Design, Analysis and Reasoning about Tools: Abstracts from the Third Workshop

Flemming Nielson
(editor)

October 1993

1 Introduction

The third DART workshop took place on Thursday August l9th and Friday August 20th at the Department of Computer Science (DIKU) at the University of Copenhagen; it was organized by Mads Rosendahl and others at DIKU, and Torben Amtoft and Susanne Brønberg helped producing this report. The first day comprised survey presentations whereas the second contained more research oriented talks. The primary aim of the workshop was to increase the awareness of DART participants for each other's work, to stimulate collaboration between the different groups, and to inform Danish industry about the skills possessed by the groups.

The DART project started in March 1991 (prematurely terminating a smaller project on *Formal Implementation, Transformation and Analysis of Programs*) and is funded by the Danish Research Councils as part of the Danish Research Programme on Informatics. To date it has received about 8 million Danish kroner in funding for activities going on at four Danish institutions: The Department of Computer Science at Aarhus University (Uffe Engberg, Peter Mosses, Flemming Nielson, Hanne Riis Nielson, Michael Schwartzbach, Glynn Winskel, and others); The Department of Computer Science at Copenhagen University (Klaus Grue, Fritz Henglein, Neil Jones, Torben Mogensen, Mads Rosendahl, Mads Tofte, and others); The Department of Computer Science at Aalborg University Centre (Hans Hüttel, Anna

Ingolfsdottir, Kim Larsen, Arne Skou, and others); The Department of Computer Science at The Technical University of Denmark (Bo Hansen, Peter Sestoft, and others). This spring the project underwent international review as part of an overall review of the Danish Research Programme on Informatics; copies of the evaluation report are available from the Danish Research Councils.

Goals of the project

The application and development of formal methods is an integral part of the research on theoretical computer science. A main goal of this project has been to act as a catalyst to develop further contacts among the worlds of techniques, methods and tools in different application domains.

An increasing awareness that techniques and tools from one application domain are often relevant to other domains indicates that our project is of relevance to advanced parts of Danish industry.

It may be a somewhat ideal goal to aim for all software and hardware systems to be validated formally. However, the concept of "safety critical systems" has been coined to clearly identify those systems where safety is of paramount importance; examples include traffic control systems for airplanes, trains and cars, process control at nuclear power stations, chemical and biological plants etc. Performing such validation is difficult partly because of the sheer size of realistic applications and partly because of a lack of tools: languages sufficiently powerful to express real-world requirements; programming systems to transform specifications into solutions; and sufficiently powerful and automated validation techniques to prove that a given system satisfies a given specification.

This project participates in international activities enhancing the world state-of-the-art in methods, tools and techniques to ensure the correct performance of systems. A major objective is full or partial automation of tasks aiding the overall goal of producing reliable systems that are faithful to the semantics of various application areas, thereby facilitating the handling of more than just toy applications. This necessitates a strong foundation in semantics and may require the development or adaptation of new theories. Our overall approach is appropriate since our methods are firmly based on semantic foundations. Techniques we have emphasized include automated theorem proving, program analysis, and fully automatic program optimization and transformation techniques.

Activities of the project

Almost all of our activities presuppose the ability to make a semantic description of the system under scrutiny. The last decades of research have shown the advent of Denotational Semantics and Structural Operational Semantics. The ideal semantic description method must be universally applicable, it must be understandable to programmers as well as specialists, it must allow modularity in semantic definitions and needs to scale up well, and it must support the study of program analyses and transformations. None of the existing frameworks fully live up to all these goals.

One of the activities of the project is to continue the development of Action Semantics, which is thought to be more understandable to programmers than either of Denotational Semantics or Operational Semantics. It also allows modularity and scales up well. So far activities have mostly concentrated on descriptive power and on ensuring understandability among non-specialists, but future activities will move on to developing the underlying theory.

A major technique for program analysis is the framework of abstract interpretation. The formulation and underlying principles have been strongly influenced by Denotational Semantics (at least in the UK-Danish school). It is applicable to all languages handled well by Denotational Semantics and a rather general framework for the specification of program analyses has been developed. This may then be used in the implementation of programming languages where one may generate less naive code that takes advantage of the knowledge of the "state" supplied by program analyses. Validating such improved code generations is possible in many instances, but for advanced language constructs obstacles still remain. Also it is necessary to increase the class of programming languages to which the techniques are applicable and methods based on Operational Semantics appear to be very fruitful for this.

Adding concurrency presents additional problems to those outlined above. It is possible in principle to validate that processes satisfy certain specifications but most solutions are subject to combinatorial explosion. To improve upon this so-called local techniques have been developed and an automatic verification tool has been constructed. Much research goes into extending the expressiveness of specifications with more quantitative information like real-time and priorities. This is of great practical interest and some progress in this direction has been made, both theoretically and practically through the construction of a verification tool for real-time processes. Among the future goals are the inclusion of higher-order and context-free processes.

Concrete applications have a tendency to produce a vast number of verification goals. Machine assistance seems called for in dealing with these and in the project we have based most of our work in this direction on the HOL-system. So far the results include the validation of while programs over arbitrary types and the formalisation of domain theory (which is at the heart of Denotational Semantics). But we have also studied other theorem proving systems such as LP in connection with reasoning about parallel programs using a Temporal Logic of Actions. We intend to continue such work in order to increase the Danish expertise in this potentially vital field.

Partial evaluation is a now proven technique for automatic program transformation. It works by specializing a general-purpose program on the basis of partially known data. Its main advantage over other program transformation techniques is that it is almost completely automatic. Initial and successful applications were to develop compilers from interpreters, and automatically to produce compiler generators. Significant recent progress has been made in obtaining better execution efficiency, and applications in other domains are being investigated. A very promising direction is the automatic transformation of specifications in formalised specification languages into solutions in program form. This work has also lead to theoretical insights into the role of constant factors in problem solving, and into relations between partial evaluation and complexity theory. A system partially evaluating programs in the functional language Scheme has been developed and distributed to several hundred sites. A similar system but for programs in the pragmatically important imperative language C is under construction.

Semantic principles also show up in language design. The functional language Standard ML has a modules concept, which is intended for making it easier to write large programs. The key idea is that a certain kind of modules, functors, can be parameterised on simpler modules, structures. We have succeeded in extending the concept of functor, so that functors can be parameterised on functors as well as structures, thereby adding to the expressive power to the modules language. Recently these insights have been incorporated within the ML implementation developed by Bell Labs (USA), which is now available for world-wide use. Related to this is the use of inference systems for analysing when the usual heap-based implementations of functional languages may be replaced by the more efficient stack-based implementations as known for imperative languages.

Our work also touches upon other programming paradigms including object oriented languages. A number of techniques have been developed for the analysis of sub-classes in such language. As an illustration of our belief that research may progress more rapidly when the techniques and methods of different areas meet, it turned out that essentially the same class of techniques could be used to improve upon the partial evaluation of functional languages.

2 Abstracts of Talks given

Kim Guldstrand Larsen: Timed Modal Specifications - Theory and Tools

In this tutorial we present the theory of *Timed Modal Specifications* (TMS) together with its implementation, the tool EPSILON. TMS and EPSILON are timed extensions of respectively *Modal Specifications* and the TAV system.

The theory of TMS is an extension of real-timed process calculi with the specific aim of allowing *loose* or *partial* specifications. Looseness of specifications allows implementation details to be left out, thus allowing several and varying implementations. We achieve looseness of specifications by introducing two modalities to transitions of specifications: a *may* and a *must* modality. This allows us to define a notion of *refinement*, generalizing in a natural way the classical notion of bisimulation. Intuitively, the more must-transitions and the fewer may-transitions a specification has, the finer it is. Also, we introduce notions of refinements abstracting from time and/or internal computation.

TMS specifications may be combined with respect to the constructs of the real-time calculus by Wang Yi. The "time-sensitive" notions of refinements are preserved by these constructs, thus enabling compositional verification. (To be precise, when abstracting from internal composition refinement is not preserved by choice for the usual reasons.)

EPSILON provides automatic tools for verifying all of the refinements presented. We apply EPSILON to a compositional verification of a Train Crossing example.

Neil Jones: When does Specialization Pay off on Practical Algorithms?

Program specialization can lead to significant running time improvements for a number of well known and good algorithms for practical problems, but not for all; and code explosion sometimes occurs. The talk will discuss benefits gained and new problems encountered in applying partial evaluation to speed up programs in areas outside programming languages.

Situations in which partial evaluation can be of benefit will be analyzed in general terms, and applications to computer graphics and some other problems will be described. A few types of program especially susceptible to such optimizations will be characterized. Principles that lead to speedup are emphasized, particularly for the ubiquitous "divide and conquer" paradigm, so one can see when to consider using specialization for practical problems.

A class of "oblivious" algorithms not suffering from code explosion will also be described. Surprisingly, the speedup obtained by specializing to small problem size sometimes speeds up the entire computation by nearly the same amount as n goes to infinity, and avoids problems of code explosion.

Hanne Riis Nielson: Semantics as an Analytical Tool

One of the problems studied in the DART project is: given a program in a language with higher-order abstraction mechanisms, how do we implement it efficiently on a realistic hardware configuration.

The main *techniques* we use are those of program analysis and program transformations. The idea is first to develop *program analyses* that statically extract information about the dynamic execution of the programs. Based on that information we will next suggest *program transformations* that will improve the efficiency of the resulting implementation. The work is firmly based on *semantics notions* thereby allowing us to reason about the correctness of the program analyses as well as the program transformations in a rigourous way.

In this talk we illustrate the approach by showing how to analyse the communication behaviour of a program in Concurrent ML with the aim of implementing it on a simple transputer architecture.

Klaus Grue: Mathematics and Computation

The talk gives an overview of how the parallel graph reducer developed under DART fits into a general mathematical environment. In that environment it is possible to express mathematical definitions, computer programs and mathematical proofs in a textbook format and to execute the programs and verify the proofs using the graph reducer. The goal of this project is to establish a uniform environment that can support theoretical as well as practical computer science developments within the same system.

Torben Amtoft: A new approach to constraint based type inference

Strictness types (originally proposed by Wright) are types where the function arrows are annotated: if f has type $t_1 \rightarrow_0 t_2$ we know that f is strict; but if f has type $t_1 \rightarrow_1 t_2$ we do not know anything for sure. We shall present an inference system for strictness types, formulated in terms of *constraints* between variables ranging over 0 and 1.

The crux of our approach is to distinguish between *positive* and *negative* variables; the former (latter) being those occurring on a function arrow in covariant (contravariant) position. Now we (roughly speaking) define a *normalized* set of constraints to be one where all constraints are of form $b^+ \geq s$, where b^+ is a positive variable and where s is an expression built up from $\sqcap$, $\sqcup$ and negative variables.

It turns out that it is possible to normalize the constraint system *on the fly* by a leaf to root traversal of the proof tree, that is instead of collecting all constraints and then solve them. We shall employ e.g. Tarski's fixed point theorem in this process.

It would be interesting to see if the technique can be (has been?) applied elsewhere.

Kirsten Lackner Solberg: Strictness and Totality Analysis

For the simply typed lambda calculus with constants and fixed points and lazy evaluation we define a strictness analysis. In the strictness analysis we are looking at, we want to do both strictness analysis and totality analysis at the same time. That is we want to distinguish between terms that are surely bottom (does not evaluate; does not have a Weak Head Normal Form) and terms that are surely not bottom (does evaluate; does have a WHNF). First we have a normal type inference for the lambda calculus. Above these underlying types we have the strictness types with strictness information. We have coercions between the strictness types. We have proven the analysis correct with respect to a natural operational semantics.

Fritz Henglein & David Sands: Binding Time Analysis: Models and Logics

We present the first concrete connection between binding time logics (type systems), and the semantic concept of a congruent program division.

The correctness of type-based binding time analysis has been argued by appealing directly to the actions of an off line partial evaluator. By contrast, the correctness criterion introduced by Jones, called *congruence*, can be argued to be independent of (more fundamental than) the partial evaluation process, as it is based on semantic dependencies within the program, and gives the potential for a more modular approach to the correctness of a binding-time-based partial evaluator.

We present a binding time logic (type system) which captures structured binding time properties in a typed lambda calculus with constants and recursively defined data types. The system is sound with respect to the PER model of binding times (a generalisation of Launchbury's projection-based account of Jones' congruence condition) and, we conjecture, can also be shown to be complete (in the spirit of Jensen's strictness logic) with respect to Launchbury's (finite) projection semantics, and Hunt and Sands' PER-based abstract interpretation.

The end product of a binding time analysis is more than just a binding time property for the program. It is a global (sticky/collecting) analysis which associates information with program points. In terms of the logic, this amounts to asking not only *what* properties hold, but *how*. An example of such a global analysis is Launchbury's monovariant binding time analysis which produces a congruent program division. It is shown that the program division is congruent if a certain judgment can be proved in the logic. However, certain simple (sound) extensions to the type system break this congruence condition, suggesting that the particular form of Launchbury's congruence condition is somewhat overly restrictive.

Flemming Nielson: From CML to Process Algebras

Reppy's language CML extends Standard ML of Milner et al. with primitives for communication. It thus inherits a notion of strong polymorphic typing and may be equipped with a structural operational semantics. We formulate an effect system for statically expressing the communication behaviours of CML programs as these are not otherwise reflected in the types.

We then show how types and behaviours evolve in the course of computation: types may decrease and behaviours may loose alternatives as well as decrease. It will turn out that the syntax of behaviours is rather similar to that of a process algebra; our main results may therefore be viewed as regarding the semantics of a process algebra as an abstraction of the semantics of an underlying programming language. This establishes a new kind of connection between "realistic" concurrent programming languages and "theoretical" process algebras.

Jarl Tuxen Lang: Model Construction for Implicit Specifications in Modal Logic

In top-down design of reactive systems, implicit specifications of the form $C(P_1, \ldots, P_n) \models F$ are often encountered, where $C(P_1, \ldots, P_n)$ is a system containing the (unknown) processes $P_1, \ldots, P_n$ and F is a specification. We present a method for constructing the processes $P_1, \ldots, P_n$ (as labelled transition systems) when C is given as a context of process algebra (such as CCS), and F is given as a formula of Hennessy-Milner Logic extended with maximal recursion. The main contribution is the treatment of the simultaneous construction of several processes which together act as a model for the specification. We have implemented two prototype tools (a semi-automatic as well as an automatic one) which are based on the presented theory.

Joint work with Ole Høgh Jensen, Christian Jeppesen and Kim Guldstrand Larsen.

Lars Birkedal & Morten Welinder: Partial Evaluation of Standard ML

In this talk we describe offline partial evaluation of the core of Standard ML, a large typed functional language. It is based on work for our Master's Thesis. Unlike previous partial evaluators for larger languages (like for instance Similix for a subset of Scheme or C-Mix for a subset of C) we have chosen not do to the partial evaluation directly, but to use an untraditional method to transform a program into its generating extension. We show that this approach is in many aspects superior to the traditional approach and that it eliminates the need for self-applying the specializer.

We have developed a binding-time analysis based on non-standard type inference and produce a very efficient implementation of it using constraints. Notice that the analysis is implemented and is not only efficient in theory but also in practice. While this has been done before, we have for the first time succeeded in using the typedness of the source language to make the analysis simple and therefore more trustworthy. We do not have time to go into the analysis in this talk.

To our best knowledge our thesis also describes the first successful strategy for partially evaluating complicated patterns with variable bindings. Earlier attempts have either been for a much simpler class of patterns or have stranded on the need/wish for self-application of the specializer. We show in this talk how the generating extension for programs with pattern matching looks like.

A complete system for partial evaluation of Standard ML with parsing, type checking, binding-time analysis, compiler generation, and pretty printing has been implemented and we report on some experiments with this system. For a standard example, compiling a program by specializing an interpreter for a simple flow-chart language, we obtain a speedup which is a factor of 10 bigger than what other partial evaluators have given according to the literature.

Olivier Danvy: On the Transformation between Direct and Continuation Semantics

Proving the congruence between a direct semantics and a continuation semantics is often surprisingly complicated considering that direct-style lambda-terms can be transformed into continuation style automatically. However, transforming the representation of a direct-style semantics into continuation style usually does *not* yield the expected representation of a continulation-style semantics (i.e, one written by hand).

The goal of our work is to automate the transformation between textual representations of direct semantics and of continuation semantics. Essentially, we identify properties of a direct-style representation (e.g, totality), and we generalize the transformation into continuation style accordingly. As a result, we can produce the expected representation of a continuation semantics, automatically.

It is important to understand the transformation between representations of direct and of continuation semantics because it is these representations that get processed in any kind of semantics-based program manipulation (e.g, compiling, compiler generation, and partial evaluation). A tool producing a variety of continuation-style representations is a valuable new one in a programming-language workbench.

Joint work with John Hatcliff, Kansas State University.

An earlier version was presented at the 9th Conference on Mathematical Foundations of Programming Semantics, New Orleans, spring 1993.

Sten Agerholm: Domain Theory in HOL

The HOL system is an interactive proof-assistant system for conducting proofs in higher order logic. In this talk we present a formalization of domain theory in HOL. The notions of complete partial order, continuous function and inclusive predicate are introduced as semantic constants in HOL and fixed point induction is a derived theorem, just as we can derive other techniques for recursion. We provide proof tools which prove certain terms are cpos, continuous functions and inclusive predicates, automatically. An interface and various definition tools are implemented on top of these basic proof tools. In this way we obtain an integrated system where cpo, continuity and inclusiveness facts are proved behind the scenes.

3 Current Status of DART (Summer 1993)

3.1 SDT: Semantics as a Descriptive Tool (by P. D. Mosses)

The research in this area of DART is centred on *action semantics*, a framework that is intermediate between denotational and operational semantics. Action semantics has significant pragmatic advantages over other frameworks. For example, it scales up smoothly to the description of realistic programming languages (such as Standard Pascal), and an extension to a described language (e.g., from Standard ML to Concurrent ML) requires only an extension of the semantic description — rather than its widespread reformulation.

'Industrial' Applications:

The pragmatic advantages of action semantics were acknowledged in a recent (independent) survey of formal semantics techniques. This led to the adoption of action semantics for the formal description of ANDF (Architecture-Neutral Distribution Format, a high-level 'universal' intermediate code developed and used by the Open Software Foundation and under the ESPRIT project OMI/GLUE). The action semantic description of ANDF being developed (at DDC-International) is represented in RSL (the RAISE Specification Language) so as to allow the use of the sophisticated tools available for RAISE.

Tools:

A tool for parsing, editing, checking, and executing action semantic descriptions is under development, in collaboration with CWI, Amsterdam. It is based on CWI's ASF+SDF system. A prototype tool was presented and demonstrated (by PDM and Arie van Deursen, CWI) at AMAST'93. (The Mathematica-based tool developed earlier has been shelved, so as to concentrate the modest resources available to SDT on the more promising ASF+SDF-based tool.)

Language Descriptions:

The action semantic description of Standard Pascal has been completed (by PDM and David Watt, Glasgow), and is available via FTP. An action semantics for ML has been brought up-to-date by a Ph.D. student (Martín Musicante), and extended to the core constructs of Concurrent ML.

Semantics-Based Implementation:

Jens Palsberg has collaborated with Anders Bondorf on using the Similix system to implement the action notation used in action semantic descriptions. An M.Sc. student (Peter Ørbæk) has implemented an optimizer for action notation. Another M.Sc. student (Christian Lynbech) is investigating direct implementation of the formal operational semantics of action notation.

Theory:

An M.Sc. student (Tony Jakobsen) has devised and implemented an accurate type inference algorithm for action notation. Martín Musicante is investigating how to prove equivalences between action semantic descriptions and other kinds of formal semantic descriptions. A new Ph.D student (Søren B. Lassen) is to develop the laws and logic of action notation.

3.2 SAT: Semantics as an Analytical Tool (by H.R. Nielson)

In the last year the main activities have been within the following areas:

Model Based Program Analysis

We have continued our previous studies of program analysis based on model theoretic notions, in particular abstract interpretation.

The two-level approach to program analysis has previously been used to verify the correctness of abstract interpretations as well as code generations. In his M.Sc.-thesis Torben Lange showed how the techniques can be combined to proving the *correctness of an optimizing compiler.* This work was presented at the PEPM'93 conference.

The efficient implementation of abstract interpretation relies on the cost of *computing fixed points.* A paper (by Flemming Nielson, Hanne Riis Nielson) accepted for WSA'93 presents a structural approach to predicting the number of unfoldings needed to compute the fixed points of functionals arising when performing strictness analysis. Another report (by Hanne Riis Nielson, Flemming Nielson) suggests various iterative algorithms for computing fixed points of special forms of functionals.

Logic Based Program Analysis

Recently we have shifted our attention towards using logical methods in the specification of program analyses, in particular we have studied the use of annotated type systems. Basically there are two approaches: One is to annotate the base types and the other is to annotate the type constructors. One of the goals of this work is to study the expressive power of these methods and to relate them to existing model based approaches.

The use of logic systems was already pioneered in the two-level approach to program analysis where a *binding time analysis* was specified using an annotated type system. This work has been refined by Kirsten Solberg (Odense) and an alternative type reconstruction algorithm based on constraint solution has been developed. This work was presented at WSA'92.

Strictness analysis has received a great deal of attention internationally. We have studied the specification of a *strictness and totality analysis* using an annotated type system with conjunction types and we have proved its correctness with respect to an operational semantics. This work will be reported in a forthcoming paper (by Kirsten Solberg, Flemming Nielson, Hanne Riis Nielson).

One of the drawbacks of the above analysis is that it is very difficult to infer good termination properties for recursive programs. To study this problem we have developed a *termination analysis* for a language with abstract data types. This approach includes some form of well-founded induction. The correctness of the analysis is proved with respect to an operational semantics and shows how well-known notions from denotational semantics such as monotonicity and continuity can be captured in an operational setting. The work is reported in a paper by Flemming Nielson and Hanne Riis Nielson.

Finally, we have studied the correct *application of strictness analysis in a program transformation* that introduces thunks. The analysis as well as the transformation is specified by logical means and the correctness is proved with respect to an operational semantics using a notion of logical relations. This work has been carried out by Torben Amtoft (supported by DART) and will be presented at WSA'93. Also a type reconstruction algorithm with constraint solution "on the fly" has been developed for the strictness analysis.

Functional Languages with Concurrency Primitives

Recently there has been great interest in the development of functional languages with higher-order concurrency primitives, as e.g. CML and Facile. Several members of the DART project have an interest in this area and a workshop is planned on this topic.

So far only very few analyses have been developed for these languages. We have shown how to develop an effect system that captures the *communication behaviour of CML programs* and furthermore that the language of behaviours can be viewed as a process algebra. This is a novel view of process algebras in that a subject reduction theorem for the effect system relates the process algebra to a realistic programming language. This work (by Flemming Nielson, Hanne Riis Nielson) will be presented at CONCUR'93.

The above work has been extended to a polymorphic version of CML and a recent paper (by Hanne Riis Nielson, Flemming Nielson) shows how the behaviour information can be used to analyse whether or not the CML program has a *finite communication topology*, i.e. whether or not only a finite number of processes and channels will be generated during the execution of the program.

Future Work

Most likely, the future work will be centered around program analysis and its application to program transformation. We plan to study model based as well as logic based techniques and we shall be interested in a variety of programming paradigms, in particular functional languages with concurrency primitives. To be able to handle such languages we will study the problems encountered when replacing the traditional denotational basis with an operational basis.

The group has recently been extended with Olivier Danvy (Assistant Professor) and Karoline Malmkjæer (Research Assistent).

3.3 SOC: Semantics of Concurrency (by K.G. Larsen)

During the first half of 1993 considerable progress has been made with respect to the analysis of (dense) real-time systems. A new specification formalism for real-time systems extending Wang Yi's Timed Calculus of Communicating Systems into a Modal Transition System has been introduced. The specification formalism is equipped with a number of equivalences and refinernents for expressing various types of correctness properties. An automatic tool EPSILON for establishing these equivalences and refinements has been constructed. Also a number of (smaller) real-time systems (protocols, train-crossing scenarios, etc.) has been analysed using the tool. This very recent work has been performed in collaboration with Wang Yi (Uppsala University) and Karlis Cerans (Chalmers, Gothenhurg) and will be (or has been) presented at a number of conferences (MFPS'93, FME'93, CAV'93).

Our work on compositional verification has continued. In particular an automatic tool (and its underlying theory) for solving simultaneous "equations" with respect to suitable equivalence "equations" of the form $C(X_1, \ldots, X_n) \sim S$ has been constructed. Here $X_1, \ldots, X_n$ are the unknown components to be found, C is the context in which they are placed, S is the specification of the overall system, and $\sim$ is a "suitable" correctness relation. This work will be presented at CONCUR'93.

Henrik Reif Andersen's thesis was examined this May. His PhD is an impressive piece of work, covering compositionality and the most efficient model-checking algorithms to date for the mu-calculus—this wide-ranging thesis would surely make a welcome book! Winskel and Nielsen's handbook chapter has spawned several research developments: papers with Sassone appearing at MFCS93 and CONCUR93, and one with Joyal at LICS93; PhD students Cheng and Torp Jensen and "speciale" (M.Sc.) student Clausen are putting the ideas to work in operational semantics and model checking (especially of eventuality properties) allied with a study of priority. Winskel's book (MIT Press, January 1993), in particular its chapter on model checking, makes our (and the Edinburgh-Sussex) approaches to model checking accessible. Uffe Engberg is working on symbolic bisimulation and techniques for its determination, applicable also to process calculi with value passing.

This August Aarhus held a highly successful two-week summerschool on "Logical methods for concurrency", funded by the Human Capital and Mobility Programme (invited speakers: Dill, Harel, Moss, Stirling, Thiagarajan, Wolper). Winskel lectured at the TEMPUS summerschool in Brno.

Finally, theoretical work on calculi mixing the functional and parallel paradigm has been carried out. In particular, an extensive analysis of equivalences and their axiomatization within the Fork calculus (a "projection" of CML) has been offered, and has been presented at ICALP'93. Preliminary work on *dynamic* typing of such mixed calculi has been studied, and will be subject for future work. This has many points of contact with the work on "Functional Languages with Concurrency Primitives" pursued under "Semantics as an Analytical Tool".

3.4 SBD: Semantics Based Deduction (by G. Winskel)

A PhD student (Urban Engberg) continues his implementation of a system to support proofs in TLA, the termporal logic of Lamport with whom he is in close correspondence. This work overlaps with the area "Semantics of Concurrency".

Sten Agerholm (funded by DART) has been successful in implementing a significant part of domain theory in the proof assistant HOL. The advantage of HOL over LCF is that the metalanguage of HOL allows much more general reasoning, sometimes necessary in arguing about inclusive predicates or using the new coinduction principles for recursive data types. This work has led to a paper, to be presented at the International HOL meeting in Vancouver, and to appear in the proceedings.

Agerholm and Winskel have jointly supervised Hougaard's student project on computability, recently completed and receiving grade 13. Agerholm will give a 3 week course in HOL as part of Mosses' graduate course in Logic in the Fall. This may involve others from TFL, with whom Agerholm has begun a tentative collaboration on extending HOL tools.

Theoretical work continues in several areas. In particular, Uffe Engberg and Winskel's completeness results for linear logic with respect to Petri nets as a model are to appear in MFCS93. Linear logic reappears in the PhD work of Brauner, Sørensen and Cattani. In particular, Sørensen's work using ideas from category theory to include time in domains was boosted by Winskel's meeting with P-L Curien—the latter has a semantics for linear logic in concrete domains. Linear logic, and especially linear classical logic, are leading to a refined analysis of domains suitable for denotational semantics, and are shedding light of the old and hard PCF full abstraction problem.

3.5 SBPM: Semantics Based Program Manipulation (by N.D. Jones)

Overview

This part of DART is still very much on track with significant outside recognition, and many interesting tasks to be done and leads to be followed. Increasing collaboration within DART is evidenced by joint work between Bondorf and Palsberg, two Aarhus Ph.D. degrees in SBPM (Amtoft and Palsberg, with censor from DIKU), and by employment at DAIMI of partial evaluation researchers O. Danvy and K. Malmkjær (earlier at DIKU).

There has been much research and educational activity in this area since September 1992 with numerous published articles, some in journals and some in a variety of high-level conferences including IFIP W.G. 2.8 Functional

Programming, Formal Methods in Programming and Applications, FPGA, PARLE, PEPM, PLILP, State in Programming Languages, STOC, and the Workshop on Static Analysis. In addition a special issue of the *Journal of Functional Programming* was guest edited by Neil Jones. A new book collecting many results accomplished under DART was published: *Partial Evaluation and Automatic Program Generation* by Jones, Gomard, Sestoft (Prentice Hall International, 425 pp.).

Education. A D.Sc. thesis was defended at DIKU by Klaus Grue (with Henk Barendregt as censor), and 5 students are working on Ph.D. degrees: Lars Ole Andersen, Jesper Jørgensen, Christian Mossin, Kristoffer Rose, and Morten Welinder. 3 M.Sc. theses were written, by Christian Mossin, Eigill Rosager, and one jointly by Lars Birkedal and Morten Welinder.

Guests. We have unusually many guests in 1993-1994 — all of whom came with funding from outside Denmark, and for 1 year or more:

Assoc. Prof. Robert Paige (New York University Courant Institute), Thomas Reps and Susan Horwitz (University of Wisconsin), Assis. Prof. Robert Glück (Technical University of Vienna), Researchers Ryo Nakashige (Hitachi Japan) and Shmuel Sagiv (IBM Israel), graduate student Li Ping Zong (Academia Sinica, Beijing).

Staff. Lars Birkedal was appointed as research assistant under DART funds, and Jakob Rehof in the DART area by a guest's funds. David Sands (Imperial College, London) was employed as postdoctoral guest during all of 1993. Much useful work has been done by DART-employed programmer Peter Holst Andersen.

1993 international meetings held at DIKU. IFIP Working Group 2.8 on Functional Programming (40 participants), FPCA: Functional Programming and Computer Architecture (180 participants), SIPL: State in Programming Languages (80 participants), PEPM: Partial Evaluation and Semantics-Based Program Manipulation (80 participants).

Research goals and their evolution

This past year has seen the first practical application of a new technique for partial evaluation: construction of hand-written versions of "cogen", rather than producing "cogen" by self-applying "mix". This has several advantages, particularly for strongly typed languages and for pattern matching, and has been used in all the three systems described below.

Further development of Similix. Version 5 of this mature and much-copied partial evaluator for Scheme was recently released. It is now more user-friendly and significantly more efficient, both in the code it generates and in its transformation speed. Groups outside Denmark are publishing papers using Similix.

Development of C-mix. An M. Sc. thesis describing this system by L. O. Andersen was awarded a prize in 1992, and has been developed considerably since then. It has been extended and applied to various problems, a recent application being an "off-the-shelf" ray tracing program selected from a textbook on Computer Graphics. C-mix, by specializing the tracer to a fixed scene, resulted in a program that ran around twice as fast as the original. In this use of C-mix, a relatively small amount of tuning was sufficient to obtain the observed speedup.

The experiment is convincing since the "off-the-shelf" program had been engineered to be especially fast. Current work concerns strengthening the system, and making it more user-friendly, robust, and efficient.

Development of ML-mix. A thesis by L. Birkedal and M. Welinder on a partial evaluator for ML was awarded the highest possible grade. ML is a much larger, more complex, and widely used language than Similix's language SCHEME. This (rather sophisticated) ML partial evaluator builds to a high degree on experiences gathered from two research enviromnents: those of Similix and of ML.

ML-mix is quite new, but people in the US and England have already shown interest in using it (Bell Labs, and Tom Melham for the HOL application mentioned below). Much remains to be done but the core is robust and well-founded, so ML-mix preserves exactly its input program's semantics during transformation (essential for tools to be used by others). It should clearly be developed further.

Static program analyses. Several new program analyses were developed and implemented, including those used in Similix, C-mix, and ML-mix.

Future plans

We will continue to pursue our current research directions, and as well to look into the following leads, which indicate an increasing outside interest in partial evaluation.

The HOL system. (Higher Order Logic) is a theoretically based system with widespread practical and industrial use; but a system whose run time is often a limiting factor. It is currently used by DART members Aarhus and Aalborg. A central HOL figure, Tom Melham (Cambridge and Glasgow), recently lectured on the possibility of using ML-mix to speed the system up, and will visit DIKU this Fall to investigate its feasibility. This could be a significant application of partial evaluation, due to HOL's widespread use.

ERSEM is an EC-supported project for *computational ecological modeling* of the North Sea, with six European partners. Their new simulation package, written in C, has significant computational bottlenecks. Hand analysis indicates that partial evaluation could make many of their programs run at least twice as fast. ERSEM plans a pilot study with DIKU of partial evaluation automatically to improve their programs. This is interesting, as it tests the scalability of our methods to realistic problems not devised by ourselves.

3.6 OST: Operational Semantics, Types and Language Implementation (by M. Tofte)

The project proposal defined three areas of activities: the semantics of higher-order functors in ML, type inference and storage allocation, and the ML Kit. Progress in these areas has been as follows:

Higher-order Functors

We have explored the semantics of functor application in the presense of higher-order functors. A new style of inference rules for defining the semantics of modules is under development. The purpose is to simplify earlier approaches to modules semantics and to be able to define the operations of structure matching and propagation of sharing in a more operational way, by encoding them in a typed λ-calculus with dependencies at the type level formalized by an abstraction mechanism. This work is joint work with David B. MacQueen, from A.T.&T. Bell Labs in New Jersey.

Type Inference and Storage Allocation

We have generalized type region inference system to handle polymorphic recursion in regions. This means that different invocations of the same function can operate on different regions. For many functions, the effect of region inference is to re-discover regions that correspond to the usual stack frames in a normal stack-based implementation of block-structured languages. (However, unlike normal stack implementations, region inference is also able to handle higher-order functions and recursive data types.) We have developed several region inference algorithms and implemented one of them. Furthermore, we have measured the space requirements of sample programs, when they are compiled by the region inference algorithm and interpreted on an abstract machine. The results show that in most cases, space requirements are very modest indeed. They also show that region allocation and deallocation is extremely frequent and must be made very efficiently in a real implementation. We are currently looking at ways of achieving this. Also, we are writing a compiler from region-annotated programs to C and a runtime system in C. This software will be integrated with the ML Kit. When completed, this system will allow us to obtain figures for actual running times.

This work has been done by Mads Tofte and Lars Birkedal in collaboration with Jean-Pierre Talpin, Ecole des Mines, Paris.

The ML Kit

Version 1 of the ML Kit has been completed and made available via anonymous ftp. About 45 sites copied it. The documentation of the ML Kit has been made into a technical report at DIKU.

Students at DIKU have been using the ML Kit for various projects, for instance Peter Skadhauge has implemented a type checker based on semi-unification, by replacing some of the modules of the ML Kit by modules for generating and solving systems of equations and inequalities.

3.7 Other Topics

3.7.1 Types, Constraints, and Analysis (by M. Schwartzbach)

This year has seen the rounding off of several projects, as well as the initiation of new activities.

The joint work with Jens Palsberg on static analysis of object-oriented languages has been matured through cooperation with Ole Agesen of Stanford University. Our constraint-based algorithms have been extended and refined to produce a full-scale implementation for the SELF language being developed by SUN Microsystems. While SELF is generally recognized as a most challenging language, our static analyses now form the basis for several tools that are being integrated into the standard programming environment.

Jens Palsberg and Michael Schwartzbach have for some time now worked on object-oriented type systems. This activity is being summed up in our recent book, which will be published by Wiley in September 1993. Accompanying this advanced undergraduate text, a forthcoming journal article will contain the semantical foundations for our approach.

The work with Nils Klarlund on graph types has been continued. Guided by the principle that data types are invariants, we devise a logical and decidable framework for expressing global properties of a store consisting of records and pointers. Common properties, which seem to have called for a full Hoare logic beyond the reach of type checking and decidability, can now be expressed in a uniform descriptive language integrating types and program assertions. Our contributions are: a formalism for describing stores based on monadic second-order logic and our concept of pointer constraint; an extension of the method of semantic interpretation to show decidability of Hoare triples; and a sketch of a novel software methodology suggested by the extensive automated analysis that follows from our techniques.

A recent project, joint with Jens Palsberg and Bjorn Freeman-Benson, aims to perform static analysis of *constraint imperative programs*, in order to conservatively approximate freeness of variables and satisfiability of constraints.

3.7.2 Map Theory and Parallel Graph Reduction (by K. Grue)

Concerning map theory, DART has funded part of a travel to Paris to visit University VII. Here map theory was presented and discussed, and an outline of a simplified consistency proof was established. A joint paper with Chantal Berline on the simplified proof is under preparation.

Lars Lassen has defined a functional language and implemented a compiler which allows to execute programs on the parallel graph reducer. The language is very close to a subset of Common Lisp. Martin Funk Larsen has implemented a number of algorithms in that subset of Common Lisp and is now ready to run them on the parallel graph reducer. This allows benchmarking of the graph reducer on medium size software.

3.7.3 Activities at The Technical University (by B.S. Hansen)

Peter Sestoft is studying validity checking for a subset of Duration Calculus. One of the results is an implementation. The problem turned out to have very high complexity, so alternative approaches are being considered (joint work with M.R. Hansen, J.U. Skakkebæk, Zhou Chaochen).

A current student project supervised by Peter Sestoft concerns non-self-applicable partial evaluation of a subset of Standard ML.

Bo Stig Hansen is working jointly with Flemming M. Damm on type systems including a set theoretic union operator. This has resulted in decidability results for the subtype relation in a system with union, product, function space and recursion. A paper on type checking by generation of proof obligations has been accepted for the Workshop on Semantics of Specification Languages, October 1993; the proof obligations ensure absence of run-time type errors.

4 External and Internal Cooperation

Members of the project participate in a number of international research projects. Also members in Aarhus and Aalborg participate in the newly created *BRICS* project, a centre for Basic Research in Computer Science, funded by The Danish Research Foundation. Also when the need arises smaller workshops are arranged on specific topics.

4.1 Participation in International Projects

CONCUR2

Concurrency theory is important for the specification and verification of concurrent and distributed systems. CONCUR2 is an Esprit Basic Research Action with the specific aim at extending process algebra and logical calculi to incorporate real-time aspects, probabilistic non-determinism, value passing and infinite state spaces. CONCUR2 aims for a unified view on process algebra, and intends to design, specify and implement supporting software tools, and common formats and interfaces for such tools.

Partners of CONCUR2 besides Aalborg University are: CWI, Edinburgh, Eindhoven, INRIA Sophia Antipolis, Oxford, Sussex and SICS.

CLICS

Winskel's Esprit BRA "Categorical Logic in Computer Science" funds the category theorist and proof theorist Sergei Soloviev (on leave from the University of St. Petersburg) and will, in addition, employ Claudio Hermida this Fall.

Semantique

Semantique is an Esprit working group on Semantics-Based Program Manipulation. It has as partners Chalmers University (Gothenburg), University of Aarhus, University of Copenhagen, École Normale Supérieure (Paris), Imperial College of Science, Technology and Medicine (London), and the University of Glasgow. Imperial College is the coordinator.

Atlantique

Atlantique is a new Esprit working group whose purpose is to facilitate European-American joint research in the same area. It consists of the Europeans above, plus eight American research groups: Carnegie Mellon University, the City University of New York, Kansas State University, Northeastern University, New York University: Courant Institute, Oregon Graduate Institute, Stanford University, and Yale University. DIKU is the coordinator of Atlantique.

The Esprit grant to Atlantique includes travel by researchers and students among the European partners for research cooperation for shorter periods of rescarch cooperation (a few days to a few weeks). Further, we understand that NSF, the American *National Science Foundation*, will fund student visits from the US to the European partners.

COMPASS

Peter Mosses is a member of ESPRIT Basic Research Working Group 6112 COMPASS: *A Comprehensive Algebraic Approach to System Specification and Development*. The partners of this working group are the 19 major European sites for research on algebraic specification, located at universities and research institutes in Aarhus, Barcelona, Berlin, Braunschweig, Dresden, Edinburgh, Genoa, Lisbon, Munich, Nancy, Nijmegen, Oslo, Oxford, Paris, and Saarbrücken.

ML2000

Mads Tofte is a member of the ML2000 group. This group is designing a new programming language (ML2000), which is intended to be a successor to Standard ML by the year 2000. The group consists of Andrew Appel (Princeton University), Luca Cardelli (DEC SRC), Carl Gunter (University of Pennsylvania), Elsa Gunter (AT&T Bell Labs), Robert Harper (Carnegie Mellon University), David MacQueen (AT&T Bell Labs), John Mitchell (Stanford University), John Riecke (AT&T Bell Labs), John Reppy (AT&T Bell Labs), Mads Tofte (DIKU) and Xavier Leroy (Stanford University).

ProCoS II

The work by Peter Sestoft on validity checking for a subset of Duration Calculus has been done in close collaboration with the ESPRIT project ProCoS II.

LOMAPS

LOMAPS is an acronym for Logical and Operational Methods in the Analysis of Programs and Systems. It is an ESPRIT BRA project (starting fall 1993) on the development of advanced methods for analysing and verifying properties of multiparadigmatic programming languages like functional languages with concurrency primitives. The semantic framework will be operational semantics and for the specification of analyses various logical approaches will be explored.

The following sites participate in the project: Aarhus University (coordinator), Swedish Institute of Computer Science, Ecole des Mines, European Computer-Industry Research Centre, Cambridge University, University of Pisa, Ecole Normale Superieure and Ecole Polytechnique.

4.2 Mini-workshop on Program Analysis (by H.R. Nielson)

On October 2nd a small workshop devoted to program analysis was held at DAIMI. The purpose was to have a setting for presenting and discussing current work with two guests at DAIMI: *Olivier Danvy*, Kansas State University, USA and *Karoline Malmkjær*, Kansas State University, USA. The meeting was centered around four talks:

Karoline Malmkjær, Department of Information and Computer Sciences, Kansas State University: Predicting properties of residual programs

We present an analysis detecting structural properties of the results of partial evaluators for Scheme-like languages. The analysis is based on abstract interpretation of the generating extension produced by the partial evaluator. It has recently been extended to handle higher-order functions.

Jens Palsberg, DAIMI: Correctness of binding time analysis

A binding time analysis is correct if it always produces consistent binding time information. Consistency means that if the binding time information is used by a partial evaluator then the partial evaluator cannot commit errors such as applying something unknown to an argument. A sufficient and decidable condition for consistency, called well-annotatedness, was first presented by Gomard and Jones.

In this talk we present a weaker consistency condition. Our condition is decidable, subsumes the one of Gomard and Jones, and was first studied by Schwartzbach and the speaker.

We have proved that our condition indeed implies consistency. As a corollary, we get the first proof of correctness of the core of the binding time analyses of Bondorf, Consel and Mogensen. We have also proved that a whole family of partial evaluators will on termination have eliminated all "eliminable"-marked parts of an input which satisfies our condition. This generalizes a result of Gomard. Our development is for the pure λ-calculus with explicit binding time annotations.

Olivier Danvy, Department of Information and Computer Sciences, Kansas State University: On continuation-passing style

Continuation-passing style (CPS) is a programming style stressing the control flow of a program and allowing to express functionally operations that are non-functional a priori (jumps, exceptions, coroutines, etc.) Its essential property: a CPS program yields the same result independently of its evaluation strategy — call by name or call by value. Its domain of application vary surprisingly: proof of algorithm termination, semantics-directed compiling, compiler generation, parallelization, single-threading detection, double negation elimination in proof theory, and so on. This variety is reflected by a number of specifications of the CPS transformation — in striking contrast with the 3-lines & 2-passes (transformation + simplification) of the original literature [Fischer, Reynolds, Plotkin].

The goal of this talk is to enlighten CPS by going back to the sources, i.e. to the original 3-lines and 2-passes specification. By a simple transformation of this specification, we obtain a static separation of the results in terms of "essential" and "administrative" constructs (very much like Binding Time Analysis). By interpreting the essential constructions as syntax constructors and administrative constructions as executable code, we obtain a one-pass transformation algorithm.

We also considered the inverse transformation: the Direct Style transformation. This transformation and its proof suggest to reformulate the CPS transformation in 3 successive passes that appear considerably simpler than the 2 original passes.

Finally, we situate this development among related work. Part of this work is joint work with Andrzej Filinski.

Torben Lange, DAIMI: The correctness of an optimized code generation

For a lazy functional programming language with combinators, we first specify a standard semantics and a strictness analysis upon the language. Using the information from the analysis we can specify an optimized code generation avoiding delay closures otherwise generated around the argument to an application. By defining a layer of admissible predicates we are finally able to prove the correctness of the code generation with respect to the standard semantics.

5 Publications from DART (from March 1991 to August 1993)

Below we list the publications from the project. The overall criterion has been that publication took place in the period from March 1991 to August 1993, but we have marked with an asterisk those entries where almost all scientific work was performed before March 1991.

References

[Aceto1] L. Aceto and U. H. Engberg,
"Failures semantics for a simple process langnage with refinement,"
in *FST and TCS 11*, vol. 560 of *Lecture Notes in Computer Science*, pp. 89-108, Springer-Verlag, 1991. [DAR.T-129].

[Aceto2] L. Aceto and A. Ingólfsdóttir,
"A theory of testing for ACP,"
in *Proceedings of CONCUR'91*, Lecture Notes in Computer Science, 1991. [DART-l].

[Agerholm1] S. Agerholm,
"Mechanizing program verification in HOL,"
in *Proceedings of the International HOL users Meeting, Davis, California*, 1991. [DART-2].

[Agerholm2] S. Agerholm,
"Mechanizing program verification in HOL,"
Tech. Rep. DAIMI IR-111, Computer Science Dept., Aarhus Univ., 1992. [DART-197].

[Agerholm3] S. Agerholm,
"Domain theory in HOL,"
in *Proceedings of the 1993 International Meeting on Higher Order Logic Theorem Proving and Its Applications, Vancouver Canada, 11-13 August 1993*,
Lecture Notes in Computer Science, Springer-Verlag, 1993. [DART-198].

[Amtoft1] T. Amtoft,
"Properties of unfolding-based meta-level systems,"
in *Proceedings of the Symposium on Partial Evaluation and Semantics-Based Program Manipulation*,
SIGPLAN NOTICES vol. 26, no. 9, pp. 243-254, 1991. [DART-3].

[Amtoft2] T. Amtoft,
"Unfold/fold transformations preserving termination properties,"
in *4th International Symposium on Programming Language Implementation and Logic Programming (PLILP 92), Leuven, Belgium* (M. Bruynooghe and M. Wirsing, eds.), vol. 631 of *Lecture Notes in Computer Science*, pp. 187-201, Springer-Verlag, 1992. [DART-133].

[Amtoft3] T. Amtoft,
"Minimal thunkification,"
in *3rd International Workshop on Static Analysis (WSA '93), September 1993, Padova, Italy*, no. 724 in Lecture Notes in Computer Science, pp. 218-229, Springer-Verlag, 1993. [DART-168].

[Amtoft3] T. Amtoft,
Sharing of Computations.
PhD thesis, Computer Science Dept., Aarhus Univ., 1993. DAIMI-technical report PB-453. [DART-169].

[Amtoft4] T. Amtoft,
"Strictness types: An inference algorithm and an application,"
Technical Monograph DAIMI PB-448, Computer Science Dept., Aarhus Univ., 1993. [DART-167].

[Amtoft5] T. Amtoft and J. Larsson Träff,
"Partial memorization for obtaining linear time behavior of a 2DPDA,"
Theoretical Computer Science, vol. 98, pp. 347-356, 1992. [DART-4].

[Andersen1] H. R. Andersen,
"Local computation of alternating fixed-points,"
Tech. Rep. 260, Computer Laboratory, University of Cambridge, 1992. [DART-5].

[Andersen2] H. R. Andersen,
"Local computation of simultaneous fixed-points,"
Technical Monograph DAIMI PB-420, Computer Science Dept., Aarhus Univ., 1992. Submitted for publication. [DART-6].

[Andersen3] H. R. Andersen,
"Model checking and boolean graphs,"
in *Proceedings of ESOP'92, vol. 582 of Lecture Notes in Computer Science*, Springer-Verlag, 1992. [DART-7].

[AndersenWinskel1] H. R. Andersen and G. Winskel,
"Compositional checking of satisfaction,"
in *Proceedings of CAV, Aalborg*, vol. 575 of *Lecture Notes in Computer Science*, Springer-Verlag, 1991. Extended abstract. Journal version as DART-9. [DART-8].

[AndersenWinskel2] H. R. Andersen and G. Winskel,
"Compositional checking of satisfaction,"
Formal Methods in System Design, vol. 1, 1992. Extended abstract as DART-8. [DART-9].

[LAndersen] L. Andersen,
"Binding-time analysis and the taming of C pointers,"
in *Proc. of ACM Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, PEPM'93 (D. Schmidt, ed.), 1993. To appear. [DART-146].

[LOAndersen1] L.O. Andersen,
"C program specialization,"
Tech. Rep. 12/14, DIKU, University of Copenhagen, Denmark, Universitetsparken 1, DK-2100 Copenhagen East, 1992. (revised version). [DART-12].

[LOAndersen2] L.O. Andersen,
"Partial evaluation of C and automatic compiler generation (extended abstract),"
in *Proceedings of Compiler Constructions - 4th International Conference, CC '92* (U. Kastens and P. Pfahler, eds.), vol. 641 of *Lecture Notes in Computer Science*, pp. 251-257, Springer-Verlag, 1992. [DART-13].

[LOAndersen3] L.O. Andersen,
"Self-applicable C program specialization,"
in *Proceeding of PEPM'92: Partial Evaluation and Semantics-Based Program Manipulation*, pp. 54-61, 1992. Available as Technical Report from Yale University. [DART-10].

[LOAndersenGomard] L.O. Andersen and C. K. Gomard,
"Speedup analysis in partial evaluation: Preliminary results,"
in *Proc. of Workshop on Partial Evaluation and Semantics-Based Program Manipulation (PEPM'92)*, pp. 1-7, 1992. Available as Technical Report from Yale University. [DART-11].

[PHAndersen] P. H. Andersen,
"Partial evaluation applied to ray tracing."
Unpublished, 1993. [DART-l91].

[BirkedalRothwellTofteTurner] L. Birkedal, N. Rothwell, M. Tofte, and D. N. Turner,
"The ML kit (version 1),"

Tech. Rep. DIKU-report 93/14, Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen, 1993. [DART-194].

[BirkedalWelinder] L. Birkedal and M. Welinder,
"Partial evaluation of Standard ML,"
Master's thesis, DIKU, University of Copenhagen, Denmark, 1993. [DART-189].

[Bondorf1] A. Bondorf,
"Compiling laziness by partial evaluation,"
in *Functional Programming, Glasgow 1990. Workshops in Computing* (S. L. P. Jones, G. Hutton, and C. K. Holst, eds.), pp. 9-22, Springer-Verlag, 1991. [DART-14].*

[Bondorf2] A. Bondorf,
"Similix manual, system version 3.0,"
Tech. Rep. 91/9, DIKU, University of Copenhagen, Denmark, 1991. [DART-15].

[Bondorf3] A. Bondorf,
"Similix manual, system version 4.0."
Included in Similix distribution, 1991. [DART-16].

[Bondorf4] A. Bondorf,
"Improving binding times without explicit CPS-conversion,"
in *1992 ACM Conference on Lisp and Functional Programming. San Francisco, California*, pp. 1-10, 1992. [DART-17].

[Bondorf5] A. Bondorf,
Similix 5.0 Manual.
DIKU, University of Copenhagen, Denmark, 1993. Included in Similix distribution, 82 pages. [DART-174].

[BondorfJorgensen1] A. Bondorf and J. Jørgensen,
"Efficient analyses for realistic off-line partial evaluation,"
Journal of Functional Programming, special issue on partial evaluation, vol. 11, 1993. [DART-175].

[BondorfJorgensen2] A. Bondorf and J. Jørgensen,
"Efficient analyses for realistic off-line partial evaluation,"

Tech. Rep. 93/4, DIKU, University of Copenhagen, Denmark, 1993. [DART-145l.

[BondorfPalsberg] A. Bondorf and J. Palsberg,
"Compiling actions by partial evaluation,"
in *FPCA'93, Conference on Functional Programming and Computer Architecture, Copenhagen, Denmark*, pp. 308-317, ACM, 1993. [DART-176].

[Borjesson] A. Børjesson,
"Distinguishing properties and model checking in TAV."
In preparation, 1992. [DART-18].

[BorjessonLarsen] A. Børjesson and K. G. Larsen,
"Equation solving using TAV."
In preparation, 1992. [DART-19].

[BorjessonLarsenSkou] A. Børjesson, K. G. Larsen, and A. Skou,
"Generality in design and compositional verification using TAV,"
In *Proceedings of FORTE'92,* 1992. To appear. [DART-20].

[CamilleriWinskel] J. A. Camilleri and G. Winskel,
"CCS with priority choice,"
In *Proceedings of LICS'91*, 1991. To appear in Information and Computation. [DART-21].

[Cerans] K. Cerans,
"Decidability of bisimulation equivalences for parallel timer processes,"
in *Proceedings of CAV'92*, Lecture Notes in Computer Science, 1992. [DART-22].

[CeransGodskesenLarsen] K. Cerans, J. Godskesen, and K. Larsen,
"Timed modal specifications-theory and tools,"
tech. rep., Aalborg University, Dept. of Math. and Comp. Sc., 1993. [DART-154].

[ChristensenHuttelStirling] S. Christensen, H. Huttel, and C. Stirling,
"Bisimulation equivalence is decidable for all context-free processes,"
in *Proceedings of CONCUR'92, vol. 630 of lecture Notes in Computer Science*, Springer-Verlag, 1992. ECS-LFCS-92. [DART-23].

[Dybkjaer] H . Dybkjær,
Category Theory, Types, and Programming Languages.
PhD thesis, DIKU, University of Copenhagen, Denmark, 1991. vi+146pp. Available as DIKU report 91/11. [DART-24].*

[DybkjaerMelton] H. Dybkjær and A. Melton,
"Comparing Hagino's categorical programming language and typed lambda calculi,"
Theoretical Computer Science, vol. 111, pp. 145-189, 1993. [DART-25].*

[EngbergGronningLamport] U. Engberg, P. Grønning, and L. Lamport,
"Mechanical verification of concurrent systems with TLA."
To appear in the Proceedings of the Fourth International Workshop on Computer-Aided Verification, 1992. [DART-26].

[EngbergWinskel] U. H. Engberg and G. Winskel,
"Completeness results for linear logic on Petri Nets."
Submitted to MFCS'93, Gdańsk, Poland, August 30 - September 3, 1993.
Full version will appear as DAIMI PB, 1993. [DART-153].

[Gammelgaard1] A. Gammelgaard,
"Constructing simulations chunk by chunk,"
Internal Report DAIMI IR106, Computer Science Dept., Aarhus Univ., 1991. [DART-27].*

[Gammelgaard2] A. Gammelgaard,
"Reuse of invariants in proofs of implementation,"
Technical Monograph DAIMI PB-360, Computer Science Dept., Aarhus Univ., 1991. [DART-28].*

[GammelgaardLovengreenRumpSogaardAndersen] A. Gammelgaard, H. H. Løvengreen, C. 0. Rump, and J. F. Søgaard-Andersen,
"Base system verification."
Submitted for publication, 1992. IDART-29].*

[GlindtvadNielson] K. Glindtvad and H. R. Nielson,
"Correctness preserving transformations on a multipass OCCAM compiler,"

Technical Monograph DAIMI PB-368, Computer Science Dept., Aarhus Univ., 1991. [DART-30].

[GluckKlimov] R. Glück and A. V. Klimov,
"Occam's razor in metacomputation: the notion of a perfect process tree,"
in *Static Analysis. Proceedings* (P. Cousot, M. Falaschi, G. File, and A. Rauzy, eds.), vol. 724 of *Lecture Notes in Computer Science*, pp. 112-123, Springer-Verlag, 1993. [DART-177].

[GodskesenLarsen1] J. C. Godskesen and K. G. Larsen,
"Real time calculi and expansion theorems (extended abstract),"
in *Proceedings of the First North American Process Algebra Workshop*, Dept. of Comp.Sc, The Johns Hopkins University, 1992. [DART-157].

[GodskesenLarsen2] J. C. Godskesen and K. G. Larsen,
"Real time calculi and expansion theorems,"
in *Proceedings of FST- TCS'92*, vol. 652 of *Lecture Notes in Computer Science*, Springer-Verlag, 1992. [DART-31].

[Gomard1] C. K. Gomard,
Program Analysis Matters.
PhD thesis, DIKU, University of Copenhagen, Denmark, 1991. DIKU report 91/17. [DART-32].

[Gomard2] C. K. Gomard,
"A self-applicable partial evaluator for the lambda calculus: Correctness and pragmatics,"
TOPLAS, vol. 14, no. 2, pp. 147-172, 1992. [DART-33].*

[GomardJones] C. K. Gomard and N. D. Jones,
"A partial evaluator for the un-typed lambda calculus,"
Journal of Functional Programming, vol. 1, pp. 21-69, 1991. [DART-34].*

[GomardSestoft1] C. K. Gomard and P. Sestoft,
"Evaluation order analysis for lazy data structures,"
in *Functional Programming, Glasgow Workshop 1991* (Heldal, Holst, and Wadler, eds.), pp. 112-127, Springer-Verlag, 1991. [DART-35].

[GomardSestoft2] C. K. Gomard and P. Sestoft,
"Globalization and live variables,"

in *Proceedings of the Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, SIGPLAN NOTICES vol. 26, no. 9, pp. 166-177, ACM Press, 1991. [DART-36].

[GomardSestoft3] C. K. Gomard and P. Sestoft,
"Path analysis for lazy data structures,"
in *Programming Language Implementation and Logic Programming, 4th International Symposium, PLILP '92, Leuven, Belgium* (M. Bruynooghe and M. Wirsing, eds.), vol. 631 of Lecture Notes in Computer Science, pp. 54-68, Springer-Verlag, 1992. [DART-37] .

[GrooteHuttel] J. F. Groote and H. Hüttel,
"Undecidable equivalences for basic process algebra,"
Tech. Rep. ECS-LFCS-91-169, Department of Computer Science, University of Edinburgh, 1991. [DART-38].

[Grue] K. Grue,
"Map theory,"
Theoretical Computer Science, vol. 102, pp. 1-133, 1991. [DART-39].

[GurrBrown1] D. Gurr and C. Brown,
"Relations and non-commutative linear logic,"
Technical Monograph DAIMI PB-372, Computer Science Dept., Aarhus Univ., 1991. [DART-40].

[GurrBrown2] D. Gurr and C. Brown,
"A representation therorem for quantales."
To appear in Jounal of Pure & Applied Algebra, 1992. [DART-41].

[HankinMetayerSands] C. Hankin, D. L. Metayer, and D. Sands,
"A parallel programming style and its algebra of programs,"
in *Proceeding of Parallel Architectures and Languages Europe* (PARLE), vol. 694 of *Lecture Notes in Computer Science*, pp. 367-378, Springer-Verlag, 1993. To appear. [DART-178].

[Hannan1] J. Hannan,
"Making abstract machines less abstract,"
in *Proceedings of the 5th ACM Conference on Functional Programming and Computer Architecture* (J. Hughes, ed.), vol. 523 of *Lecture Notes in Computer Science*, pp. 618-635, Springer-Verlag, 1991. [DART-42].

[Hannan2] J. Hannan,
"Staging transformations for abstract machines,"
in *Proceedings of the ACM SIGPLAN Symposium on Partial Evaluation and Semantics Based Program Manipulation* (P. Hudak and N. Jones, eds.), pp. 130-141, ACM Press, 1991. [DART-43].

[Hannan3] J. Hannan,
"Implementing λ-calculus reduction strategies in extended logic programming languages,"
in *Proceedings of the Second Workshop International Workshop on Extensions of Logic Programming* (L.-H. Eriksson, L. Hallnäs, and P. Schroeder-Heister, eds.), no. 596 in Lecture Notes in Computer Science, pp. 193-219, Springer-Verlag, 1992. [DART 44].

[HannanMiller] J. Hannan and D. Miller,
"From operational semantics to abstract machines,"
Mathematical Structures in Computer Science, vol. 2, 1992. To appear. [DART-45].

[HannanPfenning] J. Hannan and F. Pfenning,
"Compiler verification in LF,"
in *Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science* (A. Scedrov, ed.), IEEE Computer Society Press, 1992. [DART-46].

[HavelundLarsen] K. Havelund and K. G. Larsen,
"The fork calculus,"
tech. rep., Aalborg University, Dept. of Math. and Comp. Sc., 1993. [DART-159].

[Henglein1] F. Henglein,
"Efficient type inference for higher-order binding-time analysis,"
in FPCA (J. Hughes, ed.), vol. 523 of *Lecture Notes in Computer Science*, pp. 448-472, 5th ACM Conference, Cambridge, MA, USA, Springer-Verlag, 1991. [DART-47].

[Henglein2] F. Henglein,
"Type inference with polymorphic recursion,"
ACM Transactions on Programming Languages and Systems (TOPLAS), 1991. [DART-48].*

[Henglein3] F. Henglein,
"Dynamic typing,"
in *Proc. European Symp. on Programming (ESOP), Rennes, France* (B. Krieg-Brückner, ed.), vol. 582 of *Lecture Notes in Computer Science*, pp. 233-253, Springer-Verlag, 1992. [DART-49].

[Henglein4] F. Henglein,
"Global tagging optimization by type inference,"
in *Proc. 1992 ACM Conf. on LISP and Functional Programming (LFP), San Francisco, California*, ACM Press, 1992. [DART-50].

[Henglein5] F. Henglein,
"Dynamic typing: Syntax and proof theory,"
Science of Computer Programming, 1993. Special Issue on European Symposium on Programming 1992 (to appear). [DART-179].

[HengleinJorgensen] F. Henglein and J. Jørgensen,
"Formally optimal boxing."
accepted for POPL'94, 1993. [DART-192].

[HolmerLarsenYi] U. Holmer, K. G. Larsen, and W. Yi,
"Decidability of bisimulation equivalence between regular timed processes,"
in *Proceedings of CAV'91*, vol. 575 of *Lecture Notes in Computer Science*, 1992. [DART-51].

[HolstGomard] C. K. Holst and C. K. Gomard,
"Partial evaluation is fuller laziness,"
in *Proceedings of Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, SIGPLAN NOTICES vol. 26, no. 9, pp. 223-233, ACM Press, 1991. [DART-52].

[HudakJones] P. Hudak and N. Jones,
eds., *Proceedings of the Symposium on Partial Eualuation and Semantics-Based Program Manipulation (PEPM), New Haven, Connecticut*,
Sponsored by the ACM Special Interest Group SIGPLAN, in cooperation with IFIP, ACM Press, 1991. [DART-53].

[Huttel1] H. Hüttel,
"Decidability, behavioural equivalences and infinite transition graphs," ECS-LFCS-91-191, Department of Computer Science, University of Edinburgh, 1992. [DART-54].

[Huttel2] H. Hüttel,
"Silence is golden: Branching bisimilarity is decidable for context-free processes,"
in *Proceedings of CAV91*, vol. 575 of *Lecture Notes in Computer Science*, Springer-Verlag, 1992. The full version is available as Tech. Rep. ECS-LFCS-91-173, Department of Computer Science, University of Edinburgh. [DART-55].

[HuttelChristensenStirling] H. Hüttel, S. Christensen, and C. Stirling,
"Bisimulation equivalence is decidable for all context-free processes,"
in *CONCUP 92* (W. Cleaveland, ed.), vol. 630 of *Lecture Notes in Computer Science*, pp. 138-147, Springer-Verlag, 1992. [DART-156].

[HuttelLarsen] H. Hüttel and K. G. Larsen,
"A dynamic type system for higher-order processes."
Aalborg Technical report R93-2003, 1992. [DART-155]

[HuttelStirling] H. Hüttel and C. Stirling,
"Actions speak louder than words: Proving bisimilarity for context-free processes,"
in *Proceedings of 6th Annual Symposium on Logic in Computer Science (LICS 91)*, pp. 376-386, IEEE Computer Society Press, 1991. [DART-56].

[IngolfsdottirSteffen] A. Ingolfsdottir and B. Steffen,
"Characteristic formulae,"
Information and Computation, 1992. To appear. [DART-57].

[IngolfsdottirThomsen] A. Ingolfsdottir and B. Thomsen,
"Semantics models for CCS with values,"
In *Proceedings of the Workshop on Concurrency, Båstad, Sweden*, 1991. [DART-58].

[Jensen] C. T. Jensen,
"The concurrency workbench with priorities,"

in *Proceedings of CAV'91, Aalborg, Denmark,* vol. 575 of *Lecture Notes in Computer Science*, Springer-Verlag, 1992. [DART-59].

[Jones1] N. D. Jones,
"Computer implementation and applications of Kleene's s-m-n and recursive theorems,"
in *Lecture Notes in Mathematics, Logic From Computer Science* (Y. Moschovakis, ed.), pp. 243-263, Springer-Verlag, 1991. [DART-60].*

[Jones2] N. D. Jones,
"Efficient algebraic operations on programs,"
in *Preliminary Proceedings, University of Iowa* (T. Rus, ed.), 1991. Accepted for publication by Theoretical Computer Science, 1992. [DART-61].

[Jones3] N. D. Jones, ed.,
Selected Papers of ESOP'90. Science of Computer Programming. Volume 17, numbers 1-3, pages 1-271, Elsevier, 1991. [DART-62].*

[Jones4] N. D. Jones,
"Static semantics, types and binding time analysis,"
in *Images of Programming* (D. Bjørner and V. Kotov, eds.), North-Holland, 1991. Further appeared in Theoretical Computer Science 90 (1991), pages 95-118. [DART-63].

[Jones5] N. D. Jones,
"Constant time factors *do* matter,"
in *STOC '93. Symposium on Theory of Computing* (S. Homer, ed.), ACM, 1993. [DART-151] .

[Jones6] N. D. Jones,
"Partial evaluation and the generation of program generators."
Accepted to appear in Communications of The ACM, 1993. [DART-136].

[Jones7] N. D. Jones, ed.,
Special issue on Partial Evaluation, 1993 (Journal of Functional Programming, vol. ?, no. ?),
Cambridge University Press, 1993. 5 accepted papers being revised; to be printed in 1993. [DART-152].

[JonesGomardSestoft] N. D. Jones, C. Gomard, and P. Sestoft,
Partial Evaluation and Automatic Program Generation.
International Series in Computer Science: Prentice Hall International, 1993. Series editor C.A.R. Hoare. ISBN number 0-13-20249-5 (pbk). [DART-180].

[JonssonLarsen1] B. Jonsson and K. G. Larsen,
"On the complexity of equation solving in process algebra,"
in *Proceedings of TAPSOFT'91*, vol. 493 of *Lecture Notes in Computer Science*, Springer-Verlag, 1991. [DART-64].

[JonssonLarsen2] B. Jonsson and K. G. Larsen,
"Specification and refinement of probabilistic processes,"
in *Proceedings of LICS'91*, 1991. [DART-65].

[Jorgensen1] J. Jørgensen,
"Compiler generation by partial evaluation,"
Master's thesis, DIKU, University of Copenhagen, Denmark, 1991. [DART-66].

[Jorgensen2] J. Jørgensen,
"Generating a compiler for a lazy language by partial evaluation,"
in *Nineteenth Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. Albuquerque, New Mexico*, pp. 258-268, 1992. [DART-67].

[KlarlundSchwartzbach] N. Klarlund and M. I. Schwartzbach,
"Graph types,"
in *Proc. POPL '93, Principles of Programming Langauges*, (Charleston), ACM, 1993. [DART-143].

[KozenPalsbergSchwartzbach1] D. Kozen, J. Palsberg, and M. I. Schwartzbach,
"Efficient inference of partial types,"
in *Proc. FOCS'92, 33rd IEEE Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania*, 1992. Also available as Tech. Rep. DAIMI PB-394, Computer Science Department, Aarhus University. [DART-68].

[KozenPalsbergSchwartzbach2] D. Kozen, J. Palsberg, and M. I. Schwartzbach,
"Efficient recursive subtyping,"
Technical Monograph DAIMI PB-405, Computer Science Dept., Aarhus Univ., 1992. [DART-69].

[Krishnan1] P. Krishnan
"Distributed CCS,"
in *Proc. CONCUR-91*, vol. 527 of *Lecture Notes in Computer Science*, pp. 393-407, Springer-Verlag, 1991. [DART-70].*

[Krishnan2] P. Krishnan,
"A model for real-time systems,"
in *Proc. MFCS'91*, vol. 520 of *Lecture Notes in Computer Science*, Springer-Verlag, 1991. [DART-71].*

[Krishnan3] P. Krishnan,
"Real-time action,"
in *Proc. Euromicro Workshop on RealTime Systems*, l991.[DART-72].*

[Krishnan4] P. Krishnan, "A semantics for multiprocessor systems,"
in *ESOP'92, Proc. European Symposium on Programming, Rennes*, vol. 582 of *Lecture Notes in Computer Science*, Springer-Verlag, 1992.[DART-73].

[KrishnanMosses] P. Krishnan and P. D. Mosses,
"Specifying asynchronous transfer of control,"
in *RTFT'92, Proc. Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems, Delft*, vol.571 of *Lecture Notes in Computer Science*, Springer-Verlag, 1992.[DART-74].

[Lange] T. P. Lange,
"The correctness of an optimized code generation,"
in *PEPM'93, ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, pp.167-178,ACM, 1993. [DART-132].

[Larsen1] K. G. Larsen,
"Proof systems for satisfiability in Hennessy-Milner logic with recursion,"
Theoretical Computer Science, vol.72,1990. [DART-75].

[Larsen2] K. G. Larsen,
"The expressive power of implicit specifications,"
in *Proceedings of ICALP'91*, vol. 510 of *Lecture Notes in Computer Science*, Springer-Verlag, 1991. [DART-76].

[Larsen3] K. G. Larsen,
"Efficient local correctness checking,"
In *Proceedings of CAV'92*. To appear in *Lecture Notes in Computer Science.*, 1992. [DART-77].

[LarsenSkou1] K. G. Larsen and A.Skou,
"Bisimulation through probabilistic testing,"
Information and Computation, vol. 94, no. 1, 1991. [DART-78].

[LarsenSkou2] K. G. Larsen and A. Skou,
"Compositional verification of probabilistic processes,"
in *Proceedings of CONCUR '92*, vol. 630 of *Lecture Notes in Computer Science*, Springer-Verlag, 1992.[DART-79].

[LarsenXinxin] K. G. Larsen and L. Xinxin,
"Compositionality through an operational semantics of contexts,"
Journal of Logic and Computation, vol. 1, no. 6, pp. 761-795, 1991.[DART-80].*

[LarsenYi] K. G. Larsen and W. Yi,
"Time abstracted bisimulation: Implicit specifications and decidability,"
tech. rep., Aalborg University, Dept. of Math. and Comp.Sc., 1993. [DART-158].

[LarsenSchmidtSchwartzbach] K. S. Larsen, E. M. Schmidt, and M. I. Schwartzbach,
"A new formalism for relational algebra,"
Information Processing Letters, vol. 41, no. 3, 1992. [DART-81].

[Malmkjaer1] K. Malmkjær,
"Predicting properties of residual programs,"
in *PEPM'92, ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation* (C. Consel, ed.), pp. 8-13, 1992. Available as Technical Report YALEU/DCS/RR-909 from Yale University. [DART 137].

[Malmkjaer2] K. Malmkjær,
"Towards efficient partial evaluation,"
in *ACM SIG-PLAN Symposium on Partial Evaluation and Semantics Based Program Manipulation*, 1993. To appear. [DART-150].

[MarquardSteensgaard] M. Marquard and B. Steensgaard,
"Partial evaluation of an object-oriented imperative language,"
Master's thesis, University of Copenhagen, Department of Computer Science, Universitetsparken 1, 2100 Copenhagen O., Denmark, 1992. [DART-138].

[MarriotSondergaardJones] K. Marriot, H. Søndergaard, and N. Jones,
"Denotational abstract intrepretation of logic programs,"
ACM Transactions on Programming Languages and Systems, 1991. To appear. [DART-82].

[MilnerTofte] R. Milner and M. Tofte,
"Co-induction in relational semantics,"
Theoretical Computer Science, vol. 87, pp. 209-220, 1991. [DART-144].*

[Mogensen1] T. Æ. Mogensen,
"Efficient self-interpretation in lambda calculus,"
Functional Programming, vol. 2, pp. 345-364, 1992. [DART-149].

[Mogensen2] T. Æ. Mogensen,
"Efficient self-interpretation in lambda calculus."
Version of DART-149 with full proofs, 1992. [DART-83].

[Mogensen3] T. Æ. Mogensen,
"Self-applicable partial evaluation for pure lambda calculus,"
in *ACM SIGPLAN Workshop on Partial Evaluation and Semantics-based Program Manipulation* (C. Consel, ed.), pp. 116-121, ACM, Yale University, 1992. [DART-84].

[Mogensen4] T. Æ. Mogensen,
"Constructor specialization,"
in *ACM Symposium on Partial Eualuation and Semantics-Based Program Manipulation* (D. Schmidt, ed.), ACM press, 1993. To appear. [DART-148].

[MogensenBondorf] T. Æ. Mogensen and A. Bondorf,
"Logimix: a self-applicable partial evaluator for Prolog,"
in *Proceedings of LOPSTR 92. Workshops in Computing* (K.-K. Lau and T. Clement, eds.), Springer-Verlag, 1993. ISBN: 3-540-19806-7. [DART-85].

[Mosses1] P. D. Mosses,
"A practical introduction to denotational semantics,"
in *Formal Description of Programming Concepts* (E. J. Neuhold and M. Paul, eds.), IFIP State-of-the-Art Report, pp. 1-49, Springer-Verlag, 1991. [DART-87].

[Mosses2] P. D. Mosses,
Action Semantics.
No. 26 in Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1992. [DART-88].

[Mosses3] P. D. Mosses,
"On the action semantics of concurrent programming languages,"
Technical Monograph DAIMI PB-424, Computer Science Dept., Aarhus Univ., 1992. Accepted for publication in Proc. of the REX Workshop on Semantics – Foundations and Applications, Beekbergen, The Netherlands, June 1992. [DART-141].

[Mosses4] P. D. Mosses,
"The operational semantics of action notation,"
Technical Monograph DAIMI PB-418, Computer Science Dept., Aarhus Univ., 1992. Submitted for Proc. 8th Workshop on Mathematical Foundations of Programming Semantics, which is to appear as a special issue of TCS, 1993. [DART-140].

[Mosses5] P. D. Mosses,
"The use of sorts in algebraic specifications,"
in *Recent Trends in Data Type Specification* (M. Bidoit and C. Choppy, eds.), vol. 655 of *Lecture Notes in Computer Science*, Springer-Verlag, 1992. [DART-139].

[Mosses6] P. D. Mosses,
"An introduction to action semantics,"
in *Logic and Algebra of Specification, Proc. Marktoberdorf Summer*

School 1991, vol. 94 of *NATO ASI Series F*, pp. 247-288, Springer-Verlag, 1993. [DART-86]

[MossesWatt] P. D. Mosses and D. A. Watt,
"Pascal action semantics, version 0.6."
Available by FTP from ftp.daimi.aau.dk in pub/action/pascal, 1993. [DART-162].

[Mossin1] C. Mossin,
"Similix binding time debugger manual, system version 4.0."
Included in Similix distribution, 1991. [DART-89].

[Mossin2] C. Mossin,
"Partial evaluation of general parsers (extended abstract)," in *1993 ACM Symposium on Partial Evaluation and Semantics-Based Program Manipulation* (D. Schmidt, ed.), 1993. To appear. [DART-147].

[Mossin3] C. Mossin,
"Polymorphic binding time analysis,"
Master's thesis, DIKU, University of Copenhagen, Denmark, 1993. [DART-181].

[Musicante] M. A. Musicante,
"The Sun RPC language semantics,"
in *Proc.18th Latin-American Conference for Informatics*, 1992. [DART-90].

[MusicanteMosses] M. A. Musicante and P. D. Mosses,
"Communicative action notation with shared storage,"
Technical Monograph DAIMI PB-452, Computer Science Dept., Aarhus Univ., 1993. [DART-163].

[MycroftRosendahl] A. Mycroft and M. Rosendahl,
"Minimal function graphs are not instrumented,"
in *WSA '92 Bordeaw 1992*, pp. 60-67, Bigre, Irisa, Rennes, France, 1992. [DART-91].

[NielsonNielson1] F. Nielson and H. R. Nielson,
"Forced transformations of OCCAM programs,"
Information and Software Technology, vol. 34, no. 2, 1992. [DART-92].*

[NielsonNielson2] F. Nielson and H. R. Nielson,
"Layered predicates,"
in *REX'92 workshop "Semantics - foundations and applications"*, vol. 666 of *Lecture Notes in Computer Science*, pp. 425-456, Springer-Verlag, 1992. [DART-131].

[NielsonNielson3] F. Nielson and H. R. Nielson,
"The tensor product in Wadler's analysis of lists,"
in *Proceedings of ESOP'92*, vol. 582 of *Lecture Notes in Computer Science*, Springer-Verlag, 1992. [DART-93].

[NielsonNielson4] F. Nielson and H. R. Nielson,
Two-Level Bunctional Languages, vol. 34 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1992. [DART-94].

[NielsonNielson5] F. Nielson and H. R. Nielson,
"Finiteness conditions for strictness analysis,"
in *WSA Proceedings*, vol. 724 of *Lecture Notes in Computer Science*, pp. 194-205, Springer-Verlag, 1993. [DART-166].

[NielsonNielson6] F. Nielson and H. R. Nielson,
"From CML to process algebras,"
Technical Monograph DAIMI PB-433, Computer Science Dept., Aarhus Univ., 1993. [DART-165].

[NielsonNielson7] F. Nielson and H. R. Nielson,
"From CML to process algebras (extended abstract),"
in *Concur'93*, vol. 715 of *Lecture Notes in Computer Science*, pp. 495-508, Springer-Verlag, 1993. [DART-164].

[FNielson1] F. Nielson, (ed.),
"Design, analysis and reasoning about tools: Abstracts from the first workshop,"
Technical Monograph DAIMI PB-367, Computer Science Dept., Aarhus Univ., 1991. [DART-95].

[NielsonNielson8] H. R. Nielson and F. Nielson,
"Using transformations in the implementation of higher-order functions,"

Journal of Functional Programming, vol. 1, no. 4, pp. 459-494, 1991. [DART-96].*

[NielsonNielson9] H. R. Nielson and F. Nielson,
"Bounded fixed-point iteration,"
J. Logic Computat., vol. 2, no. 4, pp. 441-464, 1992. [DART-134].

[NielsonNielson10] H. R. Nielson and F. Nielson,
"Bounded fixed point iteration,"
in *Proceedings of POPL'92*, ACM Press, 1992. This is an "extended abstract" of DART-134. [DART-97].

[NielsonNielson11] H. R. Nielson and F. Nielson,
"Finiteness conditions for fixed point iteration"
in *Proceedings of Lisp and Functional Programming*, 1992. The full version appeared as DAIMI PB-384, Aarhns University. lDART-98].

[NielsonNielson12] H. R. Nielson and F. Nielson,
Semantics with Applications: A Formal Introduction. Wiley, 1992. [DART-99].

[NielsonNielson13] H. R. Nielson and F. Nielson,
"Iterative algorithms for fixed point computation,"
Technical Monograph DAIMI PB-441, Computer Science Dept., Aarhus Univ., 1993. [DART-160].

[NielsonNielsonPilegaardLange] H. R. Nielson, F. Nielson, A. Pilegaard, and T. Lange,
"The PSI system,"
Internal Report DAIMI IR-114, Computer Science Dept., Aarhus Univ., 1992. [DART-130].

[FNielson2] F. Nielson, (ed.),
"Design, analysis and reasoning about tools: Abstracts from the second workshop,"
Technical Monograph DAIMI PB-417, Computer Science Dept., Aarhus Univ., 1992. [DART-100].

[OxhojPalsbergSchwartzbach] N. Oxhøj, J. Palsberg, and M. I. Schwartzbach,

"Making type inference practical,"
in *Proc. ECOOP'92, Sixth European Conference on Object-Oriented Programming, Utrecht, The Netherlands*, vol. 615 of *Lecture Notes in Computer Science*, pp. 329-349, Springer-Verlag, 1992. [DART-101].

[Palsberg1] J. Palsberg,
"An automatically generated and provably correct compiler for a subset of Ada,"
in *ICCL'92, Proc. Fourth IEEE Int. Conf. on Computer Languages, Oakland*, pp. 117-126, IEEE, 1992. [DART-102].

[Palsberg2] J. Palsberg,
Provably Correct Compiler Generation.
PhD thesis, Aarhus University, 1992. [DART-103].

[Palsberg3] J. Palsberg,
"A provably correct compiler generator,"
in *ESOP'92, Proc. European Symposium on Programming, Rennes*, vol. 582 of *Lecture Notes in Computer Science*, pp. 418-434, Springer-Verlag, 1992. [DART-104].

[PalsbergKozenSchwartzbach] J. Palsberg, D. Kozen, and M. I. Schwartzbach,
"Efficient recursive subtyping,"
in *Proc. POPL'93, Principles of Programming Langauges*, (Charleston), ACM, 1993. [DART-142].

[PalsbergSchwartzbach1] J. Palsberg, and M. I. Schwartzbach,
"Object-oriented type inference,"
in *Proc. OOPSLA '91, ACM SIGPLAN Sixth Annual Conference on Object-Oriented Programming Systems, Languages and Applications, Phoenix, Arizona*, pp. 146-161, 1991. [DART-105].

[PalsbergSchwartzbach2] J. Palsberg, and M. I. Schwartzbach,
"Static typing for object-oriented programming,"
Technical Monograph DAIMI PB-355, Computer Science Dept., Aarhus Univ., 1991. [DART-106].

[PalsbergSchwartzbach3] J. Palsberg, and M. I. Schwartzbach,
"Types, inheritance and assignments,"

1991. Workshop held at ECOOP'91 in Geneva, Switzerland, July 1991. The collection of position papers is available from Computer Science Department, Aarhus University as PB-357. [DART-107] .

[PalsbergSchwartzbach4] J. Palsberg, and M. I. Schwartzbach,
"What is type-safe code reuse?,"
in *Proc. ECOOP'91, Fifth European Conference on Object-Oriented Programming, Geneva, Switzerland*, vol. 512 of *Lecture Notes in Computer Science*, pp. 325-341, Springer-Verlag, 1991. [DART-108].

[PalsbergSchwartzbach5] J. Palsberg, and M. I. Schwartzbach,
"Safety analysis versus type inference,"
Technical Monograph DAIMI PB-389, Computer Science Dept., Aarhus Univ., 1992. [DART-109].

[PalsbergSchwartzbach6] J. Palsberg, and M. I. Schwartzbach,
"Safety analysis versus type inference for partial types,"
Information Processing Letters, vol. 43, pp. 175-180, 1992. Also available as Tech. Rep. DAIMI PB-404, Computer Science Department, Aarhus University. [DART-110].

[PalsbergSchwartzbach7] J. Palsberg, and M. I. Schwartzbach,
"Three discussions on object-oriented typing,"
ACM SIGPLAN OOPS *Messenger*, vol. 3, no. 2, pp. 31-38, 1992. [DART-111].

[RehofSorensen] J. Rehof and M. H. Sørensen,
"The λ_δ-calculus."
Technical report from DIKU, 1993. [DART-182].

[Rose1] K. H. Rose,
"Explicit cyclic substitutions,"
in *CTRS '92–3rd International Workshop on Conditional Term Rewriting Systems* (M. Rusinowitch and J.-L. Rémy, eds.), no. 656 in Lecture Notes in Computer Science, (Pont-a-Mousson, France), pp. 36-50, Springer-Verlag, 1992. Also available as DIKU semantics note D-143. [DART-113] .

[Rose2] K. H. Rose,
"GOS - graph operational semantics,"

M.Sc.-thesis 92-1-9, DIKU, University of Copenhagen, Denmark, 1992. Awarded a silver medal by the University of Copenhagen. [DART-114].

[Rose3] K. H. Rose,
"Graph-based operational semantics for lazy functional languages," in *Term Graph Rewriting: Theory and Practice* (M. R. Sleep, M. J. Plasmeijer, and M. van Eekelen, eds.), (Nijmegen, Holland), pp. 203-225, John Wiley & Sons, 1992. [DART-112].

[Rose4] K. H. Rose,
"Graph-based operational semantics of a lazy functional languages," in *Term Graph Rewriting: Theory and Practice* (M. R. Sleep, M. J. Plasmeijer, and M. C. D. J. van Eekelen, eds.), ch. 22, pp. 234-247. John Wiley & Sons. 1992. [DART-183].

[Rose5] K. H. Rose,
"How to typeset pretty diagram arrows with TEX—design decisions used in XY-pic,"
in *EuroTEX'92—Proceedings of the 7th European TEX Conference* (J. Zlatuska, ed.), (Prague, Czechoslovakia), pp. 183 190, Czechoslovak TEX Users Group, 1992. [DART-173].

[Rose6] K. H. Rose,
"Explicit cyclic substitution."
Selected for special issue of Journal of Symbolic Computation, 1993. [DART-184].

[Rose7] K. H. Rose,
"Explicit recursion."
Available as DIKU semantics note D-147, 1993. [DART-185].

[Rosendahl1] M. Rosendahl,
Abstract Interpretation and Attribute Grammars.
PhD thesis, Cambridge University, 1991. [DART-172].

[Rosendahl2] M. Rosendahl,
"Strictness analysis for attribute grammars,"
in *PLILP'92*, vol. 631 of *Lecture Notes in Computer Science*, pp. 145-157, Springer-Verlag, 1992. [DART-115].

[Rosendahl3] M. Rosendahl,
"Higher-order chaotic iteration sequences,"
in *PLILP'93, Tallinn, Estonia*, no. 714 in Lecture Notes in Computer Science, pp. 332-345, Springer-Verlag, 1993. [DART-193].

[Sands1] D. Sands,
"A compositional semantics of combining forms for Gamma programs,"
in *International Conference on Formal Methods in Programming and Their Applications.*, Lecture Notes in Computer Science, 1993. [DART-187].

[Sands2] D. Sands,
"Laws of parallel synchronised termination,"
in *Theory and Formal Methods 1993: Proceedings of the First Imperial College, Department of Computing, Workshop on Theory and Formal Methods* (G. Burn, S. Gay, and M. Ryan, eds.), (Isle of Thorns, UK), Springer-Verlag Workshops in Computer Science,29 - 31 March 1993. [DART-188].

[Sands3] D. Sands,
"A naïve time analysis and its theory of cost equivalence."
Submitted for publication at DIKU, Copenhagen, 1993. Can be obtained by ftp from site `ftp.diku.dk` in `semantics/papers/D-173.ps,dvi`. [DART-186].

[Schwartzbach] M. I. Schwartzbach,
"Type inference with inequalities,"
in *Proc. TAPSOFT'91*, vol. 493 of *Lecture Notes in Computer Science*, Springer-Verlag, 1991. [DART-116].

[Sestoft] P. Sestoft,
Analysis and efficient implementation of functional programs.
PhD thesis, DIKU, University of Copenhagen, Denmark, 1991. DIKU Research Report 92/6. [DART-117].

[Solberg] K. L. Solberg,
"Inference systems for binding time analysis,"
Tech. Rep. 25, Odense University, 1993. [DART-161].

[SolbergNielsonNielson] K. L. Solberg, H. R. Nielson, and F. Nielson,
"Inference systems for binding time analysis (extended abstract),"
in *WSA '92: Work-shop on Static Analysis*, Bigre no. 81-82, pp. 247-254, University of Bordeaux, 1992. [DART-135].

[SondergaardSestoft] H. Søndergaard and P. Sestoft,
"Non-determinism in functional languages,"
Computer Journal, vol. 35, pp. 514-523, 1992. [DART-118].

[SorensenClausen] B. B. Sørensen and C. Clausen,
"Adequacy results for a lazy functional language with recursive and polymorphic types,"
Internal Report DAIMI IR-113, Computer Science Dept., Aarhus Univ., 1992. Submitted to Theoretical Computer Science. [DART-119].

[Sorensen] M. H. Sørensen,
"A new means of ensuring termination of deforestation."
Accepted for the Workshop of Global Compilation in connection with the Internation Logic Programming Symposium, 1993, Vancouver, Canada. Copies available from the author (at DIKU)., 1993. [DART-190].

[Tofte1] M. Tofte,
"Code generation using standard ML."
Lecture notes from DIKU, 1991. [DART-170].

[Tofte2] M. Tofte,
"Tutorial on standard ML,"
Tech. Rep. 91/18, DIKU, University of Copenhagen, Denmark, 1991. Presented at FPCA, Boston, 1991. [DART-171].

[Tofte3] M. Tofte,
"Principal signatures for higher-order program modules,"
in *The 19th Annual ACM Symposium on Principles of Programming Languages, Albuquerque, New Mexico*, pp. 189-199, 1992. [DART-120].

[TofteTalpin1] M. Tofte and J.-P. Talpin,
"A theory of stack allocation in polymophically typed languages,"
Tech. Rep. DIKU-report 93/15, DIKU, University of Copenhagen, Denmark, 1993. [DART-195].

[TofteTalpin2] M. Tofte and J.-P. Talpin,
"Implementation of the typed call-by-value lambda-calculus using a stack of regions,"
in *Proceedings from the 21st annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 1994. (Accepted for publication). [DART-196].

[Winskel1] G. Winskel,
"On local model checking the modal ν-calculus,"
1991. In the ICALP'89, special issue of Theoretical Computer Science, vol.83, 1991. [DART-121].

[Winskel2] G. Winskel, ed.,
CLICS Workshop-Parts I and II. Proceedings of the Workshop on Categorical Logic in Computer Science, 1992.
Also available a.s Tech. Rep. DAIMI PB-397 I and II, Computer Science Departement, Aarhus University. [DART-122].

[Winskel3] G. Winskel,
The formal semantics of programming languages.
MIT Press, 1993. [DART-123].

[WinskelCamilleri] G. Winskel and J. Camilleri,
"CCS with a priority choice,"
in *Proceedings of LICS*, 1991. [DART-124].

[WinskelLarsen] G. Winskel and K. Larsen,
"Using informations systems to solve recursive domain equations,"
Information and Computation, vol. 91, no. 2, 1991. [DART-125].*

[WinskelNielsen] G. Winskel and M. Nielsen,
"Models for concurrency."
To appear as a chapter in the Handbook of Logic and the Foundations of Computer Science, Oxford University Press. [DART-126].

[Xinxin] L. Xinxin,
Specification and Decomposition in Concurrency.
PhD thesis, Department of Mathematics and Computer Science, Aalborg University, Denmark., 1992. [DART-127].

[YiLarsen] W. Yi and K. G. Larsen,
"Testing probabilistic and nondeterministic processes,"
Proceedings of PSTV'92, 1992. [DART-128].