http://www.cs.rice.edu/˜vardi/papers/cav011.ps.gz`