An International Symposium
on
Theoretical Computer Science
in honor of
Boris Trakhtenbrot
on the occasion of his retirement and 70'th birthday
Tel-Aviv University, June 10-13, 1991

Sponsored by
The Moise & Frida Eskenasy Institute of Computer Sciences
Tel-Aviv University

Abstracts of Lectures

Proofs as Processes

Samson Abramsky
Imperial College

We present what can be seen as either a type inference system for a communicating process calculus, or a realizability interpretation for a constructive logic, in which the realizers are processes rather than, say, functional programs. The logic is Classical Linear Logic; there is considerable latitude as to the choice of process calculus, but we shall use Milner's π-calculus as our principal example. The whole construction may be seen as a particular instance of a general notion of realizability for Classical Linear Logic, leading to the construction of a categorical model of the logic as the category of partial equivalence relations over the realizability algebra. The approach can be seen as a first step towards a "Propositions as Types" paradigm for processes comparable to what has been achieved for functional programs.

Problems of Decidability and Safety in Databases

Arnon Avron and Yoram Hirschfeld
Tel-Aviv University

From Trakhtenbrot's Theorem it easily follows that it is undecidable whether a given first-order query is *safe*, i.e., has a finite number of solutions in all possible database interpretations, or if it is domain-independent. It is known, however, that if these questions are considered only relative to a given database in a language with no function symbols (and no built-in infinite relations) then they are decidable, and every query of the second type belongs also to the first. It was conjectured by Kiffer that the same is true also for languages with function symbols. In this paper we prove these conjectures, using methods from model theory. Moreover, we construct algorithms that, given any relational database, corresponding database structure, and first-order query in a language with function symbols, determine if the number of solutions is finite and if so, lists them all.

Law-Abiding Constructors

Val Breazu-Tannen and Ramesh Subrahmanyam
University of Pennsylvania

The talk will discuss the foundations of programming by structural recursion on datatypes whose constructors are characterized by equational laws, such as sets, bags, and the *append* presentation of lists. Naive programs written in this style may, under straightforward evaluation, yield inconsistent outputs when given as inputs different constructor expressions denoting the same data object. In order to exclude such unsafe programs, we propose a denotational semantics (based on partial functions) in which both the constructor laws and the evaluator are sound, hence all the programs whose meaning is defined are operationally safe. Not surprisingly, definedness in this model is in general not r.e. We propose a logic that soundly approximates both definedness and equality in the model. Moreover, on provably defined programs, the logic is at least as strong as the evaluator.

We will also show how safe programs using structural recursion on the *cons* (*increment*, *insert*) presentation of lists (bags, sets) can be uniformly transformed into safe programs using structural recursion on the *append* (*sum*, *union*) presentation of lists (bags, sets), and conversely. Moreover, we examine how these transformations interact with provable definedness and equality, for certain fragments of our logic. This application was a main source of technical inspiration for the general theory.

A Functional Calculus for Nondeterministic Interactive Systems

Manfred Broy
Technische Universität München

A specification and programming language is introduced that allows formulation of interactive systems in a purely functional style. In particular, the language comprises deterministic and nondeterministic expressions as well as recursive definitions of deterministic and nondeterministic functions. For this language a functional and logical calculus is given that allows reduction of nondeterministic expressions to logical formulas. The use of the calculus is demonstrated by a number of examples.

Constructive Type Theory as a Foundation for Computer Science

Robert Constable
Cornell University and Weizmann Institute

We discuss some of the boons as well as shortcomings of constructive type theory as a foundation for computer science. Certain new concepts are offered for tailoring these theories to this task including an idea for collecting objects into subtypes and a proposal for using logic variables and treating them as part of the definition of the logic.

Infinite Rewriting

Nachum Dershowitz
University of Illinois

Some rewrite systems are not terminating, but allow instead for transfinite derivations that have limits. Under certain conditions, the limit is in normal form; other conditions ensure uniqueness of normal forms. Together, these properties imply the existence of a unique (potentially infinite) normal form for any input term, which can be viewed as the "value" of the term initiating the derivation.

This is joint work with Stephane Kaplan and David Plaisted (the results of which will appear in a paper entitled, "Rewrite, rewrite, rewrite, rewrite, rewrite,...," in *Theoretical Computer Science*), embellished by recent work of others. Many open questions remain.

Parallel Algorithms for String Matching, Upper and Lower Bounds

Zvi Galil
Tel-Aviv University and Columbia University

The string matching problem is one of the most studied problems in Computer Science. While it is very easily stated, and many of the simple algorithms perform very well in practice, numerous works have been published on the subject and research is still very active. In this talk we survey recent results on parallel algorithms for the string matching problem.

Bisimulation Semantics for Higher Dimensional Automata

Rob van Glabbeek
Stanford University

A geometric model of concurrency based on higher dimensional automata is presented. I present homomorphisms (with respect to standard process algebra operators) from behaviour structures, event structures and Petri nets into the geometric model. A bisimulation equivalence on HDA's is provided such that two behaviour structures are behaviour structure bisimilar (respectively two event structures are history preserving bisimilar or two Petri nets are fully concurrent bisimilar) iff the associated HDA's are bisimilar.

An Abstract Standardization Theorem for Orthogonal Transitions Systems

Georges Gonthier, Jean-Jacques Lévy, Paul-André Melliès
INRIA-Rocquencourt, France

The standardization theorem, which is central to the lambda calculus, shows the completeness of call-by-name evaluation strategies for obtaining normal forms of lambda expressions. Unfortunately, this theorem uses a left-to-right property of the lambda calculus, which does not always apply to arbitrary term rewriting systems (TRS). A resolution of this problem was presented by Huet and Lévy in the case of *orthogonal* TRS by use of residuals of any subexpression.

We present here an abstract solution, not using the structure of terms, based on modeling a reduction system as a multigraph, where nodes are terms and arcs are contraction of redexes. An abstract notion of residual is given on the graph, as a relation between arcs indexed by reductions. Finally, a partial order between outgoing arcs of a node is supposed to model the nesting of redexes inside terms. We first give a definition of an abstract standard reduction, and show how three very simple axioms are enough to prove the standardization theorem. Moreover, a fourth axiom (analogous to Berry's stability property) gives the uniqueness of a standard reduction in a permutation class.

This abstract standardization works for transition systems like the lambda calculus, lambda calculus with delta rules (PCF), orthogonal TRS, orthogonal DAGs, interaction networks, proof networks.

Evolving Algebras

Yuri Gurevich
University of Michigan

Evolving algebras were introduced as an attempt to improve on Turing's thesis. They simulate algorithms step for step and provide operational semantics to programs and programming languages. We will describe the basics of the evolving algebra approach and illustrate its application to the Occam programming language.

Transparent proofs

Leonid Levin
Boston University

We propose very fast and generic mechanisms of checking computations.

Some mathematical proofs are too large to be checked by a single human, e.g., the Four Color Theorem or the "Enormous Theorem" on classification of simple finite groups. Another example is a statement that a given large tape contains the program, input, and correct output of a large computation. One might attempt to verify the claim by repeating the computation, but what if there is a systematic error in the implementation?

We show that in almost linear time, any proof, i.e., nondeterministic computation, can be transformed into a *transparent* form, verifiable in *polylogarithmic* sequential Monte-Carlo time. Correct proofs are always accepted. Wrong proofs either prompt fast rejection with overwhelming probability or allow an easy correction. The statement being proven must be given in error-correcting code, since it cannot be reviewed in sublinear time.

This is joint work with Laszlo Babai, Lance Fortnow, and Mario Szegedy of Chicago University, inspired by Manuel Blum's concept of instance checking and exploiting recent advances in interactive proof protocols (while not being interactive themselves). More details are given in the ACM STOC '91 proceedings.

Fair Completeness without Ordinals

Zohar Manna
Stanford University and Weizmann Institute
and
Amir Pnueli
Weizmann Institute

We reconsider the proof rules for proving termination (and other response properties) of reactive systems, where the validity of the property strongly depends on the assumption of *fairness*. Such rules are usually based on the identification of a well-founded pre-order such that taking a *helpful* transition from a state s leads to a state s' which is strictly lower than s according to this pre-order, while taking an *indifferent* transition leads to a state which is not higher than s.

The most recent proofs of completeness for these rules suggested that the assertional language in which this pre-order can be expressed must include *fixpoints* and *ordinals*. In the paper we show that fixpoints are sufficient and no explicit reference to ordinals is necessary.

The main place where ordinals have been used in previous proofs is in showing that any well-founded partial order $\sqsupset$ can be embedded in a total well-founded pre-order $\succeq$. We replace this use by showing that the same pre-order can be obtained as the maximal fixpoint of the implication

$$a \succeq b \quad \rightarrow \quad (\forall b' \sqsubset b)(\exists a' \sqsubset a)\ a' \succeq b'.$$

Systems of Concurrent Automata

Albert R. Meyer
MIT Laboratory for Computer Science

Algebraic approaches to specifying and verifying concurrent processes, such as Hoare's CSP and Milner's CCS, have been richly developed for more than a decade. So too has Petri Net theory. In the CSP/CCS theories, concurrency is modelled by nondeterministic state machines, with the consequent absence of distinctions between concurrent and interleaved actions, or local and global system states. Petri Net theory, on the other hand does distinguish concurrency from nondeterminism by axiomatizing a "causal" partial order on process actions. But net algebra remains undeveloped, and a persuasive explanation of how an external observer can detect causality is just being developed.

We describe an elementary operational model of concurrent systems of sequential automata which yields straightforward notions of concurrently performable actions and concurrently enabled but conflicting actions. Our model is close to that of "safe" Petri Nets, but offers a more generous notion of local versus global behavior. It also allows a simply defined operation of "action-interrupt" which resembles action-refinement. We show that interrupts allow ordinary observers indirectly to discover causality. So in this framework, causality is justified as a derived notion—not an axiomatic primitive.

Resumption Semantics for Dataflow Nets

Gordon Plotkin
University of Edinburgh

Using powerdomains, denotational semantics have been given for a variety of models of concurrency, but—as it seems—not so far for dataflow. We present such a semantics. While it is compositional and does avoid the Brock-Ackermann type problems, it is not fully abstract. However, by using instead a category of cpos with extra algebraic structure for nondeterminism, a fully abstract semantics is obtained. This is just the well-known trace semantics, but now recaptured in a systematic setting.

Presentation of these semantics raises interesting issues. It is convenient to adopt the monadic approach of Moggi for the resumption semantics, with the powerdomain as the monad. The presentation language is then a variant of Moggi's computational lambda calculus, and the semantics of that language is given via the Kleisli category. However, this is not so appropriate for the "algebraic" semantics, where extra type constructors such as tensor product are featured. Now one needs a language over the category of algebras. It turns out that a linear lambda calculus corresponding to intuitionistic linear logic is appropriate.

Must Schedule-Automaton Duality Entail Conflict Persistence?

Vaughan Pratt
Stanford University

An automaton exhibits persistent conflict when it is a tree, or in the case of a concurrent automaton modelled as a multidimensional cell complex, when it has the homotopy of a tree. Operationally this means the automaton permanently records every prior conflict in its states. Existing models of computation having a satisfactory duality between schedules as conjunctive sets of events and automata as disjunctive sets of states all exhibit such persistent conflict.

However the natural representation of mutual exclusion is not as a persistent conflict but rather as a hole denoting an ephemeral conflict, one soon to be forgotten. Automata defined as cell complexes admit ephemeral conflict. However no duality is presently known for which such automata have corresponding schedules.

We pose the problem of giving plausible notions of schedule and automaton such that schedules and automata stand in a dual or contravariant correspondence and such that the class of automata thereby defined admits ephemeral conflict.

As background for this talk we briefly review the geometric notion of a cellular automaton we introduced at POPL'91 and the vector-space-like notion of an event space we introduced at AMAST'91.

The Consequence Relation of Process Logic

A. Rabinovich
IBM Watson, USA

We compare two algebras: an algebra of linear trace languages and an algebra of relations. The fundamental operations in the trace algebra are synchronization (parallel composition) of two trace languages and port-hiding—erasing the symbols associated with a port. The corresponding operations in relational algebra are conjunction and existential quantification. These two algebras are shown to have the same laws, i.e., two terms have the same meaning in all trace interpretation iff they have the same meaning in all relational interpretations. Moreover, we show that these algebra have the same consequence relations. We embed both algebras into first order logic and through this embedding obtain a sound and complete proof system for reasoning in the algebras.

Concurrent Temporal Logic

W. Reisig
Technische Universität München

There is an ongoing discussion about "true concurrency" (also called partial order-, pomset-, or causality based semantics): is it just a technicality, is it useful, or is it essential for a proper understanding of concurrent systems?

Temporal Logic is currently a favorite instrument to describe and to prove liveness properties of concurrent systems. So, it may be useful to search for a Temporal Logic that exploits "true concurrency."

We suggest such a "Concurrent Temporal Logic," based on concepts of the UNITY-logic. Formulated in terms of interleaving semantics, system properties such as "each partial order run has an interleaving sequence with property p" can be expressed. There exists a proof rule of the form "if p leads to q and if r leads to s, and if p, q, r and s are pairwise independent, then $p \wedge r$ leads to $q \wedge s$."

As an example application, we consider a distributed scheduling algorithm for allocation of scarce resources.

Some Dependence-Independence, Conservativity and Expressibility Results on Weak Theories Without Exponentiation

V. Yu. Sazonov
Institute of Program Systems

Some versions of Bounded Arithmetic, Theory of Binary Strings and Set Theory (BST) are considered where the provably recursive functions, are precisely those definable by terms, and further coincide with the polynomial-time computable functions. In case of Bounded Set Theory with arbitrary universes of finite or even infinite sets, the inductively definable functions again provide the same class of functions. BST is also considered as a Nested Data Base Query and Specification Language.

The exponentiation function is here only partial recursive. That is why in the framework of such theories, finite discrete objects such as strings may be correctly divided into constructive and nonconstructive ones. This is very close to Kolmogorov's complexity theory and analogous to Gödel's notion of constructible sets in ZF.

As in ZF, much depends on existence axioms concerning nonconstructive objects which imply some independence results relating the $P = NP$ problem, constructivity of Markov's Principle, and a Formal Church-Kleene Thesis.

In one such weak theory the problems of existence of complete calculi for classical predicate and propositional logics, which are open due to partiality of exponentiation, are shown equivalent. So, these results demonstrate that $P = NP$ and $Co\text{-}NP = NP$ problems are natural foundational questions, rather than purely complexity theoretic.

It is shown that Kripke-Platek Set Theory without Foundation Axiom but with Transitive Closure Operation—the kernel part of BST—is a conservative extension of the Extensionality Axiom only, relative to bounded, i.e., Δ^0-formulas.

A very simple version of Theory of Feasible Numbers is introduced where "the" number 2^{1000}, and even, in some more special sense, the number 1000, are treated quite rigorously as infinite.

Learning Concept Classes by Examples via Quantization of Underlying Sample Space and via Boundary Search

Eli Shamir
Hebrew University

A novel approach to the selection of a concept out of a class, on the basis of yes/no labelled samples, was initiated by Valiant. It was later seen to connect strongly to sampling theory in Statistics—VC index and uniform convergence of empirical measures. The approach goes over to learning fuzzy concept classes. Two extreme cases learning algorithms are required to cope with are:

A. The distribution-independent case, emphasized by Valiant, and

B. the case of fixed distribution over the sample space X, which occurs often in engineering applications.

We examine the role of "quantizations" of X for both cases, giving some theorems and examples. A very useful handle in case B. is the intuitive notion of the boundary $E(c)$ of a concept c, which is a set of small measure in X, or in the dual of X, where the specific concept c undergoes an abrupt change, hence learning $E(c)$ is a key to a learning algorithm for the class.

Datalog and Expert Systems

Mikhail Taitslin
Tver University

A deterministic semantics for Datalog, similar to that of Prolog, is considered. Predicates definable by Datalog-programs over finite linearly ordered domains are shown to be exactly those computable in *PTIME*. Two modifications of Datalog similarly correspond to *PSPACE* and *EXPTIME*.

My Logical Progression

Boris Trakhtenbrot
Tel-Aviv University

I take this opportunity to reflect on the shift of my research interests from classical logic and computability into theoretical Computer Science. I recount especially the scientific, ideological, and human environment of the early years not covered in previous memoirs.

Complexity Classes Defined by the Local Behaviour of Nondeterministic Machines

Klaus W. Wagner
Universität Würzburg

The acceptance of an input by a nondeterministic polynomial time machine can be considered as a combination of two processes: the generation of the computation tree with 0-1-values at its leaves, and the evaluation of all nodes of the tree with 0-1-values by boolean functions given in the nodes. The input is accepted iff the root of the computation tree gets value 1. Different computation models can be obtained if different boolean functions (or sets of boolean functions) are allowed in the evaluation process. It is well known that one can obtain in such a way exactly the complexity classes *P, NP, co-NP, Parity-P*, and *PSPACE*. We investigate the case where partially defined boolean functions are allowed, and we obtain exactly one new complexity class. This class, which apparently has not been studied before, has an interesting characterization as a counting class, i.e., a class defined by the number of accepting computations, based on computation trees which are in a sense "balanced."

Partially defined boolean functions are special cases of functions over a finite set. Some results on complexity classes defined by this general type of evaluation functions are also given.