MINUTES
of the IFIP Working Group 2.2 Meeting
13-17 June 1995 in Amsterdam, The Netherlands
– Technical Sessions –

The meeting of the Working Group took place at the "Centrum voor Wiskunde en Informatica" (CWI) in Amsterdam.

In this meeting the members J.W. de Bakker (host) E. Best D. Bjørner, M. Broy (first day), P. Darondeau, R. De Nicola, S. Igarashi, F. Kröger, P.E. Lauer, P.D. Mosses, E.-J. Neuhold (chair), E.-R. Olderog (secretary), K.V.S. Prasad, W.P. de Roever, B. Steffen, A. Tarlecki, and the following observers participated: E. Börger, R. Cleaveland, J. Esparza, R. Gorrieri, B. v. Karger, S. Martini, K.V.S. Prasad, D. Sannella, S.A. Smolka, F. Vaandrager.

The technical sessions of the working group comprised of 19 presentations followed by extensive discussions. This time most talks addressed aspects of reactive systems, in particular temporal logic, real-time and hybrid systems, and semantics of concurrency. Other topics were system analysis, design and verification as well as semantics of specifications, problems in the λ-calculus, and tools for proof development. It follows a collection of authors' abstracts ordered alphabetically.

Dines Bjørner, Macau (`db@iist.unu.edu`):
On Domain Analysis, Requirements Capture and Software Architecture – Illustrated through an "Air Traffic" Example

Domain analysis (D/A), requirements capture (R/C) and software architetcure (S/A) are terms frequently used in connection with software conception. Through an example we illustrate their meaning and relations between them.

What is Air Traffic? We try to illustrate an answer to this question by presenting the basics of an Intrinsics description of Airports and Airways, of Time Tables, and of Actual, Scheduled and Re-scheduled Traffic. The answer illustrates use of "theories" of Geography (Air Space), 3D Bodies, Time, Time Tables, etc., and their projection onto a theory of Graphs. The former theories now relate via relations between the projections.

Note : All this is part speculative, part embryonic !

Our answer also allow us to "speculate" about other Domain Analysis components than Intrinsics: Support Technology, Rules & Regulations, Staff and Client Behaviour, Computing & Communications Platforms – all in preparation for Requirements Capture. We are then in a position to isolate "exact" meta-requirements as to what Requirements Capture "is all about"! ¿From sketches of a Requirements Specification we can then – incl. further injection of ideas – construct a suitable Software Architecture (i.e. give its Specification).

The talk illustrates UNU/IIST's R&D work with young Fellows from Developing Countries on *Software Support for Infrastuctures*, one of UNU/IIST's main three lines of Programmatic Activity.

Egon Börger, Pisa (boerger@di.unipi.it):
A Correctness Proof for Pipelining in RISC Architectures
(joint work with S. Mazzanti)

We provide a logical specification of Hennessy's and Patterson's RISC processor DLX and prove the correctness of its pipelined model with respect to the sequential ground model. In the first part we concentrate our attention to the provably correct refinement of the sequential ground model DLX on the pipelined parallel version DLXp in which structural hazards are eliminated. Then we extend the result to the full pipelined model DLXpipe in which also data and control hazards are treated.

Our specification proceeds by stepwise refinement, the proof method is modular. The specification and verification method is general and can be applied to other microprocessors and pipelining techniques as well.

Don Sannella, Edinburgh (dts@dcs.ed.ac.uk):
On Behavioural Abstraction and Behavioural Satisfaction in higher-order logic
(joint work with Martin Hofmann)

The behavioural semantics of specifications with higher-order logical formulae as axioms is analyzed. A characterization of behavioural abstraction via behavioural satisfaction of formulae in which the equality symbol is interpreted as indistinguishability, which is due to Reichel and was recently generalized to the case of first-order logic by Bidoit *et al*, is further generalized to this case. The fact that higher-order logic is powerful enough to express the indistinguishability relation is used to characterize behavioural satisfaction in terms of ordinary satisfaction, and to develop new methods for reasoning about specifications under behavioural semantics.

W.P. de Roever, Kiel (wpr@informatik.uni-kiel.d400.de):
Compositional Verification and Refinement in a Dense Time Temporal Logic (DTL)

The success of temporal logic is due, apart from to the tremendous energy and perseverance of its prime exponent in computer science, to the facts that (1) temporal logic handles fairness adequately and (2) its propositional version allows for an automatic decision procedure – model checking, introduced by Emerson and Clarke in 1980 – when applied to finite state systems.

However, the limits of its effectivity are reached when applied to concurrent automata of the kind and size required for describing pipelining and caching algorithms as recently required for new VLSI designs. Then more sophisticated techniques are called for, in which first refinement of complex systems is reduced to refinement between their components, this is called *compositional refinement*, and then standard model checking is applied. Of course, this presupposes a temporal framework in which refinement between concurrent systems is accomodated for, i.e. in which Lamport's stutter problem has been solved.

Both these requirements are meaning a simple dense time temporal logic, called DTL, designed by Tonio Cau (now at Liverpool) and based on a generalization of Peter Aczel's compositional model for shared variable concurrency and joint work with Pierre Collette (now at Manchester). Cau designed this logic in able to solve complicated large size verification problems in concurrency involving fault-tolerance. Notably correctness of stable storage and of a transformation of general semaphores to binary split semaphores (a technique introduced by Dijkstra in the '80s). DTL's underlying semantic model coincides with that recently introduced by Krogh and others in control engineering for specifying and designing hybrid systems.

Burkhard von Karger, Kiel (bvk@informatik.uni-kiel.d400.de):
Temporal Algebras: Calculating with "next" and "previous"

We present Linear Temporal Logic as a complete Boolean algebra with two unary operators. Augmenting the boolean structure by three simple postulates enables us to derive all seventeen axioms of the sound and complete proof system given in Manna and Pnueli's book. The lesson to be learnt is summarized in the following slogan "Temporal logic is the theory of Galois connections".

Rance Cleaveland, North Carolina (rance@csc.ncsu.edu):
Divergence and Fair Testing

This talk presents a new testing-based semantic theory of processes that aims to circumvent difficulties that traditional testing / failures theories have in dealing with divergent behavior. The framework incorporates a notion of fairness into the determination of when a process passes a test; we contrast this definition with existing approaches and give characterizations of the induced semantic preorders. An extended example highlights the utility of the results.

Frits Vaandrager, Amsterdam (fritsv@cwi.nl):
Linear Hybrid Systems

Hybrid systems are systems that intermix discrete and continuous components, typically computers that interact with the physical world. In this presentation, I will discuss a simple, very restricted model of hybrid systems, namely the model of linear hybrid automata of Alur et al [1]. This model has become quite popular recently because it allows for fully automatic model checking of temporal logic formulas. The semantics of linear hybrid automata will be defined via a translation to the timed I/O automata of Lynch and Vaandrager [3]. To illustrate the model of linear hybrid automata, I will briefly present an example of [2]: a simple version of a protocol developed by Philips for the physical layer of an interface bus that connects the various devices of some stereo equipment.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J.Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[2] D.J.B. Bosscher, I. Polak, and F.W. Vaandrager. Verification of an audio control protocol. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *Proceedings of the Third International School and Symposium on Formal Techniques in Real Time and Fault Tolerant Systems,* Lübeck, Germany, September 1994, LNCS 863, pages 170–192. Springer-Verlag, 1994. Full version available as Report CS-R9445, CWI, Amsterdam, July 1994.

[3] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations for timing-based systems. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings REX Workshop on Real-Time: Theory in Practice,* Mook, The Netherlands, June 1991, LNCS 600, pages 397–446. Springer-Verlag, 1992.

Philippe Darondeau, Rennes (Philippe.Darondeau@irisa.fr):
Regions in Transition Systems

The so-called synthesis problem consists in deciding for a given graph G whether there exists a net N with case graph isomorphic to G. This problem was solved for elementary nets by Ehrenfeucht and Rozenberg. We show that their solution extends to arbitrary types of nets, where a type of nets is a deterministic transition system $\mathcal{T} = (X, Y, \tau)$ with set of transitions $\tau \subset X \times Y \times X$.

A net of type $\mathcal{T}$ is a triple $N = (P, E, F)$, where $F : P \times E \longrightarrow Y$. A marking of N is a map $M : P \stackrel{F(p,e)}{\longrightarrow} X$. The firing rule is as follows: $M[e > M'$ iff $\forall p \in P \bullet M(p) \longrightarrow M'(p)$ is in τ. Given $G = (S, E, T)$, regions in G may be identified with morphisms of transition systems $(\sigma, \eta) : G \longrightarrow \mathcal{T}$, when $\sigma : S \longrightarrow X$ and $\eta : E \longrightarrow Y$. Using this identification, we show that the synthesis problem is polynomial for P/T nets, NP-complete for elementary nets, and polynomial for flip-flop nets, a new type of C/E nets where it is possible for a transition to switch a place from 0 to 1 or from 1 to 0 depending on its actual content.

In a second part, we construct a general duality between automata and nets for arbitrary types of nets. For that purpose, we consider for each type of nets $\mathcal{T} = (X, Y, \tau)$ a schizophrenic object $(\mathcal{T}, \tau, \mathcal{N})$ where $\mathcal{N}$ is a net with one transition and set of places $\mathcal{T}$ equal to the set of transitions of $\mathcal{T}$. The dual G^* of a transition graph G is then a net (of type $\mathcal{T}$) with set of places equal to Trans $(G, \mathcal{T})$, i.e. equal to the set of regions in G. Symmetrically, the dual N^* of a net N (of type $\mathcal{T}$) is a transition system with set of places equal to Nets $(N, \mathcal{N})$, and this transition system is nothing else than the marking graph of N.

Javier Esparza, München (`esparza@informatik.tu-muenchen.de`):
Decidability of Model Checking for Infinite-State Concurrent Systems

The 80's witnessed a lively debate on the advantages and disadvantages of linear and branching-time logics for the verification of finite-state systems. In particular, there was intensive resarch on the complexity of model checking procedures. Broadly speaking, there are faster model checkers for branching-time logics, but there are also interesting properties which can be succintly expressed in some linear-time logic, and are difficult to express in branching-time logic.

In the 90's more and more research is being done on the verification of *infinite*-state systems. It is therefore time to ask if the conclusions of the finite-state debate still hold for the infinite case.

We have studied the decidability of the model checking problem for two infinite-state models of concurrent systems: Petri nets, which are not Turing powerful but still very expressive, and Basic Parallel Processes, which is probably the infinite-state concurrent model with the smallest modelling power. We prove the following two results:

- the model checking problem for Petri nets and all the usual linear-time logics (all logics contained in the linear-time mu-calculus) is decidable;
- the model checking problem for Basic Parallel Processes and almost all branching-time logics (loosely speaking, all logics able to express "in every run eventually ϕ holds") is undecidable.

Therefore, the conclusion is: in the infinite state-case, and for concurrent systems, linear-time logics are "more decidable" than branching-time ones.

However, we also show that the undecidability result does not hold for partial order semantics. In a natural partial order interpretation, the model checking problem for Basic Parallel Processes and CTL* turns out to be decidable.

References

[1] J. Esparza: On the decidability of model checking for several μ-calculi and Petri nets. Proceedings of CAAP '94 (1994), LNCS 787, 115–129 (1994).

[2] J. Esparza und A. Kiehn: On the decidability of model checking for branching time logics and Basic Parallel Processes. To appear in CAV'95 (1995).

Simone Martini, Udine (`martini@dimi.uniud.it`):
Optimal λ-reduction and the Proof-Theory of Modal Logic

The implementation of Lévy's optimal reduction strategy for lambda-terms (due to Lamping, Gonthier et al., Asperti, etc.) leaves open several problems, among which the elimination of spurious control nodes in normal forms, and the recovering of the actual term in normal form from the normal form graph ("read-back").

I present a modified approach to the problem, deriving some of its crucial ideas from previous work (joint with A. Masini) on the proof-theory of modal (and linear) logic. In this approach, formulas are decorated with levels, by which we obtain local and symmetric rules.

When this approach is tailored to linear logic and it is seen in a proof-net framework, the levels appear as a very natural information (they correspond to the number of boxes a formula is contained into). From this insight – and the well known embedding of λ-calculus into linear logic – we may improve the graph reduction implementation of optimal lambda-reduction (joint work with S. Guerrini), obtaining a system where:
(i) A graph in normal form with respect to beta-reduction has no spurious control nodes.
(ii) A graph in normal form with respect to all rules is the representation of the normal form of the original lambda-term. The rewriting system is confluent and normalising.
(iii) The erasing of subterms can be easily treated (a subject which has been neglected in all the relevant literature).

Jaco de Bakker, Amsterdam (`jaco@cwi.nl`):
Semantics for 27 Languages

We present an outline of our book (with E. de Vink as co-author) Control Flow Semantics, to be publised by MIT Press, end of 1995. The book develops operational and denotational semantics for 27 languages and gives precise relationships between these in all cases considered. Both imperative (sequential/concurrent) and other programming paradigms (LP, OO, FP) are represented. The methodology aims at modelling full histories of the computations, using metric structures as a convenient tool for this purpose. A language with second-order assigment is used as a case study to illustrate the method; it also serves as a step on the way to the modelling of second-order communication.

Scott Smolka, Stony Brook (`sas@cs.sunysb.edu`):
Real-Time Processes for the Concurrency Factory: A Proposal
(joint work with O. Sokolsky)

The Concurrency Factory is a graphical environment for the specification, simulation, verification, and implementation of concurrent systems. Specifications in the Factory take the form of hierarchical networks of finite-state processes. Here we propose to extend such specifications to real-time by annotating transitions with pairs of the form $([a,b],c)$, where $[a,b]$ is the range in which the transition becomes nondeterministically enabled, and c is the time at which the transition becomes disabled.

We provide several examples to illustrate the expressive power of the proposed notation. We also discuss some of the issues involved in translating these specifications into Alur-and-Dill-style timed automata, particularly with regard to the maximal progress assumption typically found in real-time process algebras. Finally, we provide an overview of a local model checking algorithm we have recently developed for timed automata and a real-time extension of the modal mu-calculus.

Roberto Gorrieri, Bologna (`gorrieri@cs.unibo.it`):
Towards Performance Evaluation in Process Algebra

Some ideas on the integration of performance evaluation within the formal specification of the behaviour of concurrent systems have been proposed. The talk is centered around a simple process algebra with the characterizing feature that actions are time-consuming. It is first studied the case when the duration of action is deterministic and statically fixed. The operational semantics produces a transition system which is labelled by pairs of the form $< a, n >$, meaning that action a has been completed at time n. The proposed behavioural equivalence, called *performance equivalence*, is based on standard bisimulation on such a labelled transition system. It turns out that performance equivalence is a non-interleaving semantics which equates systems whenever they perform the same actions in the same amount of time.

Then, the theory has been extended with a preorder based on execution speed, called *performance preorder*. Two processes P and Q are related if they are strongly bisimilar (i.e., functionally equivalent) and the first one is at least as fast as the second one. Hence, this preorder supports the stepwise refinement "from specification to implementation" by increasing efficiency while retaining the same functionality. It was hinted that the problem of finding faster implementations for a specification is connected to the problem of finding more distributed implementations of the same specification. This is an immediate consequence of the proof that the location preorder, which is based on a measure of distribution, implies the performance preorder. In the last part of the talk, it was investigated a natural generalization of the theory above to the case when the duration of action is represented by stochastic variables. Some introductory features of one of these languages, called Markovian Process Algebra, have been presented. A modeling technique for concurrent systems based on MPA is also proposed which integrates different views of concurrent systems (centralized vs distributed) as well as different aspects of their behaviour (qualitative-functional vs quantitative-temporal). Finally, one example was shown which demonstrates the expressiveness and the compositionality of MPA.

Berhard Steffen, Passau (`steffen@fmi.uni-passau.de`):
Parallelism for Free
(joint work with J. Knoop and J. Vollmer)

One of the major problems in the verification of distributed systems is the combinatorial explosion of the size of an explicit state-oriented representation. Current research indicates that there is no general way around this problem, but there is a huge variety of sometimes unexpected special problem classes, where the 'state explosion' can be avoided.

One such class is given by the bitvector problems for parallel programs with shared memory. We can show that they can be solved as efficiently as their purely sequential counterparts. Central idea behind our algorithm is the separation of the determination of the interference information from the subsequent use of this information for solving the considered bitvector problem. Both steps can be performed efficiently on the parallel program. Thus using our method, the standard algorithms for sequential programs computing liveness, availability, very business, reaching definitions, definition-use chains, or performing partially redundant expression and assignment elimination, partial dead code elimination or strength reduction, can straightforwardly be transferred to the parallel setting at almost no cost.

Peter Lauer, McMaster, Hamilton (`lauer@maccs.dcss.mcmaster.ca`):
Tools for Tool-Builders

The present evaluation of the Synthesizer Generator as a vehicle for experimentation with computer based systems of logic, forms part of a broader evaluation of the most advanced implementations of logics, logical frameworks and theory building tools.

The Synthesizer Generator seems particularly suited for the rapid development of alternative presentations of logics and user interfaces. In what follows, this statement is refined in the context of the development of a series of proof-development-assistants for a classical first order logic. We use the term proof-development-assistant to differentiate it from what is usually called a proof-checker, since the Synthesizer Generator generates a series of user interaction platforms, namely the editor for the specific logic in question, the enclosing general emacs-like text editor always generated for any logic, and the outermost enclosing X Window interface. This nested set of platforms provides a much richer set of possible user interactions during the development of logics and while experimenting with them than do special purpose logical frameworks. We shall illustrate precisely what we mean with concrete examples from our experimentation. Particularly, we will point out seminal directions for research wherever they occur to us.

Ernst-Rüdiger Olderog, Oldenburg (olderog@informatik.uni-oldenburg.de):
Design of Real-Time Systems
(joint work with He Jifeng and A.P. Ravn)

We present an approach to the development of correct programs for the control of real-time systems. System requirements are specified in Duration Calculus (DC for short), a real-time interval temporal logic developed by Zhou Chaochen. In this logic the desired relationships between the time dependent observables of the system can be expressed in a convenient manner.

The initial DC specifications are then transformed stepwise into more specific formats, so-called DC Implementables and finally into Sequential DC. That subset of DC can be translated one-to-one into timed regular expressions. Symbols in these expressions correspond to communications with sensors and actuators for the system environment.

The approach is developed within the ESPRIT BRA ProCoS II (Provably Correct Systems) and illustrated with the Generalized Railroed Crossing example due to C. Heitmeyer and N. Lynch.

Eike Best, Hildesheim (E.Best@informatik.uni-hildesheim.de):
Soving Recursive Equations on Nets

This talk describes a denotational approach to the Petri net semantics of recursive expressions. A domain of nets is identified such that the solution of a given recursive equation can be found by fixpoint approximation from some suitable starting point. In turn, a suitable starting point can be found by fixpoint approximation on a derived domain of sets of trees. It is shown that there always exists a countable solution, which is unique in case recursion is guarded.

K.V.S. Prasad, Gothenburg (prasad@cs.chalmers.se):
Computer Aided Reasoning about Broadcasting Systems

This is a report on work in progress. The goal is to produce correct, executable, concurrent programs. The programs are in CBS (a Calculus of Broadcasting Systems) embedded in ALF, a proof checker based on type theory. Thus they are typed and allow the use of all usual data types for transmission. Executable Haskell programs can be extracted from ALF proofs. Since correctness is established by proof rather than by model checking, value passing is easily taken care of by quantifying over elements of the data type being transmitted.

The method is to first represent CBS processes as a new data type, parameterised by the data type of transmited values. Next, the operational semantics (SOS rules) is represented as an inductive set. Bisimulation can now be defined as a predicate on relations between processes. (Though not bisimulation equivalence, which is impredicative). Also Hennessy-Milner logic and its satisfaction relation can be represented. Proofs about processes can now be carried out. To complete the link to execution, we use the CBS interpreter, which is itself represented in ALF and proved sound and complete wrt the operational semantics.

Much of this work was carried out over the last two years. A major result proved is that late and early bisimulations coincide for CBS. Also several small example bisimulations have been carried out. As the ALF systems is now being completely redesigned, the CBS project has to be redone in the new version. Simultaneously, recursion is being represented as an explicit process constructor, rather than being borrowed from ALF. This avoids dependence on the theory of infinite objects in ALF, which is still under development.

Andrzej Tarlecki, Warsaw (tarlecki@mimuw.edu.pl):
Moving Between Logical Systems

In the process of systematic development of large computer systems from their formal specifications, it is often most natural to use a number of different logical systems to specify various aspects of system behaviour, to describe various modules of the resulting composite heterogenous system, and to capture features typical for different stages of system development. For example, we may want to join the full expressive power of first-order logic with "constructiveness" of equational logic, to combine the use of software specification techniques and of hardware description languages for various system components, to switch from high-level axioms describing the behaviour of system modules to a logic of a specific programming language when nearing the completion of the development, etc. All this requires the possibility of using multiple logical systems in a single specification/development task.

An initial step towards this is to formalise the concept of a logical system, and then to develop a formal framework for working within an arbitrary but fixed logical system. Much work in this direction has been done based on the theory of *institutions* (the concept introduced by Goguen and Burstall in the early 80s). Then, a necessary prerequisite for specifications/developments spanning a number of logical systems is some notion of a *morphism between institutions* to allow specifications and developments to migrate from one institution to another.

In my talk I will present a number of possible definitions of such a morphism together with the intuitions they attempt to capture. I will discuss their mutual merits and demerits, and try to sketch their role in specification/development process. Each such notion of a morphism between institutions gives rise to a category, often enjoying nice structural properties. I will discuss to what extent categorical constructions may be viewed as a tool for building more complex logical systems from simpler ones. It turns out that working with *logic presentations* (or *parchments*, in the case of institutions) seems to be more appropriate here.

Although this will be mostly presented using the model-theoretic view of logic, as captured by the notion of an institution, I believe that the ideas apply just as well to type-theoretic logical frameworks,based on the proof-theoretic view of logic. Some of the most intricate problems arise when we try to unify the two views and discuss relationships between logical systems covering both model-theoretic and proof-theoretic aspects.

***