http://homepages.cwi.nl/~bertl/concurrency/archive/concurrency-1988-1990

From pratt@cs.stanford.edu Wed May 29 16:39:48 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Mon, 22 Oct 90 09:57:06 EDT
To: concurrency
Subject: modelling concurrency with partial orders
From: pratt@cs.stanford.edu
Sender: meyer@theory.lcs.mit.edu

Date: 21 Oct 90 15:41:06 PDT (Sun)

Here's some correspondence that should be of interest to this list.
I was in a bit of a rush so my answer is a lot less polished than
I'd like, but it'll have to do.

Vaughan Pratt

------- Forwarded Message

Date: Sun, 21 Oct 90 13:39:23 -0700
From: dcl@anna.Stanford.EDU
Sender: dcl@anna.Stanford.EDU
To: pratt@cs.Stanford.EDU
Subject: Partially Ordered Computations

Vaughn,
In some recent discussions with people funded by ONR's
program on distributed and realtime computing,
I have found an attitude that

"sets of linear traces
are entirely sufficient for analyzing distributed/concurrent
computations, AND Partial Ordrs are unnecessary".

I also notice that sets of linear traces are the basis for
Hoare's PROCOS project.

Questions to you:
1. What is your favourite simple example of a system where
   a partial order representation of its execution is superior
   to a set of linear traces of its execution,

2. Would you disagree with the ONR people, and how?

- David [Luckham]

------- End of Forwarded Message

------- Forwarded Message

To: dcl@anna.stanford.edu
Cc: pratt@cs.stanford.edu
Subject: Re: Partially Ordered Computations
In-Reply-To: Your message of Sun, 21 Oct 90 13:39:23 -0700.
             <9010212039.AA07939@Aphid.Stanford.EDU>
Date: 21 Oct 90 15:14:37 PDT (Sun)

From: pratt@cs.Stanford.EDU

The belief that linear orders capture partial is predicated on several assumptions, most of which have to hold at the same time in order for it to be reliable. While these assumptions tend to hold in very simple or abstract systems, they all gradually fade away as the systems you look at get larger and more concrete.

Here are seven such assumptions.

1. Fixed granularity.

2. No variability of atomic events.

3. Absence of autocurrence.

4. Single-poset processes.

5. Race-free.

6. Single-observer model.

7. Discrete time.

Here is the meaning of each of these concepts.

1. Variable granularity can arise in various quite different ways. One way is just to look at a supposedly atomic event more closely and resolve substructure. But another is to take a binary program whose *specification* treats it as atomic (on the ground that the vendor doesn't want you to assume anything about the package) and find when you run it that it has a series of side effects on your system, that may interleave with the side effects of other such packages.

You might find it interesting to look at "Teams Can See Pomsets" by Plotkin and myself to see what influence variable granularity can have. It turns out this is not the theoretically worst problem in our paper, #2 below is worse, but it does have some influence.

You can anonymous-ftp a preliminary version of this paper this from boole.stanford.edu on pub/pp2.*.

2. Variability of atomic events means that although an event stays atomic it might not do identical things each time it happens. Plotkin and I use this phenomenon to show that a sufficiently large team of observers (see item 6) can distinguish *any* two finite pomsets.

3. Autocurrence means two concurrent and identical events. Without the concurrency requirement we find two such repetitions in the word "identity": there are two t's and two i's. An example with concurrence is when you ask the bank teller for two dollars. If dollars always came sequentially there'd be no quarrel about the legitimacy of the string 11 as a specification for two dollars. But what about 1|1 meaning "Give me two dollars please." This phenomenon arises as soon as you distinguish pomsets from posets.

With autocurrence you can get a|a, which traces can't distinguish from aa. This can be solved via so-called "action refinement", used in solving 1 above. But action refinement gets you only so far, in particular it can't be used in conjunction with traces to distinguish TR|TR (two parallel sequences each of T->R, e.g. two parallel message streams) from the same thing with the extra requirement that one of the T's precede *both* of the R's. But pomsets can make that distinction, using the N pomset.

4. A single-poset process is one defined by a single poset. This is a key assumption in the theorem coding posets as their linearizations. However this assumption is rarely achievable in practice. It is false that a set of posets can be encoded with the union of their respective sets of linearizations.

5. When a and b are in a race, the trace model reveals only ab+ba. But race-free nondeterminism, which chooses one of ab+ba, has the same trace representation. This matters for example in the glitch problem. You may want to implement ab+ba glitch-freely, but you cannot say it with traces. This is a pretty simple argument, so you might use it first (I suppose I should have).

The same argument applies to distinguishing the mutually exclusive execution of two atomic operations from their concurrent execution. The trace model has built into it the assumption that mutually exclusive execution and concurrent execution are the same thing for atomic events. This interacts with item 1.

6. Most models of concurrency assume that one observer collects all the observations. In practice observers are as distributed as the systems they observe, and can pool their distributed observations in ways entirely unrelated to the computational model used to prove correctness of a particular distributed system. This is a subtle point that Plotkin and I go to pains to explain in detail in our paper.

7. Time must be discrete for traces to model interleaving. Just what exactly is the set of all interleavings of two copies of the unit interval [0,1]? Consider a dual beam oscilloscope. Are you going to describe its two beams in terms of their interleavings?

These issues are specific technical problems that arise with traces. But besides any question of what might actually go wrong, there is also the question of the most *natural* model. I feel that models should attempt to be reasonably faithful to what they model, *if* the mathematics supports this. Even if your unnatural model happens to be working today, my feeling is that unnatural models are more likely to break down in the future than natural ones..

When you have a computer in Europe talking via satellite to one in the US, the time between instructions is thousands of times less than that between computers. A natural way to model the instruction streams of the two computers then is with two sequences. The trace model does not accept this, on the ground that a computation consists of one sequence. It says that you must interleave the two sequences in all possible ways before you can reason soundly about the system.

The problem is that the only serious mathematics that many practicing computer scientists get exposed to is computation theory, where they are taught that all computation is sequential. Getting through their computation theory course was one of the bigger struggles of their college education, but mastery of it vindicated the enormous outlay of tuition and board for all those years when they could have been learning on the job.

So then they run into concurrency in the real world and they simply cannot cope with the concept of two parallel streams, because they have never seen any such concept in their textbooks, nor any theorems about such concepts. Therefore they do the only thing possible: they interleave in order to reduce to a known model with known theorems.

I can say on the basis of having worked with both models for many years that posets are far more flexible and easier to work with than traces. Having to think about systems in terms of traces is like trying to do arithmetic with Roman numerals. Yes, Roman numerals indeed code integers, and furthermore the algorithms for adding and multiplying Roman numerals do work, but that's not a great reason to stick with Roman numerals.
- -v

------- End of Forwarded Message

From rance@adm.csc.ncsu.edu Wed May 29 16:39:49 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Mon, 22 Oct 90 11:41:30 EDT
To: concurrency
Subject: Re: modelling concurrency with partial orders
From: rance@adm.csc.ncsu.edu (Rance Cleaveland)
Sender: meyer@theory.lcs.mit.edu

Date: Mon, 22 Oct 90 11:29:48 -0400

Another reason for using posets crops up when one wishes to reason about the real-time properties of a system. Assuming that one is working in a setting where each atomic action takes 1 time unit, a|b ("a and b truly in parallel") should also take 1 time unit, while ab + ba will take 2. So it seems a bit surprising to me that a group of people interested in real time would find linearizations an adequate model of concurrency.

Rance Cleaveland

From pratt@cs.stanford.edu Wed May 29 16:39:50 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Tue, 23 Oct 90 11:27:12 EDT
To: concurrency
Subject: modelling concurrency with partial orders
From: Vaughan Pratt <pratt@cs.stanford.edu>
Sender: meyer@theory.lcs.mit.edu

Date: Mon, 22 Oct 90 12:57:26 PDT

Rance's comment on real time reminds me. I neglected to connect up with recent work explaining why true-concurrency hackers seem to prefer the poset side of an otherwise surely symmetric duality between posets as schedules and distributive lattices as automata, a duality generalized by Winskel et recently many al to event structures, dual to families of configurations.

The reason is that automata are 1-dimensional and hence can only exhibit the structure of interleaving concurrency. This is intuitively obvious to true true concurrency hackers, and I can only infer that the proponents of this duality in its published form are false true concurrency hackers.

In order to faithfully and continuously represent, on the automaton side of the duality, the structure of true concurrency that its proponents like myself so vividly imagine to exist on the poset side, automata should be made higher dimensional. This has been done implicitly by van Glabbeek and Vaandrager in PARLE-87 via the notion of ST-bisimulation. I will be momentarily sending off my POPL paper explaining how to make more explicit the geometry implicit in this (if I just can restrain myself long enough from writing these damn messages).

A propos of real time, the phenomenon by which two pencils can be put into a shirt pocket only high enough to accommodate one, impossible in the interleaving world as Rance points out, translates under this duality to the need for the L-infinity norm (i.e. max(x,y)) in measuring duration of truly concurrent processes in higher-dimensional automata. In contrast the L-1 norm or Manhattan metric x+y measures duration of interleaved processes, that operate the way a New York taxi has to in alternating between going East and North. (So you should have inferred by now that this is the model where one lays out parallel instruction streams orthogonally, as Papadimitriou does in treating deadlock).

If one tries to approach true concurrency by refining the granularity of this interleaving, one arrives in the limit at still the L-1 norm. That is, you may have a perfectly straight line running diagonally across the product square (the product of two transitions, a surface, arising just as in the product construction for automata) but it still represents interleaved concurrency by being its limit. In this extreme case true concurrency can be distinguished from interleaving not by its shape but only its speed.

Who was Procos? Zeno's turtle?
-v

From pratt@cs.stanford.edu Wed May 29 16:39:51 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Thu, 25 Oct 90 14:42:45 EDT
To: concurrency
Subject: Ad: Modeling Concurrency with Geometry
From: Vaughan Pratt <pratt@cs.stanford.edu>
Sender: meyer@theory.lcs.mit.edu

Date: Wed, 24 Oct 90 21:30:15 PDT

I've just finished "Modeling Concurrency with Geometry" for
POPL, see abstract below. It can be retrieved via

```
ftp -i boole.stanford.edu
Login: anonymous
Password: surname
cd pub
bin
mget cg.* pratt.bib
quit
```

Modeling Concurrency with Geometry
V.R. Pratt

Branching time and causality find their respective homes in the Birkhoff-dual models of automata and schedules. This creates a puzzle: if the duality is supposed to make the models completely equivalent then why does each phenomenon have a preferred side? We identify dimension as the culprit: 1-dimensional automata are skeletons permitting only interleaving concurrency, true n-fold concurrency resides in transitions of dimension n. The Birkhoff dual of a poset then becomes a simply-connected space. We distinguish Birkhoff duality from Stone duality and treat the former in detail from a concurrency perspective. We introduce true nondeterminism and define it as monoidal homotopy; from this perspective nondeterminism in ordinary automata arises from forking and joining creating nontrivial homotopy. We propose a formal definition of higher dimensional automaton as an n-complex or n-category, whose two essential axioms are associativity of concatenation within dimension and an interchange principle between dimensions.

From infhil!eike@relay.eu.net Wed May 29 16:39:51 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Fri, 26 Oct 90 12:21:47 EDT
To: concurrency
Subject: Using partial orders to define time in asynchronous systems
From: infhil!eike@relay.eu.net (Eike Best)
Sender: meyer@theory.lcs.mit.edu

Date: Thu, 25 Oct 90 16:19:22 +0100

2 self--references on that subject:
--- Weighted Basic Petri Nets, Springer Lecture Notes Vol.335, 257--276
--- DEMON Project Motivation, Springer LNCS Vol.424, 487--506.

Eike Best

From infhil!eike@relay.eu.net Wed May 29 16:39:52 1991
Return-Path: <meyer@theory.lcs.mit.edu>

Date: Fri, 26 Oct 90 12:22:58 EDT
To: concurrency
Subject: Re: The discussion on (sometime) superiority of p.orders
From: infhil!eike@relay.eu.net (Eike Best)
Sender: meyer@theory.lcs.mit.edu

Date: Thu, 25 Oct 90 16:08:58 +0100

Here are my 2 Pfennige worth of contribution.
I claim:

Sometimes partial orders let you define a concept more smoothly
than arbitrary interleavings. A case in point is "finite delay".
Finite delay is supposed to mean: if an action is continually enabled,
then it occurs sometime.

In a sequential system, finite delay can be expressed by the
maximality of an execution sequence (you would like to go as far
as possible).

Consider a*||b* versus (a[]b)* (where [] is nondet. choice).
The sequence aaaaa... (infinitely often a but no b) contradicts the
finite delay property in a*||b*, since the b is not prohibited
>from occurring and could always occur. However, aaaaa... does
NOT contradict the finite delay property in (a[]b)*, since the
occurrence of a is always alternative to b,
and so b is continually prohibited from occurring.

The distinction can be captured by noticing that aaaaa...,
while being maximal as a string, is not maximal as a
partial order of a*||b*, but IS maximal as a partial order of (a[]b)*.

Eike Best

PS I don't claim you NEED partial orders here, but I do claim that
it's nice to use them, since the concept of maximality directly
generalises
the sequential one.

From meyer@theory.lcs.mit.edu Wed May 29 16:39:52 1991
Return-Path: <meyer@theory.lcs.mit.edu>
From: meyer@theory.lcs.mit.edu (Albert R. Meyer)
Date: Fri, 26 Oct 90 14:01:09 EDT
To: pratt@cs.stanford.edu
Cc: concurrency, dcl@anna.stanford.edu
In-Reply-To: pratt@cs.stanford.edu's message of Mon, 22 Oct 90 09:57:06 EDT
<9010221357.AA10447@stork>
Subject: modelling concurrency with partial orders

I support most of your remarks, but I don't think we should accept
David Luckham's formulation of the issue as
(1) Linear versus Partial Order
but rather emphasize
(2) Interleaving Nondeterminacy versus Concurrency

Formulation (1) highlights the particular detail of whether concurrent processes are abstractly represented by some structure involving linear, rather than partial, orders. This can hardly be crucial, since, as you well know, every partial order is uniquely determined by the set of its linearizations.

Formulation (2) forces us to clarify the limitations of the in many respects successful interleaving-concurrency models of CCS, CSP, MEIJE, ACP, etc. Though the following remarks are well known to you and the Continental research community in concurrency, Luckham's note confirms my impression that the issue is still not well understood elsewhere, so maybe it's worth rehashing the basis of the story another time:

The crux of the criticism of interleaving is captured in the equation
(3) $a|b = ab+ba$.
Equation (3) may be read as asserting that the process a|b, which can CONCURRENTLY perform actions a and b, may be identified with the process ab+ba, which NONDETERMINISTICALLY chooses to do either a-then-b or else b-then-a.

Equation (3) is an axiom in the interleaving-based theories, but maintaining it RULES OUT extensions of the theory to include

(i) observations of simultaneity: a and b can be observed simultaneously in the computation of process a|b, but not in ab+ba.

(ii) observations of the same computation by two or more sequential observers at distributed locations: under reasonable assumptions about signal propagation over distance, two such observers watching a computation of a|b might see DIFFERENT linear traces (namely one could see `ab' during the same interval that the other saw `ba'), but under the same assumptions two observers would always see the SAME trace (namely, exactly one of ab or ba) in any given computation of ab+ba. I was delighted by this remark when I first learned it from you and Plotkin.

(iii) refinement of action atomicity--what you felicitously called ``variable granularity'': refining a in a|b to be the two step sequential process cd yields a process with the trace cbe, but refining a in ab+ba yields no such trace; I first learned this point from a note in 1987 by Castellano et al in the EATCS Bulletin.

Insofar as these extensions are desirable, one has to retreat from the simple interleaving model. The ideas that actions have duration, and more generally the ideas of critical regions and atomicity, are usually regarded as an important aspect of pragmatic concurrent processing. Because (iii) seems like a plausible theoretical way to model both action duration and relaxing atomicity requirements, extending the theory to cover it does seem desirable.

On the other hand, having agreed that interleaving theories need modification, I don't think we can say that your pomset models or the Mazurkiewicz-trace models have been fully justified as appropriate concurrency theories. For example, multiple observers don't justify distinguishing the pomset processes P1 and P2 where P1 is the singleton pomset (.a .b) and P2 = P1 union one of its augmentations, say the singleton

|
.b

Similarly, the various proposed event/behavior structure models are all based on generalized notions of bisimulation. I have raised my doubts in earlier messsages to this forum about how the detailed distinctions between processes made by bisimulation can be justified computationally.

Despite these reservations, let me say that I do believe that the modeling of a concurrent run of a computation with a pomset is pretty natural.

Regards, A.
Moderator, concurrency@theory.lcs.mit.edu

From pratt@cs.stanford.edu Wed May 29 16:39:53 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Sat, 27 Oct 90 16:55:21 EDT
To: concurrency
Subject: Re: modelling concurrency with partial orders
From: pratt@cs.stanford.edu
Sender: meyer@theory.lcs.mit.edu

In-Reply-To: Your message of Fri, 26 Oct 90 14:01:09 EDT.
<9010261801.AA13008@stork>
Date: 26 Oct 90 14:52:07 PDT (Fri)

I appreciate your words of support, Albert. Some minor comments on four points.

>This can hardly be crucial, since, as you
>well know, every partial order is uniquely determined by the set of its >linearizations.

This is Szpilrajn's theorem [1], a ``fragile'' theorem in the following sense. A robust theorem about a structure should remain true when one adds further structure. Szpilrajn's theorem holds neither for a set of posets nor for labeled posets. Both these structures must be added to the basic poset structure to make it useful as a model of concurrency. I therefore view David's comparison of linear to partial orders in the context of their application to concurrency as quite appropriate.

>(3) a|b = ab+ba.
>Equation (3) is an axiom in the interleaving-based theories, but
>maintaining it RULES OUT extensions of the theory to include

The equational logic of regular expressions has a very interesting property. If you regard its variables as denoting only themselves as symbols of an alphabet, the set of equations valid under that very restricted interpretation turns out to be the same as when you let the variables range over arbitrary languages. That is, the theory does not change when you treat its variables as self-denoting constants.

This interesting property fails as soon as you add almost any other operation, whether or not that operation preserves regularity. Such operations include complement -a, intersection a&b, interleaving a|b, quotient a\b, and residual a->b = -(a\-b).

Equational theories are closed under substitution. In view of this I would like to discourage extending to other languages the practice in the language of regular expressions of denoting atoms by variables. I would be more comfortable seeing (3) written as a conditional implication:

```
atomic(a) & atomic(b) ->  a|b = ab+ba.
```

or more generally:

```
atomic(a) & atomic(b) -> mutex(a,b)
mutex(a,b)  ->  a|b = ab+ba
```

since mutex(a,b) (I hope the meaning is clear) is at its most useful when it holds of particular nonatomic processes.

>For example, multiple observers don't justify
>distinguishing the pomset processes P1 and P2 where P1 is the singleton
>pomset (.a .b) and P2 = P1 union one of its augmentations, say the
>singleton

Provably so of course: our multiple observer model can't distinguish a process from its augment closure. Gordon and I now have the converse of this, at least for finite pomsets, that is that distinct augment closed processes of finite pomsets are distinguishable by sufficiently large teams (infinite when the dimension of the pomsets is unbounded).

>I have raised my doubts in earlier messsages to this forum about how
>the detailed distinctions between processes made by bisimulation can be
>justified computationally.

Having written about it you're better qualified than I to express such reservations. However my intuitive feeling is that Hennessy-Milner logic, which justifies all distinctions made by bisimulation, is not an excessively strong language in the context of debugging, where the programmer marches backwards and forwards along a misbehaved nondeterministic computation trying to find what caused the misbehavior and experimenting by making little changes and seeing how they propagate side-effects forward and predicates backwards (through predicate transformers).

[1] E. Szpilrajn, Sur l'extension de l'ordre partiel, Fund. Math. 16, 386-389, 1930.

From tcipro!ramu@unix.sri.com Wed May 29 16:39:55 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Sat, 27 Oct 90 18:07:44 EDT
To: concurrency
Subject: [tcipro!ramu@unix.sri.com: modelling concurrency with partial orders]
From: Ramu Iyer <tcipro!ramu@unix.sri.com>
Sender: meyer@theory.lcs.mit.edu

Date: Fri, 26 Oct 90 16:09:54 PDT
Cc: sri-unix!cs.stanford.edu!pratt@unix.sri.com,

sri-unix!anna.stanford.edu!dcl@unix.sri.com, armen@unix.sri.com,
dbb@unix.sri.com, ramu@unix.sri.com, bads@unix.sri.com
In-Reply-To: Albert R. Meyer's message of Fri, 26 Oct 90 14:01:09 EDT
<9010261801.AA13008@stork>

>>>>> On Fri, 26 Oct 90 14:01:09 EDT, unix.sri.com!theory.lcs.mit.edu!meyer
(Albert R. Meyer) said:

Albert> I support most of your remarks, but I don't think we should
accept
Albert> David Luckham's formulation of the issue as
Albert> (1)   Linear versus Partial Order
Albert> but rather emphasize
Albert> (2)   Interleaving Nondeterminacy versus Concurrency
^^^

Here are three references that discuss these pioneering issues (^^^^) :

L. Castellano, G. De Michelis, L. Pomello. Concurrency vs Interleaving:
An Instructive Example. Bulletin of the EATCS, 31, 1987, pp. 12-15.

D.B. Benson, Concurrency and Interleaving are Equally Fundamental.
Bulletin of the EATCS, 33, 1987.

W. Reisig, Concurrency is More Fundamental than Interleaving, Bulletin
of the EATCS, ??, 1988.

Cheers,

--Ramu Iyer

Email: ramu%tcipro.uucp@unix.sri.com

From pratt@cs.stanford.edu Wed May 29 16:39:55 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Sat, 27 Oct 90 18:12:24 EDT
To: concurrency
Subject: modelling concurrency with partial orders
From: Vaughan Pratt <pratt@cs.stanford.edu>
Sender: meyer@theory.lcs.mit.edu

Date: Sat, 27 Oct 90 00:55:01 PDT

>>>From: tcipro!ramu@unix.sri.com (Ramu Iyer)
>>>Subject: modelling concurrency with partial orders
>>>Here are three references that discuss these pioneering issues (^^^^) :
>>> <3 references from 1987-88: Castellano et al, Benson, Reisig>

I'd like to suggest some earlier dates than 1987 or 1988 as more
suitable candidates for "pioneering."

The earliest proposal I'm aware of to model concurrency with partial
orders is Irene Greif's MIT Ph.D. thesis from 1975. Jan Grabowski and
Nielsen-Plotkin-Winskel both have 1981 journal papers on partial orders

for concurrency, with both parties reporting on work done at the end of the 1970's. C.A. Petri allegedly had advocated partial orders long ago, though not in writing as far as I'm aware.

Unlike these pioneers I did not appreciate the need for partial orders in concurrency myself until 1980. This was not for want of experience with concurrent computing. I had implemented various interrupt-driven packages in 1967-69, and I wrote and thought a fair bit about concurrency during the 1970's (1972: thesis chapter on sorting networks; 1974: showed with Larry Stockmeyer that P=NP on parallel computers; 1974-5: two circuit complexity results; 1976: solved the mutual exclusion problem for unreliable processes with Ron Rivest; 1979: axiomatized process logic).

But I did not appreciate the advantages of partial orders for concurrency until early 1980 when I was trying to understand Brock and Ackerman's paper. My pomset campaign began with my POPL-82 paper on that subject, "On the Composition of Processes" which proposed formalizing Brock and Ackerman's solution to their anomaly in terms of partially ordered multisets. I coined the abbreviation "pomset" a few months later.

I wrote a short paper on applying pomsets to the Two-Way-Channel-With-Disconnect problem for the 1983 concurrency workshop in Cambridge UK, LNCS 207, as well as a statement I circulated at IFIP-83 a week after that conference as part of a concurrency panel session chaired by Robin Milner in which I argued the case for pomsets. I also spoke about pomset semantics at Logics of Programs 1983 (no written paper unfortunately), and again in LOP 85.

This last paper was subsequently published in International Journal of Parallel Programming, 15:1, 33-71, 1986, as "Modeling Concurrency with Partial Orders" (same title as the subject line of the last 10 messages). (If you don't have that journal in your library you can retrieve this paper by anonymous FTP from boole.stanford.edu as /pub/ijpp.{tex,dvi}.)

I reproduce here the arguments I gave in that 1986 paper in support of partial orders. Note particularly item (v), which begins

> (v) "A serious difficulty with the interleaving model is that exactly what is interleaved depends on which events of a process one takes to be atomic."

and goes on to explain how refinement (as it is now called) distinguishes a|b from ab+ba and hence makes the meaning of interleaving dependent on granularity. While I know of no prior reference in the literature to the use of refinement to distinguish a|b >from ab+ba I'm sure the idea had occurred to many people before, even if writing it down had not.

See also the postscript-1990 at the end, on the outcome of my long-standing problem of axiomatizing the equational theory of concatenation and interleaving for formal languages. It is noteworthy that the solver independently invented pomsets for the express purpose of solving this purely interleaving question.

=======Extract from "Modeling Concurrency with Partial Orders, 1986=======

1.2 Why Partial Orders?

Strings arise naturally in modelling ongoing sequential computation, whether the symbols in the string correspond to states, commands, or messages. Thus the string uvu may model the sequential execution of three commands u,v,u, or a transition from state u to state v followed by a transition back to u, or a sequence of three messages u,v,u transmitted sequentially on some channel.

Strings are linearly ordered sets, or rather linearly ordered multisets (since repetitions are possible), of symbols from some alphabet. In unison with the workers mentioned at the end of this section we advocate partial orders in place of linear orders in modelling concurrent computation. At present however partial orders have nowhere near the popularity of linear orders for modelling concurrent computation. This could be for any of the following reasons.

(i) Languages and their associated operations, particularly union, concatenation, Kleene star, and shuffle, provide a natural model for the corresponding programming language control structures: choice, sequence, iteration, and concurrency. The behavior of languages under these operations has been studied intensively for more than two decades. Thus languages provide a familiar and well-understood model of computation. In this model the linear order on the elements of a string is interpreted as the linear temporal order of events, and the operations on languages may be interpreted as control structures: concatenation as begin-end sequencing, star as iteration, shuffle as concurrency, etc.

(ii) Every poset is representable as the set of its linearizations. This theorem would appear to confer on linear orders the same representational ability as partial orders.

(iii) Linear orders appear to be faithful to physical reality. In the practical engineering world, as opposed say to the physicist's relativistic world, instantaneous events have a well-defined temporal order, justifying the assumption of linearly ordered time. Furthermore, in any rigid system temporal order is well-defined even in a relativistic model. Any departures from rigidity are assumed to be sufficiently minor in practice as to justify adhering to a linear-order model.

Reason (i) would lose most of its force if partial orders were to be equipped with operations analogous to those of formal languages that could be interpreted as programming language control structures. This is just what this paper does; some of the operations on pomsets that we introduce correspond to more or less familiar programming language constructs, others are merely candidates for possible future programming or hardware languages.

Reason (ii) is based on the following well-known theorem, which shows that a partial order can be represented as the set of its linearizations.

{\bf Theorem 1.} The intersection of the linearizations of a partial

order is that partial order.

(For the purposes of defining intersection, a partial order is considered to be its graph, that is, the set of all pairs (a,b) such that $a\leq b$.)

This theorem is easily proved under the (non-obvious) assumption that every partial order has at least one linearization, by showing that any partial order in which a and b are incomparable can be extended to one in which $a<b$ and to another in which $b<a$.

This theorem about posets runs into two difficulties when trying to apply it to processes modelled as sets of pomsets. The theorem generalizes neither to {\it pomsets} nor to sets of {\it posets}, and {\it a fortiori} not to sets of pomsets. We will return to this issue in section 2.6, after the necessary definitions have been given.

Reason (iii), that the engineer's world is linear in time, fails in at least three situations: complex systems, nonatomic events, and relativistic systems. Beyond a certain scale of system complexity it becomes infeasible to keep thinking in terms of a global clock and a linear sequence of events. A cover story in the magazine Electronics$^{(3)}$ describes a growing trend in the design of logic circuits to eliminate global clocks and rely more on self-timed circuits. On a larger scale asynchrony has been with us for a long time. When a large number of computers communicate with each other over channels whose delay is several orders of magnitude greater than the clock time of each computer, the concept of global time provides neither a faithful account of the concurrent computation of all those computers nor even a particularly useful one. There is no reason to suppose that the various instructions streams of these computers are interleaved to form one stream. Indeed it is much more convenient, both conceptually and computationally (e.g. when computing with such streams as part of reasoning about them) just to lay down these streams side by side and call this juxtaposition of streams a model of their concurrent execution. Data flowing between the computers may augment the order implicit in the juxtaposition, but this relatively sparse augmentation of the order is motivated by the actual mechanics of communication, unlike the more stringent and totally artificial ordering requirement of completely interleaving the streams.

A concrete situation that may make this more compelling consists of a ship rolling somewhere in the Pacific, in satellite communication with another ship in the Indian Ocean. The events on the buses of the computers on each ship take place with a precision measured in nanoseconds, but the delay in getting a packet from one computer to another may be on the order of a second or more. The idea that the totality of events in the two computers has a well-defined linear ordering can have no practical status beyond that of a convenient mathematical fiction. Our position is that it is neither convenient nor mathematically useful. It is just as convenient, and more useful, to work with partial orders.

Nonatomic events provide another situation where linear orders break down. An event may be more complex than a moment in time. It may be an interval, in the sense of a convex subset of a linear order. It may be a set of intervals, such as a game punctuated by timeouts or a TV

movie punctuated by commercials. More generally still it may be some arbitrary set of moments. However even for such complex events it still makes sense to say that one event may precede or follow another, meaning that every moment of the first event precedes every moment of the second. Yet such events are clearly not linearly ordered.

Relativity provides yet another situation where time is not linearly ordered. In any nonrigid system, that is, one whose components are moving with respect to each other, simultaneity ceases to be well-defined and two moving observers can report contradictory orders of occurrence of a pair of events. Any system nontrivially subject to relativistic effects is a candidate for a partially ordered model of computation. Of course many systems will not be so subject, but we see it as an advantage of the partial-order approach that it applies equally well to relativistic and Newtonian (global-time) situations.

In addition to our responses to (i)-(iii), we have the following additional reasons for preferring partial orders.

(iv) Some concepts are only definable for partial orders, in particular orthocurrence, which amounts to the direct product of pomsets, which we define in full later. The solution given above to the problem of specifying the two-way-channel-with-disconnect contains two essential uses of orthocurrence, along with two less essential uses. The concept is an extremely natural and useful one for partial orders, but it is not at all obvious how one would go about defining it in a linear-order model, or even whether it is definable.

(v) A serious difficulty with the interleaving model is that exactly what is interleaved depends on which events of a process one takes to be atomic. If processes P and Q consist of the single atomic events a and b respectively then their interleaving is $\{ab,ba\}$. However if the same events a and b are perceived by someone else not to be atomic, by virtue of having subevents, then P and Q have a richer interleaving than $ab\cup ba$. It is reasonable to consider instantaneous events as absolutely atomic, but we would like a theory of processes to be just as usable for events having duration or structure, where a single event can be atomic from one point of view and compound from another. In the partial-order model what it means for two events to be concurrent does not depend on the granularity of atomicity.

(vi) In some situations pomsets appear to be easier to reason about than strings. For example it is relatively straightforward to axiomatize the equational theory of pomsets under the operations of concurrence and concatenation (Theorem 5.2$^{(4)}$). The corresponding theory for strings has resisted all attempts at its axiomatization. Gischer and the author have both worked extensively on the problem of whether this simply described theory has a finite axiomatization. The problem has been posed on two occasions at the (San Francisco) Bay Area Theory Symposium, generating interest but no answers in more than eighteen months.

[Postscript 1990: this problem was finally solved in 1988 by Steve Tschantz, an algebraist at Vanderbilt, who settled it in the affirmative by a truly beautiful argument only a week after I posed the problem along with a list of others at the end of an invited lecture at

a universal algebra conference in 1988. In doing so he reinvented pomsets quite independently as an essential tool in the proof; I had stated the problem purely for languages with no mention of pomsets at any point in my talk, which was about dynamic logic. -vp]

Vaughan Pratt

From sa@doc.imperial.ac.uk Wed May 29 16:39:56 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Tue, 6 Nov 90 18:52:38 EST
To: types, logic, concurrency
Subject: Computational interpretations of Linear Logic
From: Samson Abramsky <sa@doc.imperial.ac.uk>
Sender: meyer@theory.lcs.mit.edu

Date: Tue, 30 Oct 90 18:17:58 GMT

Albert,

I've just completed a paper on this topic. Please put the abstract on any of the lists (types, concurrency etc.) for which you think it would be appropriate.

Best wishes,

Samson

==========
Computational Interpretations of Linear Logic

Samson Abramsky

Imperial College

We study Girard's Linear Logic from the point of view of giving a concrete computational interpretation of the logic, based on the Curry-Howard isomorphism. In the case of Intuitionistic Linear Logic, this leads to a refinement of the lambda calculus, giving finer control over order of evaluation and storage allocation, while maintaining the logical content of programs as proofs, and computation as cut-elimination. In the classical case, it leads to a concurrent process paradigm with an operational semantics in the style of Berry and Boudol's Chemical Abstract machine. This opens up a promising new approach to the parallel implementation of functional programming languages; and offers the prospect of typed concurrent programming in which correctness is guaranteed by the typing.

From lamport@src.dec.com Wed May 29 16:39:57 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Wed, 7 Nov 90 17:11:10 EST
To: concurrency
Subject: Flame re distributed processes and granulity

From: lamport@src.dec.com (Leslie Lamport)
Sender: meyer@theory.lcs.mit.edu

Date: Tue, 6 Nov 90 17:13:59 -0800
To: meyer@theory.lcs.mit.edu
Subject: for the concurrency mailing list

I admire philosophers. They have so much to teach us. From Aristotle I learned that heavier bodies fall faster than lighter ones; Kant showed me that nonEuclidean geometry is impossible; and Spinoza proved that there can be at most seven planets. And now, the philosophers on the concurrency mailing list have told me all the things I can't do because I use a logic based on an interleaving model:

I can't reason about distributed systems.
~~~~~~
In 1982 I published a proof of the distributed algorithm then used in the Arpanet to maintain its routing tables ["An Assertional Correctness Proof of a Distributed Algorithm", Science of Computer Programming 2, 3 (Dec. 1982), 175-206]. Since then I have written more formal proofs of more complicated distributed algorithms.

I can't deal with changes in the grain of atomicity.
~~~~~~
In 1983 I published a paper ["Specifying Concurrent Program Modules", TOPLAS 5, 2 (April 1983) 190-222] containing:

> A specification of a queue, in which adding or removing an element is a single atomic operation.
>
> An implementation in which an element is moved into and out of the queue one bit at a time.
>
> A proof that the implementation satisfies the specification.

Nowadays, my standard approach to verification is to start with a high-level program having a coarse grain of atomicity, and to refine the grain of atomicity until I reach the desired program.

I can't reason about (nondiscrete) real time.
~~~~~~
At a workshop in 1988, I gave a one-hour lecture in which I:

> Specified a distributed spanning-tree algorithm having the requirement that the computation reach and maintain a correct configuration within a fixed length of (real) time.
>
> Gave an implementation using timers. I assumed only that timers ran at a rate of 1 +/- epsilon seconds per second, and that messages were delivered within delta seconds of the time they were sent. (Epsilon is any real number in the range [0, 1) and delta is any positive real number.)
>
> Sketched a proof that the implementation satisfied the specification.
~~~~~~

I have since written a detailed formal correctness proof.

I can't reason about programs without assuming a fixed granularity.
~~~~~~
A recent paper of mine ["win and sin--Predicate Transformers for Concurrency", TOPLAS 12, 3 (July 1990), 396-428] gave a rigorous correctness proof for the bakery algorithm. This algorithm makes no assumption about the grain of atomicity of its operations. (It was the first algorithm to achieve mutual exclusion without assuming lower-level mutual exclusion.)

I'm sure the philosophers can explain why I haven't really done these things. I'll be happy to listen to their explanations, as soon as they can use their philosophically approved methods to reason formally about something more complicated than a biscuit machine.

From pratt@cs.stanford.edu Wed May 29 16:39:58 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Thu, 8 Nov 90 16:23:15 EST
To: concurrency
Subject: Re: Flame re distributed processes and granulity
From: pratt@cs.stanford.edu
Sender: meyer@theory.lcs.mit.edu

Date: 08 Nov 90 12:58:19 PST (Thu)

On p.419 of the proceedings of Logics of Programs 81 (LNCS 131) appears the following extract from the panel discussion that wrapped up that conference. Context: Amir Pnueli had just expressed the wish that every paper on programming logic say something about how this programming logic is to be applied to proving something about programs.

"Nemeti: I'd like to protest a little bit about what you (Pnueli) said about our papers. The structure of our technological society is just not like that. There was a guy called Roentgen. You could have gone to him and said, `What are you doing playing around with these funny things of yours? Why don't you try to heal people who have colds?' There are theoreticians who are doing basic research, and there are less theoretical theoreticians, and there are technologists, so there is a whole spectrum of research in science. The theoreticians doing the basic research are really needed, because the basic ideas, the fundamental ways we look at things, come from there. Now, if you want to restrict them to report each time how this will be used, then it will result in impotence."

While I have nothing to add to this, I do have a question arising out of it. Who believes that "the basic ideas, the fundamental ways *systems people* look at things" come from the theoreticians? Do systems people believe this? And do theoreticians believe it?
-v
~~~~~~

From RJH@ai.mit.edu Wed May 29 16:39:58 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Mon, 12 Nov 90 11:19:35 EST
To: concurrency
Subject: re: Re: Flame re distributed processes and granulity
From: Robert J. Hall <RJH@ai.mit.edu>
Sender: meyer@theory.lcs.mit.edu

Date: Sat, 10 Nov 90 12:58 EST
To: pratt@cs.stanford.edu, concurrency@theory.lcs.mit.edu

> Date: Thu, 8 Nov 90 16:23 EST
> Subject: Re: Flame re distributed processes and granulity
> From: pratt@cs.stanford
>
> On p.419 of the proceedings of Logics of Programs 81 (LNCS 131) appears...
>
> "Nemeti: ..." (regarding need for theoreticians, etc)

It seems to me this quote does not directly address Lamport's complaint which was, I believe, that the theoreticians on this list seem to be making false claims (as enumerated by Lamport). He seemed to be fraternally suggesting that one way of avoiding such false claims may be to keep a closer contact between theory and practice, if indeed theory is attempting to have some benefits for practice. In particular, if one's claim is to the effect that a technologist "can't do" something using a theory, one must at least be more precise about what it means to do that thing. Obviously, Lamport believes he has successfully used the interleaving-based view to reason about multiple granularities, whereas previous discussions on the list seem to claim he can't have done so (similarly for the other issues raised).

-- Bob

From pratt@cs.stanford.edu Wed May 29 16:39:59 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Mon, 12 Nov 90 11:30:35 EST
To: "Robert J. Hall" <RJH@ai.mit.edu>
Cc: concurrency@theory.lcs.mit.edu
Subject: [pratt@cs.stanford.edu: Re: Flame re distributed processes and granulity]
From: pratt@cs.stanford.edu
Sender: meyer@theory.lcs.mit.edu

Date: 11 Nov 90 20:22:55 PST (Sun)

> It seems to me this quote does not directly address Lamport's complaint which was, I believe, that the theoreticians on this list seem to be making false claims (as enumerated by Lamport).

My quote addressed Leslie's complaint in the most direct way possible under the circumstances. Leslie did not identify any particular claim

made on the list. Rather he complained generally that certain contributors to the list, whom he did not specify, had claimed there were certain things he couldn't do, which he did specify. There have been various claims on this list about limitations of interleaving, but none that I recall making the claims Leslie was complaining about, nor any that conflicted with the evidence he adduced in support of his complaint.

One claim about interleaving in this forum is in my October 26 message to David Luckham. There I claimed that Szpilrajn's representation theorem for posets, that every poset is representable as the set of its linearizations, depends on several assumptions. For each assumption I showed informally in what way the theorem could fail in the absence of that assumption, in some cases giving pointers to where more detailed proofs of those failure modes could be found.

I see no logical connection between Leslie's complaint and my claim. And even if there were some connection, the existence of failure modes of trace-based logic when certain assumptions are violated in no way implies that every trace-based proof violating those assumptions must be unsound. I do not begrudge Leslie his sound proofs, however obtained.

The failure modes of Szpilrajn's theorem are not just mathematical curiosities but potentially real engineering problems. Perhaps Leslie knows how to take care of these problems using trace-based logic, but I don't see how his cited examples demonstrate this at all. How might a logic based on sets of traces deal with each of the following situations?

1. Distinguish the race implicit in a|b from the race-free situation implied by ab+ba.

2. Reason about observations made by a team of distributed observers who agree on what events happened but not in what order.

3. Reason about the possible interleavings of two concurrent sine waves. (Presumably one falls back on some other technique for combining traces than interleaving them.)

> He seemed to be fraternally suggesting that one way of avoiding such false claims may be to keep a closer contact between theory and practice

I found no hint of such a suggestion in Leslie's message.
-v

From lamport@src.dec.com Wed May 29 16:40:00 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Mon, 12 Nov 90 11:31:28 EST
To: concurrency
Subject: [lamport@src.dec.com: for the concurrency mailing list]
From: lamport@src.dec.com (Leslie Lamport)
Sender: meyer@theory.lcs.mit.edu

Date: 10 Nov 1990 1721-PST (Saturday)

Dear Dr. Roentgen,

I am writing to congratulate you on the success of your continuing experiments with X-rays. I can imagine your dismay at the many charlatans who have used your X-rays to justify "invisible ray" theories based on fancy rather than science. And those silly French physicists with their N-rays! How fortunate that we live in a society where scientific validity is determined by rigorous experiment. I presume you are aware of the disturbing developments in the Soviet Union, where Dr. Lysenko attacks the work of Mendel on idealogical grounds. I'm afraid it will be many years before the Soviets permit sound research in genetics, since they value philosophical correctness above empirical observation.

Sincerely yours,

Leslie Lamport

From jcm@cs.stanford.edu Wed May 29 16:40:01 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Tue, 13 Nov 90 08:32:56 EST
To: concurrency
Subject: philosophy
From: John C. Mitchell <jcm@cs.stanford.edu>
Sender: meyer@theory.lcs.mit.edu

Date: Mon, 12 Nov 90 11:01:53 -0800

One of the few things I remember from my first course in philosophy, taught by John Perry, was that philosphy gets a bad name from the fact that as soon as some subfield starts to become useful (e.g., mathematics, physiscs, astronomy), it stops being called philosophy. To a certain degree, I think the same is true of "theoretical" computer science. One example is parsing, and another seems to be various forms of circuit and protocol verification.

From pratt@cs.stanford.edu Wed May 29 16:40:03 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Tue, 13 Nov 90 08:47:40 EST
To: concurrency
Subject: Re: [lamport@src.dec.com: for the concurrency mailing list]
From: pratt@cs.stanford.edu
Sender: meyer@theory.lcs.mit.edu

Date: 12 Nov 90 13:20:57 PST (Mon)

Leslie's "fraternal suggestions" could easily create the impression that he is for interleaving and I am against. This construes my

position too narrowly. Let me set this in the historical perspective of a FOCS-76 paper by Ron Rivest and myself that Leslie attacked at that time.

Ron and I had given an interleaving proof of correctness of our solution of the mutual exclusion problem for two unreliable processes. The gist of our proof was that the many paths through our code fell into 6 classes, permitting a straightforward case analysis each case of which had a simple argument. We found this program by making small random perturbations to a tiny but buggy mutual exclusion protocol. Even after looking at the four instructions of our resulting program for a long time we had absolutely no intuitive understanding of why that perturbation was correct and others very like it were not!

Leslie protested to us that such a proof as ours based on classification of interleavings was inappropriate. He showed us a proof of correctness of our procedure based on a theory he had evolved of why it worked.

Had we considered our program to be the final word on this subject we could well have agreed with Leslie that having an "insightful theory" of our code was worthwhile. After all, the method used to find a prime need not be the best method to convince someone of its primality.

However even assuming that Leslie's proof gave us the additional insight into our procedure that he claimed it should, it seemed to us that our procedure was surely just one of more to come, and that the effort of making up such a theory after the fact was therefore wasted. Furthermore our strategy for discovering new such algorithms depended critically on the automatic nature of interleaving analysis; we had no idea how to write a program which given a random algorithm would generate a theory of how it might work, whereas we knew how to enumerate and check all its interleavings mechanically in a short time.

This was borne out by the subsequent extension of our work by Mike Fischer and Gary Peterson, published in STOC-77. Whereas our solution involved I think 7 states for each of two processes they had 3 states each (3+3, and another solution with 4 states at one process and 2 states at the other, 4+2). They found their very economical solutions by trying out various possible programs and checking all interleavings of each until they found one that worked. They used two such checkers, written independently by Mike and Gary.

Gary did come up with a Lamport-style after-the-fact theory of why their 3+3 mutex procedure worked. Mike's comment to me about that proof was that since they'd already mechanically checked correctness simply by running their procedure through all possible interleavings, this more conventional proof, which had to be manually checked, added nothing to Mike's confidence in the correctness of their procedure, and indeed seemed to him more likely to contain lacunae.

Now I can see clearly that such post hoc theories of these procedures might have a certain esthetic attraction, and might even be useful. My point is not to fault Leslie for coming up with such a theory but only to demonstrate that I am not a religious zealot on the use of interleaving analysis in concurrency. Indeed I still know of no

simpler proof of our FOCS-76 algorithm than our 6-case interleaving analysis, and if I were writing it up today I would still prove it correct in that way. Moreover I have no problem with the use of interleaving in any situation to which it is applicable. In particular I have no quarrel with Leslie on the applicability of logics based on interleaving to the problems he listed in his flame.

I trust that Leslie uses a different logic to prove the correctness of his algorithms from the one he uses to prove that those of us who have in the course of twenty-five years gradually moved from writing concurrent programs to reasoning abstractly about them have by so doing turned themselves into charlatans. This was the only fraternal suggestion I found in Leslie's two messages. A century ago the same logic would have demonstrated with equal validity that Cantor was a charlatan.

Vaughan Pratt

From mischu@allegra.tempo.nj.att.com Wed May 29 16:40:03 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Tue, 13 Nov 90 08:49:13 EST
To: concurrency
Subject: Begin-the great debate-End
From: mischu@allegra.tempo.nj.att.com (Michael Merritt)
Sender: meyer@theory.lcs.mit.edu

Date: Mon, 12 Nov 90 15:45:48 EST

While I can't pretend to follow all the subtleties
of the ongoing discussion, I do have a fairly specific
query for the proponents of partial orders, growing out
of my fairly extensive experience in modelling concurrent
algorithms using interleaving.

Specifically, I generally model operations as consisting
of a sequence of two atomic events, the beginning and
ending of the operation. When communication
is involved, these are described as requests and replies.
(E.g. Request-Read(register-x), Reply-Read(register-x,value).)
When operations run concurrently, their begin and end events
occur in an interleaved sequence. Using this approach,
I would resolve the a|b vs ab+ba debate by denoting a and
b by begin-a,end-a and begin-b,end-b, respectively.
Then a|b is the set of sequences:
{(begin-a,end-a,begin-b,end-b),
 (begin-b,end-b,begin-a,end-a),
 (begin-a,begin-b,end-a,end-b),
 (begin-b,begin-a,end-b,end-a)}

and ab+ba is the (very different set)

{(begin-a,end-a,begin-b,end-b),
 (begin-b,end-b,begin-a,end-a)}.

Similar causally distinct processes would

seem to be distinguished by such a semantics, as well.

When refining an operation, I never change the symbols
denoting the begin and end of the operation. I simply
change the (internal) operations that occur between
the begin and end actions.

The begin/end distinction is particularly useful at
interfaces, where the system issues a request and
the environment responds, or vice-versa.

I am interested in reactions to this method of
resolving the (over-emphasized, in my mind) debate.

On multiple observers of concurrent
systems: it seems to me that
an accurate model of such systems should distinguish
between the occurance of an event and its observation.
(I think even the physicists do this much.)
A run of such a system then consists of an interleaved sequence
of events and their observations. The subsequence
experienced by a single observer is obviously consistent
with a set of runs.

What's missing?

I'll send references and/or papers if anyone is interested
in seeing these ideas applied to algorithmic problems.
But I should say that I work within the formal framework
(I/O automata) devised by Nancy Lynch and Mark Tuttle.

Now, it is true that in reasoning about concurrent
systems I often find myself reasoning about partial
orders embedded in the language (set of sequences)
denoted by the system, and I am interested in tools
that would help me do that. But I am also reluctant
to give up induction as a proof technique. Why can't
I have both?

Michael Merritt

From pratt@cs.stanford.edu Wed May 29 16:40:04 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Wed, 14 Nov 90 10:42:20 EST
To: concurrency
Subject: Homotopy = homobject
From: Vaughan Pratt <pratt@cs.stanford.edu>
Sender: meyer@theory.lcs.mit.edu

Date: Tue, 13 Nov 90 15:55:21 PST

Cellular geometry arises both with categories, starting with
Ehresmann's 1965 notion of an n-category, and with concurrency as per
my POPL-91 paper and also as per a paper that David Murphy just brought
to my attention, ``Deterministic Asynchronous Automata,'' Mike Shields,

Proc. Formal Models in Programming (Ed. E.Neuhold, G.Choust), Elsevier 1985.

I'm not sure who in category-land cares about homotopy in n-categories, but it is the basis for distinguishing true from false nondeterminism in my POPL paper. As David points out to me, a special case of homotopy can be found in Mazurkiewicz's independence relation: the independence of a and b should be identified with the paths ab and ba being homotopic, as in a|b. In ab+ba however these two paths are not homotopic: one has to decide which of the ab or ba paths one is going to follow.

While the following is obviously too cryptic for general consumption, I am mentioning the idea here for two reasons: to mumble my obscure thought processes concerning true nondeterminism out loud on the concurrency and category lists, and to find out if this definition of homotopy as homobject rings a bell with anyone. It seems so obvious that I am fully expecting it to have been around for decades, at least somewhere. It just isn't in the places I've looked so far. If it is spelled out somewhere, any attempt on my part to expand on the mumbling below may not be necessary.

Here's the idea. It seems to me that a very natural definition of homotopy is arrived at by identifying homotopy with homobject, in the enriched category sense. That is, the homotopy of the paths from x to y is the homobject [x,y], or d(x,y) in the notation of Casley et al, CTCS-89, Manchester, LNCS 389, the "distance" from point x to point y. (The basic law governing homobjects is the abstract triangle inequality, which is why it is appropriate to think of the homobject [x,y] as an abstract distance d(x,y). This view is due to Lawvere 1974.)

Hence homotopy is governed principally by the triangle inequality, the basic law of enriched category theory. In this sense the homotopy from x to y and the distance from x to y become the same thing.

The homotopy of an ordinary category is discrete because its homobjects are sets. The homotopy of a set is nonexistent because sets don't have homobjects worth mentioning (all points are equidistant). The homotopy of a poset is trivial because its homobjects contain either no elements (i.e. paths) or one.

The intuitive notion of homotopy as an equivalence relation on paths arises for categories whose homobjects are equivalence relations; then [x,y] is a set (X,~) of paths and an equivalence ~ on paths whose blocks are the homotopy classes. However it would seem nicer to take arbitrary categories for homobjects, the homotopy of a 2-category.

The homotopy of an order-enriched category lies between that of categories and 2-categories. The simplest case of this arises for a monoid (1-object category), the basis for my recently developed "action logic" ACT (pub/jelia.{tex,dvi} via ftp from boole.stanford.edu). Action logic is accessible to anyone who understands lattice theory, and employs no categorical language or explicit categorical concepts, yet it contains interesting homotopy in the above sense, in a way that Boolean logic and intuitionistic logic as cartesian closed posets do not.

In a closed category homotopy is internalized just like a homobject, via exponentiation/implication. That is, the entire homotopy [x,y] can be compressed into the single point b^a or a=>b as its internal representation. The homotopy so coded can then be recovered as the homotopy from I (the unit of the closed category) to that point, via the isomorphism between [I,a=>b] and [a,b]. Thus isomorphic copies of all homotopy present in a closed category can be found radiating out >from its unit.

In the case of action logic the homotopies so radiating out from I (called 1 there) are exactly the theorems of action logic.

I know the above must look to many of you rather unrelated to the traditional geometry of triangles, circles, and squares. Hopefully someone will someday volunteer to draw enough pretty pictures of this really very simple notion of homotopy to dispel any remaining mystery about it. You will find a few such pictures at the end of the action logic paper, of paths with fixed endpoints sweeping across surfaces, which should fit right in with any prior intuition you had about homotopy.

Vaughan Pratt

From rvg@frege.stanford.edu Wed May 29 16:40:05 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Wed, 14 Nov 90 10:47:43 EST
To: concurrency
Subject: Begin-the great debate-End
From: Rob van Glabbeek <rvg@frege.stanford.edu>
Sender: meyer@theory.lcs.mit.edu
In-Reply-To: Michael Merritt's message of Tue, 13 Nov 90 08:49:13 EST
<9011131349.AA01750@stork>

Date: Tue, 13 Nov 90 16:53:13 PST

> Date: Tue, 13 Nov 90 08:49:13 EST
> From: mischu@allegra.tempo.nj.att.com (Michael Merritt)
> Sender: meyer@theory.lcs.mit.edu
> Date: Mon, 12 Nov 90 15:45:48 EST
>
> While I can't pretend to follow all the subtleties
> of the ongoing discussion, I do have a fairly specific
> query for the proponents of partial orders, growing out
> of my fairly extensive experience in modelling concurrent
> algorithms using interleaving.
>
> Specifically, I generally model operations as consisting
> of a sequence of two atomic events, the beginning and
> ending of the operation. When communication
> is involved, these are described as requests and replies.
> (E.g. Request-Read(register-x), Reply-Read(register-x,value).)
> When operations run concurrently, their begin and end events
> occur in an interleaved sequence. Using this approach,
> I would resolve the a|b vs ab+ba debate by denoting a and

b by begin-a,end-a and begin-b,end-b, respectively.
Then a|b is the set of sequences:
{(begin-a,end-a,begin-b,end-b),
 (begin-b,end-b,begin-a,end-a),
 (begin-a,begin-b,end-a,end-b),
 (begin-b,begin-a,end-b,end-a)}

and ab+ba is the (very different set)

{(begin-a,end-a,begin-b,end-b),
 (begin-b,end-b,begin-a,end-a)}.

I am interested in reactions to this method of
resolving the (over-emphasized, in my mind) debate.

This idea occurs in many teksts on interleaving semantics. The following formulation is taken from HOARE 85: `The actual occurrence of each event in the life of an object should be regarded as an instantaneous or an atomic action without duration. Extended or time-consuming actions should be represented by a pair of events, the first denoting its start and the second denoting its finish.'

The idea of splitting events with a duration is a very powerful one, and makes that many features of concurrent systems can in principle be modelled adequately in interleaving semantics. However, in a lot of cases one can doubt whether it is *natural* to model a concurrent system in interleaving semantics only, even if this can be done theoretically.

Take for instance the extremely useful distinction between functional behaviour and performance. The idea is that for a given (distributed) system one first studies whether it is functionally correct, and only when this has been shown (ideally), one moves to questions concerning its time/space complexity. The problem that we see in the above `solution' for dealing with actions with duration, is that the issues of functional behaviour and performance get mixed up. The following trivial example to illustrate this point comes from Frits Vaandrager, but is for the opportunity adapted by me to a setting with biscuit machines.

Suppose we are interested in a vending machine which produces two biscuits when a coin is inserted and then returns to its initial state. The machine should satisfy the following trace-specification S:

$$2*(coins - 1) <= biscuits <= 2*coins,$$

i.e. for each sequential trace of the machine we should have that the number of occurrences of the action biscuit in this trace is bounded by 2 times the occurrences of the action coin and 2 times (coins -1).

A first proposal for a machine with this property is described by the recursion equation

VMS = coin ; bisc ; bisc ; VMS .

An alternative proposal could be

VMS' = coin ; (bisc || bisc) ; VMS' .

In interleaving semantics we of course have: VMS = VMS'. This means that under certain conditions we may infer that VMS and VMS' have the same functional behaviour. So as soon as we have shown in some appropriate calculus that VMS satisfies S, we can conclude that also VMS' satisfies S. We now can make two observations:

1. Especially when dealing with the functional aspects of the system the above choice of actions seems very natural. Working with actions begin-coin, end-coin, etc. gives an overhead which nobody would like to have. The traditional problem of interleaving semantics, namely combinatorial state explosion, will arise even faster in case actions are split. Moreover the functional equivalence of the two machines can not so easily be determined.
2. Intuitively the situation concerning performance is clear: machine VMS' is *faster* than machine VMS because it will work in parallel. So why not build a semantic theory in which this intuition can be formalised?

In the view of Frits and me the above considerations strongly plead for a semantic theory with at least *two* notions of equivalence: (1) an interleaving equivalence for dealing with functional aspects, and (2) a non-interleaved equivalence for dealing with performance. The idea is then that at the non-interleaved level actions can have duration and structure, whereas at the interleaving level one abstracts from these aspects and imposes a total order on the actions.

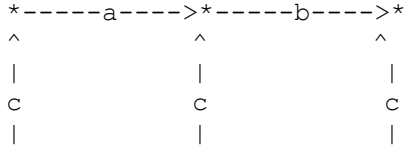
One of the options for the non-interleaved equivalence - in the spirit of Hoare and Merritt - is to say that two processes are to be regarded as equivalent iff their splitted versions have the same interleavings. This non-interleaved semantics lays somewhere between interleaving semantics and partial order semantics.

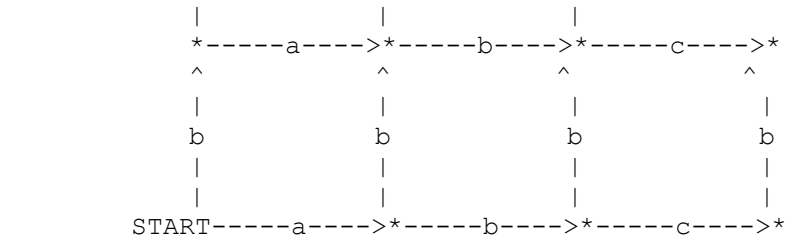
> Similar causally distinct processes would
> seem to be distinguished by such a semantics, as well.

However not *all* causally distinct processes can be distinguished by such a semantics. Especially when permitting autoconcurrency (the independent execution of two events which on the chosen level of abstraction are considered to be occurrences of the same action) the proposed semantics falls short in a number of aspects:

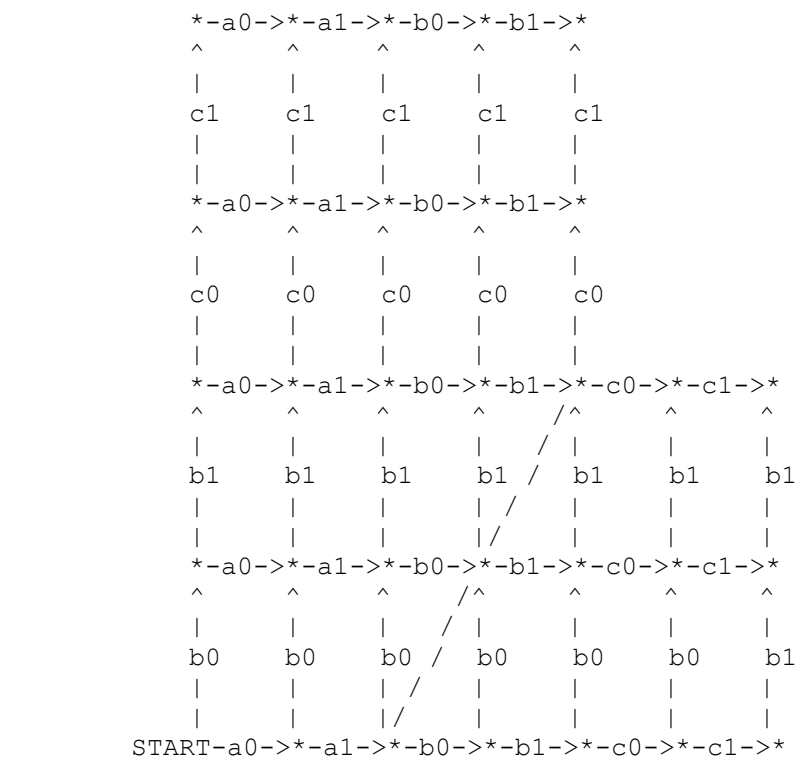
Consider the processes (abc || b) + (ab || bc) and (ab || bc).

Here ab is the sequential composition of actions a and b, ab || bc is the parallel and independent composition of the processes ab and ac, and P + Q denotes a (nondeterministic) process that behaves either like P or like Q. If we don't care for branching time the left hand side process can be represented by the automata:





After splitting al actions in two the automaton looks like:



By mirroring the right wing of the automaton in the displayed diagonal one easily sees that all interleavings originating from (abc || b) are already present in the big square (ab || bc). Hence the two processes (abc || b) + (ab || bc) and (ab || bc) (if allowed to exists) are equivalent in Merritt's semantics. Nevertheless one can argue that (ab || bc) can be executed *faster* than (abc || b) + (ab || bc): If all actions a, b and c are considered to take one hour each, and the automata don't wait needlessly, the left hand automaton has the possibility to need one hour more than the right hand one.

A slightly more complicated example shows that in fact it makes a difference whether actions are split in two or in three (considering start, end and halfway actions for instance)!

> When refining an operation, I never change the symbols denoting the begin and end of the operation. I simply change the (internal) operations that occur between the begin and end actions.

In case you don't allow autoconcurrency - as occurs in the example

above - that's fine. In order to capture the more general case, where processes like the one above are considered, you have to do some bookkeeping linking end actions explicitely to begin actions. Otherwise the operation of refining an action fails to be a congruence for your semantical equivalence, i.e. cannot be defined consistently. Counterexamples on request.

> The begin/end distinction is particularly useful at interfaces, where the system issues a request and the environment responds, or vice-versa.

Don't misunderstand me; I do think the distinction can be applied usefully.

> On multiple observers of concurrent systems: it seems to me that an accurate model of such systems should distinguish between the occurance of an event and its observation. (I think even the physicists do this much.) A run of such a system then consists of an interleaved sequence of events and their observations. The subsequence experienced by a single observer is obviously consistent with a set of runs.
>
> What's missing?

The coordination, at the end of each single run of the investigated system, of the data obtained by different observers. Suppose that the system (a || b), where the occurrences of a and b may even be considered to be instantaneous events, runs only once, and is observed by two experimentors (traveling in different inertial frames for instance). Then it may happen that one of them observes ab whereas the other observes ba. If they now would simply drop there observations into a big bag of interleavings where also sequences that where observed during other runs of the system are gathered, their work does not provide evidence for the fact that they are observing (a || b) rather than (ab + ba). However, if the two meet after their observations and compare notes, they may realize that they percieved the very same run of the system in a different way. From this they conclude that a and b must have been executed independently.

> I'll send references and/or papers if anyone is interested in seeing these ideas applied to algorithmic problems.

Send me.

> But I should say that I work within the formal framework (I/O automata) devised by Nancy Lynch and Mark Tuttle.

Oh... Well, send me anyway.

> Now, it is true that in reasoning about concurrent systems I often find myself reasoning about partial orders embedded in the language (set of sequences) denoted by the system, and I am interested in tools that would help me do that. But I am also reluctant to give up induction as a proof technique. Why can't I have both?

Yes, why can't you?

   Michael Merritt

Rob van Glabbeek

From pratt@cs.stanford.edu Wed May 29 16:40:05 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Wed, 14 Nov 90 10:53:35 EST
To: concurrency
Subject: DO the great debate CONTINUE
From: pratt@cs.stanford.edu
Sender: meyer@theory.lcs.mit.edu
In-Reply-To: Your message of Tue, 13 Nov 90 08:49:13 EST.
             <9011131349.AA01750@stork>

Date: 13 Nov 90 12:30:27 PST (Tue)

> From: mischu@allegra.tempo.nj.att.com (Michael Merritt)
> Specifically, I generally model operations as consisting
> of a sequence of two atomic events, the beginning and
> ending of the operation
> ...
> What's missing?

In fact for deterministic parallel constructs this is a provably sound abstraction (or contrapositively, languages are a fully abstract model with respect to the semantics defined by just sets of such begin-end pairs). Theorem 2.3 of Gischer's thesis (Stanford report STAN-CS-84-1033, 1984) is that two pomsets are language equivalent iff

they are digram equivalent. (I don't know why Jay omitted this theorem >from the journal version, TCS 61:199-224.) That is, take the operations of one's language to be all pomset-definable operations (namely concatenation, concurrence, N(a,b,c,d), etc.), and let the variables range over arbitrary sets of strings. Then the resulting equational theory, consisting of all equations between terms of this language that are universally true in this interpretation, is the same theory as obtained when the strings are restricted to strings of length two.

Perhaps you don't care about *all* pomset definable operations, but presumably you at least care about two of them, namely concatenation and interleaving. This case can be formally defined and treated without mentioning pomsets or true concurrency at all. In this case the theorem is just about how sets of strings combine under concatenation and interleaving. Jay's theorem 2.3 applies equally to this restricted case.

This seems to provide positive support for the two-event interpretation of operations. But in fact there *is* something missing, namely nondeterminism. (Pomset definable operations such as concurrence, although indeed nondeterministic from a false-concurrency perspective, are properly considered deterministic in the true concurrency world.)

In 1988 Van Glabbeek and Vaandrager asked whether digrams sufficed for the richer language obtained by expanding this deterministic language of pomset-definable operations with the nondeterministic choice operator p+q, interpreted simply as language union. Their initial answer was that a gap now appeared between digrams and trigrams, which they showed with an automaton they called the "owl" because of its shape. They have subsequently extended this result to show that (n+1)-grams make finer distinctions than n-grams for all n. (This incidentally is a *very* nontrivial result, which took them a long time to find. I tried very hard even just to separate 3 from 4 without success, I guess my brain is out to lunch these days.)

So why don't practitioners notice these phenomena in their work? Presumably because they don't leap out at the casual observer. For just this reason 19th century engineers did not notice discrepancies in their day-to-day work due to relativity and quantum mechanics. It is true that any engineer whose measurements depended on the velocity of light not changing between summer and winter by an amount as large as twice the earth's orbital velocity would be grateful for relativity, but how many engineers in those days felt this was a serious problem?

Nowadays surveyors who use $10,000 interferometers routinely in the field to measure hundreds of feet to an accuracy of hundredths of an inch would find these seasonal variations in the velocity of light very distracting if they existed. The earth's orbital velocity is 29.8 km/s and light travels at 299,800 km/s, so according to the ether theory the length of a 500-foot boundary would appear to be gently oscillating at 32 nanohertz with a peak-to-peak amplitude of 1.2 inches.

By the same token Wien's law did have an odd bump, but how many practicing chemical and other engineers of the day had their work thrown off by it?

Nowadays quantum mechanics explains a host of phenomena that would have started accumulating without explanation at an alarming rate during this century had quantum mechanics not been in place to account for them.

But to early 20th century engineers relativity and quantum mechanics were just theoretical curiosities that one would only notice if one looked extremely closely in the neighborhood of where their delicate effects were to be felt. Perhaps more strikingly, it has been said that a common view among late 19th century physicists was that the structural aspects of physics had been fully elucidated, with the bulk of the remaining work being a matter of measuring everything more accurately.

I suggest that we have much the same situation here. Take the largest concurrent algorithm that anyone has ever proved correct. Is the future of concurrency just a matter of extending the proof techniques that worked there to yet larger code fragments? I don't think so, for the various reasons I gave in my message to David Luckham. As we pass to more widely distributed computations, as the ratio of end-to-end time over bit-to-bit time increases, as observations become more complex, and as glitching intrudes itself into yet more situations, the linear-time model will become a Procrustean bed that some may continue to find the equal of a Beautyrest mattress but that many others will find unreasonably painful.

> Now, it is true that in reasoning about concurrent systems I often find myself reasoning about partial orders embedded in the language (set of sequences) denoted by the system, and I am interested in tools that would help me do that. But I am also reluctant to give up induction as a proof technique. Why can't I have both?

I could not ask for a better example of reason (i) in my 1986 IJPP paper (obtainable by ftp from boole.stanford.edu as ijpp.{tex,dvi}, instructions in Boole's /pub/README) for why people prefer interleaving. Over the years people have built up a substantial workshop full of tools for manipulating strings and sets of strings. Put them in a partial order environment and they feel disoriented and deprived of their tools.

My answer to this reason was that we should remove it by building the tools needed for a universe in which time is partially ordered. To this end my IJPP paper developed a number of language constructs some of which like orthocurrence had no analog in the world of linear orders, and some of which like network composition could be defined for linear orders but were then vulnerable to the Brock-Ackerman anomalies in the presence of nondeterminism.

With regard specifically to induction, my recent paper "Action Logic and Pure Induction" (similarly obtainable from Boole as jelia.{tex,dvi}) shows how to do induction in a wide range of situations, going well beyond languages and binary relations. In commutative action logic the "horizontal" operation ab becomes concurrence, a|b. Yet one can still perform induction on iterated concurrence. Another interpretation of ab is orthocurrence, as per my IJPP 86 paper. Again one can do induction with iterated

orthocurrence. And as always one can do induction on iterated concatenation, i.e. the usual Kleene star but in other settings than languages and relations, e.g. pomsets, where the concatenation of pomsets is only linear when the given pomsets are linear.

If all you want is the ability to reason as you have always done by induction, that is no reason to replace pomsets by strings.

Tony Hoare disagrees with me that unfamiliarity with partially ordered time is a major obstacle to its greater adoption. I confess I don't have any strong evidence (though the above is one data point), but I do have a very strong feeling that if people felt that they could move >from linear time to partial without giving up *any* of their tools, and also appreciated the advantages I and others have been pointing out for partial orders, there would be a lot more such migration than at present.

The argument is sometimes made that linear time is fully abstract for concurrent computation and partial time is not (i.e. it makes unobservable distinctions), e.g. Bengt Jonsson in POPL-89, Jim Russell in FOCS-89, and I think others (I recently saw a mention by Tony Hoare of a similar sounding result by Mark Josephs). While this is true in the domain of Szpilrajn's theorem, outside its domain what happens is that partial time becomes fully abstract while linear time becomes unsound (asserts false equalities), see my paper on this with Gordon Plotkin (pp2.{tex,dvi} obtainable from Boole as above).

Given the choice of two theories such that, as one moves in and out of the domain of Szpilrajn's theorem, one theory varies between being fully abstract and not fully abstract, but always remaining sound, while the other varies between sound and unsound, but always remaining fully abstract, which would *you* choose?
-v

From lamport@src.dec.com Wed May 29 16:40:06 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Fri, 16 Nov 90 18:28:10 EST
To: concurrency
Subject: Reply to Pratt
From: lamport@src.dec.com (Leslie Lamport)
Sender: meyer@theory.lcs.mit.edu

Date: Thu, 15 Nov 90 11:43:10 -0800

Vaughan asks

> How might a logic based on sets of traces deal with each of the following situations?
>
> 1. Distinguish the race implicit in a|b from the race-free situation implied by ab+ba.
>
> 2. Reason about observations made by a team of distributed observers who agree on what events happened but not in what order.

3. Reason about the possible interleavings of two concurrent sine waves. (Presumably one falls back on some other technique for combining traces than interleaving them.)

The answer is that I don't know and I don't care. These questions never arise in my work.

How can it be that I find these issues to be irrelevant when Vaughan, who's an intelligent and (generally :-) reasonable computer scientist, considers them important? To answer this, I must begin with a discussion of the nature of science.

Any science is ultimately concerned with the real world. A scientific theory consists of a mathematical formalism together with a way of relating that formalism to the real world. For example, Newtonian mechanics consists of a mathematical theory of point masses moving along trajectories in mathematical 3-space, together with a way of relating those mathematical objects to the motions of real objects, such as planets. Note that not every concept in the mathematcal formalism need correspond to something in the physical reality--for exaample, the vector potential of classical electromagnetism has no physical counterpart.

Any useful scientific theory has a limited domain of application. A theory-of-everything is generally good for nothing. Newtonian mechanics can't describe the flow of fluids, for which one needs a theory containing mathematical concepts corresponding to friction and viscosity.

For computer science, the real world usually consists of computers (hunks of wire and silicon) executing programs. Theories in computer science are based on such diverse mathematical formalisms as Turing machines, temporal logic, and CCS.

To judge a scientific theory, one must know what its claimed domain of applicability is. The work of mine that I mentioned in an earlier message involves a theory whose domain is the specification and verification of functional properties of concurrent systems. I won't describe this domain here, except to note that "functional properties" include eventual termination and upper and lower time bounds on termination; they exclude probability of termination and expected time to termination.

Computer scientists have tended to be vague about the domain of applicability of their theories. As a result, people who work in one theory often think their theory is good for everything. For example, I have heard people say that the algebraic laws of CCS make it good for verifying distributed algorithms. CCS works fine for verifying biscuit machines. It is hopelessly impractical for verifying even the simplest distributed spanning tree algorithm, let alone the more complex algorithms that system builders use. Robin Milner realizes this (I've discussed it with him), but many of his disciples don't.

This doesn't mean that CCS is worse than my theory; just that it has a different domain of applicability. It is as silly to say that CCS is better or worse than my theory as it is to say that physics is better or worse than biology. Human nature being what it is, almost all

physicists believe in their hearts that physics is more important than biology. However, physicists understand that not everyone believes this, so a university will teach biology even if the dean of faculty is a physicist. One wishes that computer scientists were as understanding.

I think there are two general reasons why a concept that's important to theory A may be absent from theory B:

(i) The concept is irrelevant to the domain of applicability of theory B.

(ii) The concept belongs to the mathematical formalism of theory A and, even though the two theories have overlapping domains of applicability, theory B's method of translating reality into mathematical formalism makes the concept irrelevant or meaningless.

Case (ii) is the more insidious cause of misunderstanding. People get so used to their favorite theory that they confuse its mathematical formalism with physical reality. For example, some advocates of CCS will say that my theory is deficient because it doesn't distinguish between internal and external nondeterminism. They don't realize that internal/external nondeterminism is part of the mathematical formalism of CCS, not a property of physical reality, so there is no reason why it should be a meaningful concept in another theory. This error is not confined to one side of any ideological fence. A colleague of mine once asserted that he could prove any kind of property of a program, since he could prove safety and liveness properties and any property is the conjunction of a safety and a liveness property. He was confusing the real-world concept of a property (in "prove any kind of property") with the mathematical concept of a property as a set of behaviors (in "any property is the conjunction ...").

It can be argued that (ii) is an unavoidable source of misunderstanding, since one can discuss physical reality only in terms of mathematical models. I don't think the situation is so hopeless. We can make statements about the physical world like "if you press this key, then the system crashes" that mean approximately the same thing to everyone, regardless of his philosophical persuasion.

I think that Vaughan's question 3 (sine waves) is an example of (i) and his question 2 (teams of observers) is an example of (ii). His question 1 (race conditions) is more interesting and warrants discussion.

A race condition is bad if it makes the circuit behave incorrectly. When verifying circuits, one is interested only in proving that a circuit behaves correctly, not that it behaves incorrectly. So, one never has to prove the existence of a race condition. The specification of the circuit describes its external behavior, and a race condition is something that happens inside the circuit. So, proving the absence of a race condition is never a primary goal. If there is a potential race condition that never actually occurs--for instance, because of the initial conditions--then the proof will contain a lemma (a mathematical formula) whose physical interpretation will be the absence of a race condition.

However, the concept of a race condition is not irrelevant. A race condition on its inputs might cause a circuit component to produce an invalid output voltage--a "1/2" instead of a "0" or a "1". In this case, a mathematical model of the component that allows only the outputs "0" and "1" is inadequate. With such a model, the domain of applicability of the theory would not include the actual circuit. Fortunately, with more sophisticated models (for example, by including a "1/2" output), I believe it is possible to use my theory to reason about real circuits. (I haven't done such reasoning myself, but others have using similar theories.) The concept of a race condition is relevant for modeling the real circuit in the mathematical formalism, but it doesn't appear in the formalism itself.

Scientific theories are useful because the mathematical formalism is simpler than physical reality. Newtonian physics eliminates an awful lot of important details--like you and me--when it represents the earth as a point mass. Those details are irrelevant for computing planetary orbits. They are not irrelevant for studying human history. Science is the art of simplification.

> A theory should be as simple as possible, but no simpler.
> - Albert Einstein

The test of a scientific theory is how well it helps us understand and/or manipulate the real world.

I will close with a word about mathematics. Many computer scientists aren't scientists at all; they're mathematicians. They work in the domain of mathematical formalism, with no concern for its application to the real world. That's fine. The world needs pure mathematicians as well as scientists. But it's important for mathematicians to realize that they're not scientists. Number theorists don't criticize Newtonian mechanics for using real numbers rather than integers. Computer-scientist/mathematicians should be equally sensible.

From sf@csli.stanford.edu Wed May 29 16:40:07 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Fri, 16 Nov 90 22:13:04 EST
To: concurrency
Subject: [sf@csli.stanford.edu: Re: Reply to Pratt]
From: Sol Feferman <sf@csli.stanford.edu>
Sender: meyer@theory.lcs.mit.edu
In-Reply-To: Your message of Fri, 16 Nov 90 18:28:10 EST

Date: Fri, 16 Nov 1990 17:40:07 PST
To: meyer@theory.lcs.mit.edu

Bravo! (I don't know anything about any of the theories involved but I agree completely with your general comments.)

From lynch@holmes.lcs.mit.edu Wed May 29 16:40:08 1991
Return-Path: <meyer@theory.lcs.mit.edu>

Date: Sun, 18 Nov 90 17:47:34 EST
To: concurrency
From: lynch@holmes.lcs.mit.edu (Nancy A. Lynch)
Sender: meyer@theory.lcs.mit.edu
Subject: On Lamport and Milner

Date: Sat, 17 Nov 90 07:03:36 EST

I have been following the debate about trace models with interest, and
liked Leslie Lamport's most recent comments.
They do seem to get at the heart of the differences between the different
research communities.

One of the most interesting (and troubling) comments he makes is the remark
about CCS not being useful for verifying distributed algorithms of any
complexity; supposedly, Robin Milner agrees with this (!).
Now, I thought I understood that a major goal of process algebraic research
WAS to verify complex concurrent and distributed algorithms.
I would like to hear more about this issue from proponents of CCS-like
methods. More specifically, can anyone tell me clearly what
types of algorithms such methods are suited for verifying, and what are
outside their domain of applicability?
If the methods so far have really had only limited success, then
is this limitation inherent in the methods
(or their intended domain of applicability)
or just a matter of time?

Nancy Lynch
lynch@theory.lcs.mit.edu

From pratt@cs.stanford.edu Wed May 29 16:40:09 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Sun, 18 Nov 90 17:52:16 EST
To: concurrency
Subject: Reply to Lamport's reply to Pratt
From: pratt@cs.stanford.edu
Sender: meyer@theory.lcs.mit.edu
In-Reply-To: Your message of Fri, 16 Nov 90 18:28:10 EST.
             <9011162328.AA05325@stork>

Date: 18 Nov 90 00:04:52 PST (Sun)

[The story so far.] On Oct. 21 David Luckham queried me about an
attitude to partial orders that he'd run into during discussions with
ONR-funded software people. I shared my reply to David with this list,
which led to considerable discussion. On Nov. 6 Leslie Lamport entered
the discussion with a complaint that certain parties to this discussion
whom he did not name were claiming that he couldn't do what he was
doing, an assertion that he could indeed do what he was doing, and a
deduction that those parties must therefore be charlatans.

I pleaded innocent to the complaint, agreed with the assertion, and, in
case Leslie had me in mind as one of the charged parties, attempted to
refute the deduction with some situations where partial orders helped.

Leslie's reply of yesterday (Nov. 16) put my situations into three classes: those outside his world, e.g. sine waves, those in his world but independent of his theory of his world, e.g. multiple observers, and those that potentially conflicted with his theory but which he felt confident his theory could be extended gracefully to handle, e.g. race conditions. He concluded by chastising mathematicians who criticize what scientists do. [Now read on.]

This conclusion leaves me puzzled. While Leslie has defended himself admirably, I cannot tell what criticism stung him into defense. Let me repeat what I said on Nov. 12:

> There have been various claims on this list about limitations of interleaving, but none that I recall making the claims Leslie was complaining about, nor any that conflicted with the evidence he adduced in support of his complaint.

Leslie's techniques seem to be fine for their purposes. I don't know why this message isn't getting through.

Echoing Sol Feferman's "Bravo," I heartily concur with the rest of Leslie's stimulating essay, to within the following differences.

> The answer is that I don't know and I don't care. These questions never arise in my work.

I know that and I didn't care at first. Robert Hall supplied the necessary existence proof that there were people on the list who did care, or I would have let the matter rest with just the Nemeti quote >from LOP-81 (LNCS 131, p.419), my initial response to Leslie's opening message.

Although Leslie's view of concurrency is adequate for him, it is also somewhat of a straitjacket. There are aspects of concurrency that he does not find worth studying but that others do. Perhaps the implications of those aspects will never insinuate themselves into Leslie's world, but who knows? Which residents of Nagasaki and Hiroshima foresaw the abrupt intrusion of the abstract equation E=mc^2 into their world?

> Fortunately, with more sophisticated models (for example, by including a "1/2" output), I believe it is possible to use my theory to reason about real circuits.

Yes, this is an excellent idea. Its origins are surely shrouded in history, but it can be found recently in van Glabbeek and Vaandrager's PARLE-87 notion of ST-bisimulation, with Leslie's 1/2 represented as marked transitions. It is also the basis for the "prosset" model Gaifman and I described in LICS-87, a model described more elegantly in "Temporal Structures" (in LNCS 389 21-51, also STAN-CS-89-1297, also available by ftp from boole.stanford.edu as man.{tex,dvi}, and to appear in Math. Struct. in CS 1:2), in terms of the "idempotent closed ordinal" 3'. In Leslie's notation 3' = {0,1/2,1}. This important (non-cartesian-closed) ordinal is also the dualizing object 3 in the Stone-Birkhoff duality described in my POPL-91 paper, though space and time have conspired to let me do little more than name 3 in that paper; a proper account of the dualizing role of 3 will appear in a subsequent

paper. The essential idea is that {0,1/2,1}, or {0,T,1} as I call it in the POPL paper, refer respectively to before, transition, and after. A race is characterized by the possibility of having two processes both being in state T. The function of mutual exclusion is to rule out that combination. This is the essential distinction between a|b and ab+ab: both permit 8 of the 9=3^2 combinations in (0,T,1)x(0,T,1), but only the former permits the 9th combination (T,T),

I apologize for the large amount of algebraic machinery in which we have embedded Leslie's 1/2 in some of this work, like Sigourney Weaver in her exoskeleton in Aliens. Those wishing to meet 1/2 in a more comfortable outfit will have to await our return to planet Earth, hopefully soon. Meanwhile let me assure you that this unnerving exoskeleton really does amplify power just like the ads promise. I had no idea by how much until my students started using it on big jobs.

> CCS works fine for verifying biscuit machines. It is hopelessly impractical for verifying even the simplest distributed spanning tree algorithm, let alone the more complex algorithms that system builders use. Robin Milner realizes this (I've discussed it with him), but many of his disciples don't.

You could get both Robin and me to agree to this, much as perhaps Robin and certainly I would agree that the axiomatic theory of vector spaces is fine for treating sums and scalar multiples of vectors, but is hopelessly impractical for inverting even the most well-conditioned matrices, let alone the ill-conditioned matrices that arise in transcontinental surveys. Surveyors just want their programs to give the right results, their passion for the axiomatic theory of vector spaces rarely exceeds that of Leslie's for CCS.

> But it's important for mathematicians to realize that they're not scientists.

This is indeed the popular, standard, and authorized view. Nicolas Goodman makes a strong argument for the opposing view in a recent article entitled "Mathematics as Natural Science," JSL 55(1)182-193 (March 1990).

My own view (I do hope no one is actually paying to receive this stuff:-) strays even further from the standard than Goodman's. I think of us as dealing with incoming data from the world mainly by inventing theories through which this data is filtered to yield predictions about the world; that, mutatis mutandis (important), natural selection selects for those theories whose predictions are more accurate; and (the most controversial bit) that the theories most successful at predicting are sufficiently like the most successful theories of pure mathematics that the latter should prove to have good survival value while the former could with little violence be turned into respectable mathematics. The controversial bit has the merit that both directions are in principle testable given suitable advances in AI and brain mapping respectively.

> A theory-of-everything is generally good for nothing... For computer science, the real world usually consists of computers (hunks of wire and silicon) executing programs.

It has not escaped the attention of some contributors to concurrency theory that it is starting to look like a "theory of everything." This is the result of abstracting away wire, silicon, and programs to leave a set of abstractions that could as readily be applied to the interactions of galaxies of stars, swarms of bees, and rioting soccer fans as to processes communicating via ethernets, IP/TCP, and remote procedure calls.

However concurrency theory is only a "theory of everything" in the same sense that number theory and group theory are "theories of everything." Just as number theory is more than the theory of counting sheep and beans, and group theory more than a means of proving that quintics don't have solutions expressible in radicals, so is concurrency theory more than the theory of what concurrent "hunks of wire and silicon" do.

There are then two roads one may follow here, the conservative and the liberal. The conservative road requires keeping wire and silicon in mind as the ultimate domain of application of concurrency research. The liberal road replaces "computer science" by "information science" and seeks instead a theory of information processing that will turn out to be applicable to information processors in general, whether dumb like galaxies, smart like bees and computers, or brilliant like us (pats all round).

I am most interested in the liberal road because it seems to me that the techniques of both computer science and engineering, provided they are not artificially constrained, should turn out to be broadly applicable.

For example today's factory designers have only relatively primitive tools to help them develop a design on line, test it out to get a better feeling for how well it might work in practice, turn it into a detailed blueprint for a factory, and make it the basis both for the ongoing operation and maintenance of the factory and for future modifications and redesigns.

The analog of this scenario for software systems is much further along, though it too has far to go or software research would have nothing left to do. There is no reason why the foundations of the latter should not also prove to be equally useful foundations for the former. If this is the case then the taxpayers' research dollars are spent more efficiently by working out concurrency theory so as to fully realize its benefits in all domains to which it is applicable.

I want very badly to follow the liberal road. My big problem has always been that I don't know how to write a good program until I understand the theory of what that program is about. Hence my current preoccupation with theory. This is now well along however, and I hope to be able to start designing and implementing soon. I'm hoping that many of Leslie's excellent ideas will prove useful in aspects of this work.

Vaughan Pratt

From luca@cogs.sussex.ac.uk Wed May 29 16:40:09 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Mon, 19 Nov 90 12:07:45 EST
To: concurrency
Subject: Two papers on begin-end
From: Luca Aceto <luca@cogs.sussex.ac.uk>
Sender: meyer@theory.lcs.mit.edu

Date: Mon, 19 Nov 90 14:20:31 GMT

In the debate on "True Concurrency vs. Interleaving" on the concurrency mailing list some of the recent messages have been concerned with the modelling of the behaviour of concurrent systems under the assumption that actions have a beginning and an ending. We have been working on semantic theories for process algebras based on variations on the above idea and our results are reported in a series of papers, which are available to whoever requests them.

L Aceto, M Hennessy

Towards Action Refinement in Process Algebras

Luca Aceto and Matthew Hennessy

ABSTRACT

We present a simple process algebra which supports a form of refinement of an action by a process and address the question of an appropriate equivalence relation for it. The main result of the paper is that an adequate equivalence can be defined in a very intuitive manner. In fact we show that it coincides with the "timed-equivalence" proposed by one of the authors in [H88]. This is a bisimulation-like equivalence based upon the idea of splitting every action into two sub-actions, the beginning and the end. For the language which we consider this equivalence also coincides with a variation, called "refine eqivalence", in which the beginings and endings of actions with the same name must be properly matched.

Reference: [H88] M. Hennessy, Axiomatizing Finite Concurrent Processes, SIAM Journal on Computing 17(5), pp. 997-1017, 1988.

Adding Action Refinement to a Finite Process Algebra

Luca Aceto and Matthew Hennessy

ABSTRACT

In this paper we present a process algebra for the specification of concurrent, communicating processes which incorporates operators for the refinement of actions by processes, in addition to the usual operators for communication, nondeterminism, internal actions and restrictions, and study a suitable notion of semantic equivalence for it. We argue that action-refinements should, in some formal sense, preserve the synchronization structure of processes and their application to processes should consider the restriction operator as a "binder". We show that, under the above assumptions, the weak version of the refine equivalence introduced in [AH89] is preserved by action refinement and, moreover, is the largest such equivalence relation contained in weak bisimulation equivalence. We also discuss an example showing that, contrary to what happens in [AH89], refine equivalence and timed equivalence are different notions of equivalence over the language considered in this paper.

Reference: [AH89] This is the paper mentioned above.

From rounds@caen.engin.umich.edu Wed May 29 16:40:10 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Mon, 19 Nov 90 12:12:42 EST
To: concurrency
Subject: can't resist a comment
From: rounds@caen.engin.umich.edu (Prof Rounds)
Sender: meyer@theory.lcs.mit.edu

Date: Mon, 19 Nov 90 12:09:21 EST

I'd like to throw two cents' worth into
what seems to be one of the best ``bulletin board''
discussions I've seen in a long time.

I agree with both Leslie Lamport and Vaughan Pratt.
A mathematical model is always just that; it represents
our cognitive abstraction of what reality we perceive.
The theorems true in the model make predictions, which
we then reinterpret in the real world, at least that
part of the world which interests us.
The best models simplify and constrain reality enough
so that they make really strong predictions
(I would put the finite-state machine in that category.)
Of course, in a particular domain, the model may not
account for observed phenomena, and may in fact be
contradicted. If one wants to predict these new
phenomena, one must refine the mathematical model.

This process, though painful for those who believe in the
old model, is at the heart of scientific progress.

The preceding paragraph talked about science; there is
another point to make about engineering. In the
field of computers we have the unprecedented opportunity
to create real-world systems which conform to our
mathematical perceptions. So, machines were designed
to mirror our conception of digital computation;
programming languages help us express mathematical algorithms,
and so forth. The fascinating thing about concurrency theory
is that it seems to be on the fence between science and
engineering. We can use it to ``explain'' race conditions,
or we can use it to help us design programs (witness
CSP, occam, and the transputer.) Of course this was
true about computability theory itself in the 30's and 40's.
Witness the creation of the stored-program machine to
embody the Universal Turing machine.

One other nice thing about mathematical models is that
they port themselves into other domains of applicability.
About 4 years ago I was working with a graduate student,
Bob Kasper, on some problems in natural language processing.
The problem involved specifying disjunctive information
in record-like structures -- more or less like variant record
types are specified in Pascal. We saw a simple way to
understand and to implement a system, using extremely
basic notions from concurrency theory. Essentially one
views a complex record as a transition system. The states
are the individual nodes, and the transitions are the field designators.
Then the simple logic of Hennessy and Milner, or the simplest
possible subcase of deterministic PDL, becomes a way
of declaring record types. Once this is seen, there are
a lot of ways to reinterpret the concepts of concurrency
in data types. I've been using the notions of Smyth and Hoare powerdomains,
along with Aczel's non-wellfounded set theory,
for example, to help understand and design so-called complex
objects in object-oriented databases. Notice
that Aczel's work came from an attempt to provide a proper
mathematical foundation for SCCS!

The point of this last experience is that one
should always keep an open mind, especially where mathematical
models are concerned.

Bill Rounds

From hg17@cunixd.cc.columbia.edu Wed May 29 16:40:11 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Tue, 20 Nov 90 09:34:00 EST
To: concurrency
Subject: Lamport on Spinoza, Science and related matters
From: Haim Gaifman <hg17@cunixd.cc.columbia.edu>
Sender: meyer@theory.lcs.mit.edu

Date: Mon, 19 Nov 90 19:39:10 EST

This is rather a belated reaction to some of the claims made in the exchange that has started with Leslie Lamport's message of November 7 ("Flame etc.") While Lamport's observations concerning Aristotles, Kant and Spinoza are marginal to the real issues of the debate, at least one point needs correction:

"...." and Spinoza proved that there can be at most seven planets."

As a matter of fact, Spinoza never "proved" that there can be at most seven planets. Lamport is probably confusing Spinoza with Hegel (who lived two centuries later). Somewhere in Hegel's dissertation, so the story goes, is buried an argument purporting to show that the number of planets should be
seven.

Perhaps the difference between Spinoza and Hegel does not mean much to Lamport.
After all, they were both philosphers, that is to say vaporizing theoreticians
making ridiciluously unfounded claims. But, as a scientist, he should have gotten his facts straight.

As to the debate itself:

If A claims to have done something that B has proved to be impossible, then either
(i) there is an errors in A's construction,
or
(ii) there is an error in B's proof,
or
(iii) they are speaking about different things.

In cases (i) and (ii) the debate can be clearly decided; the errors are found, one of the claims (perhaps both) is withdrawn and there the matter ends.
But this happy state of affairs is mostly a privelege of mathematicians. In philosophy it is usually the third case that obtains. When things get clarified, it turns out that the real issue is not the correctness of a certain proof, but the correct way of defining certain notions, or of setting up a framework. The debate is about which setup is more intuitive, illuminating, fruitful, efficient, etc.

It appears that, in this respect, many computer scientists share the fate of
philosophers. What has statrted as a claim for a contradiction ("I have done something that somebody proved cannot be done") turns out to be a claim about the relative merits of trace models versus partial order models.

Lamport is certainly entitled to the view that the methods developed by him are
simpler and more efficient, for the purposes of analysing and proving correctness of distributed algorithms. No doubt, he can produce his own impressive work as an argument for this view. The claim could be evaluated (certainly not by me!) in a matter of fact way. This does not guarantee that

the question would be settled, but at least we would have a clearer view of what is involved. Unfortunately, he has got this bad habit of philosophers to start with an imprecise presentation of the problem.

Another bad influence of popularized philosophy is the temptation to anchor one's views, no matter what the subject is, in some major principles; in the present case maxims about what is and what is not good science are mobilized for the sake of the argument:

> "Any useful scientific theory has a limited domain of application. A theory-of-everything is generally good for nothing."

In one sense, this is a sound rule of thumb that one would hardly wish to quarrel with: The more phenomena you try to accomodate the more likely you are
to get an impractical system. The rule has, nonetheless, some spectacular exceptions. A higher level description that encompasses a wider range of phenomena might be more efficient then a narrower view. Every mathematician knows cases in which generalizing (hence strengthening) a theorem leads to a conceptually clearer, hence easier, proof of it. From an Arsitotelian point of
view Newtonian physics would have been a project unlikely to succeed, because it tried to account for the immense variegated domain of movement phenomena by
few simple laws.

As a general *prescription* for science, the above quote goes certainly against the grain that is exemplified by great scientists, such as Newton, Maxwell or Einstein. A "theory-of-everything" is the elusive goal that has motivated big scientific enterprises. What else is the point of the reduction of chemistry to physics, or of finding a unified field theory?

All this has no direct bearing on whether an interleaving model, or a partial order model, or some other abstract model, is more suitable for reasoning about concurrent processes. But in trying to drag in general philosophical principles, Leslie Lamport seems to have committed himself to quite a narrow perspective of science, it is rather an engineer's view than anything else.

Haim Gaifman

From pratt@cs.stanford.edu Wed May 29 16:40:11 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Sun, 25 Nov 90 17:25:09 EST
To: concurrency
Subject: Early pomset paper
From: Vaughan Pratt <pratt@cs.stanford.edu>
Sender: meyer@theory.lcs.mit.edu

Date: Sun, 25 Nov 90 12:25:32 PST

If there are any historians of concurrency theory subscribing to this forum they might be interested in the origins (as I understand them) of the term "pomset."

The terms "labeled partial order" and "partial word" had been used previously, but the earliest paper I'm aware of that refers explicitly to partially ordered multisets as a synonym for these notions is:

```
        @InProceedings(
Pr82,   Author="Pratt, V.R.",
        Title="On the Composition of Processes",
        Booktitle="Proceedings of the Ninth Annual ACM Symposium on
                Principles of Programming Languages",
        Month=Jan, Year=1982)
```

However I had not at that time come up with the contraction "pomset." This term was first advertised in a talk I gave on Sept. 13, 1983 at a workshop whose proceedings however were not published until 1985:

```
        @InProceedings(
Pr83,   Author="Pratt, V.R.",
        Title="Two-Way Channel with Disconnect",
        Booktitle="The Analysis of Concurrent Systems:
                Proceedings of a Tutorial and Workshop, LNCS 207",
        Publisher="Springer-Verlag", Year=1985)
```

I also used it in a talk I gave the following week at IFIP-83 in Paris. It appears in the position statement I circulated at that panel, a hundred or so copies of which were distributed to the audience:

```
        @Unpublished(
Pr83b,  Author="Pratt, V.R.",
        Title="Position Statement",
        Note="Circulated at the Panel on Mathematics of Parallel
        Processes, chair A.R.G. Milner, IFIP-83",
        Month=Sep, Year=1983)
```

Now that I look at it again it seems to me that this position statement is quite clear about my motivation in those days for pomsets and how I thought they should be used. Since it's reasonably short and can't be found elsewhere I've appended it below. (My apologies for it's being in Scribe, this was what many of us at MIT and Stanford used back then. Just read the raw Scribe, the only obscurity should be x@-[y], the Scribe for x subscript y.)

The cryptic allusion therein to ab|ab and N(a,a,b,b) refers to the fact, found by my student Jay Gischer, that these two pomsets are language-equivalent. That is, regarded as language operations applied to languages a and b under the evident interpretation, they denote the same language. In 1982 Jay independently came up with the partially ordered multiset concept, though not by that name, while investigating the problem of completely axiomatizing the equational theory of concatenation and shuffle of languages which I had posed to him. Jay reduced my axiomatization problem to the question of whether for any two N-free pomsets, language-equivalence implied isomorphism. I was quite surprised to find the partially ordered multisets of my POPL-82 paper arising so naturally in connection with this question about pure interleaving semantics. Neither Jay nor I found an answer to this question, which I publicized (as an axiomatization question) on various occasions during 1986-1988. It was eventually solved in 1988 by Steve

Tschantz, an algebraist at Vanderbilt, in

```
        @Unpublished(
Tsch,   Author="Tschantz, S.T.",
        Title="Languages under concatenation and shuffling (preliminary)",
        Note="Manuscript, Department of Mathematics, Vanderbilt University",
        Month=Jun, Year=1988)
```

Steve independently discovered the same reduction of the axiomatization problem to the question about language-equivalence of N-free pomsets, which he answered affirmatively by an ingenious argument. Luca Aceto subsequently applied Tschantz's theorem to infer the surprising result [correspondence, Apr. 1989] that timed-equivalence coincides with trace-equivalence for the language p::= 0 | a | p;p | p|p.

Since 1983, starting with my LOP-85 paper

```
        @InProceedings(
Pr85,   Author="Pratt, V.R.",
        Title="Some Constructions for Order-Theoretic Models of Concurrency",
        Booktitle="Proc. Conf. on Logics of Programs, LNCS 193",
        Address="Brooklyn", Publisher="Springer-Verlag", Pages="269-283",
        Year=1985),
```

which turned into

```
        @Article(
Pr86,   Author="Pratt, V.R.",
        Title="Modeling Concurrency with Partial Orders",
        Journal="International Journal of Parallel Programming",
        Volume=15, Number=1, Pages="33-71", Month=Feb, Year=1986),
```

my thoughts on the appropriate combinators for pomsets have shifted >from the network emphasis in my POPL-82 paper and IFIP-83 statement to a more arithmetic kind of language in which pomsets are added and multiplied (and these days exponentiated, whose relevance to concurrency I did not appreciate in 1985). Nowadays, at my student Roger Crew's prodding, I regard network combination as merely one of several variants of addition.

```
        Vaughan Pratt
        Nov. 25, 1990
```

==========APPENDIX---IFIP-83 STATEMENT==========

```
@libraryfile(specialcharacters)
        @b[@center(IFIP-83 - Panel on Mathematics of Parallel Processes)
                        @center(Position Statement)
                            @center(V. R. Pratt)
                        @center(Stanford University)]
                        @center(September, 1983)
```

@b[Abstract]. The notion of function as a set of ordered pairs is mathematically appealing but not quite rich enough for modelling processes. Our position is that it suffices to generalize ordered pairs to pomsets

(partially ordered multisets) to obtain a satisfactory notion of process.

@b[Functions]. A function abstracts the essence of stimulus-response: it collects all possible stimuli and pairs each with a corresponding response. Furthermore functions obey the principle of behavioral extensionality: two functions with the same set of stimulus-response pairs are considered not merely behaviorally equivalent functions but in fact the same function. These two attributes are captured simultaneously in defining a function from A to B to be a subset of AxB (with additional conditions when being single-valued and total matters).

@b[Processes]. Processes are like functions in some respects. Processes accept stimuli and emit responses. And behavioral extensionality is just as natural for processes as for functions.

A process is not however an ordinary function. It may for example respond to each of a series of numeric inputs with the sum of all inputs to date; this is the behavior of a cumulative "function," which is not really a function since it takes memory to keep a running sum.

@b[Functions on Histories]. A process can be made a function if the domain is taken to be sequences of stimuli instead of individual stimuli. That is, a process may be defined to be a function from histories. It is natural to then take the codomain to be histories as well, i.e. a process is a function on histories.

This definition is the basis for the semantics of parallel processes given at IFIP 74 by G. Kahn [K], and elaborated on at IFIP 77 by Kahn and D. MacQueen [KM]. This definition works well for deterministic processes.

@b[The Nondeterminism Anomaly]. In 1978 D. Brock and W. Ackerman exhibited an anomaly demonstrating that the straightforward extension of Kahn-MacQueen semantics to nondeterministic processes, namely relations on histories, did not yield sensible behaviors [BA]. They identified the problem as a lack of information about the relative timing of individual input and output events. The Kahn-MacQueen model did not specify any interleaving information between input and output histories. Brock and Ackerman noted that a little additional information of this sort sufficed to dispose of the anomaly at hand.

@b[Our Position.] We consider the Brock-Ackerman fix, appropriately formalized [Pr], to provide a very attractive model of processes. Before defining this model we introduce the notion of partially ordered multiset or pomset.

@b[Pomsets]. A @i[pomset] on a set A is, up to isomorphism, a structure (U,L,<) consisting of an underlying set U, a labelling function L:U@k(rightarrow)A, and a partial order < on U.

The labels supply the elements of the pomset. The same label can be reused, hence multiset rather than set. Pomsets are defined only up to isomorphism (of structures) because the identity of the underlying set is unimportant; only the labels (the @i[real] multiset elements) and the order matter.

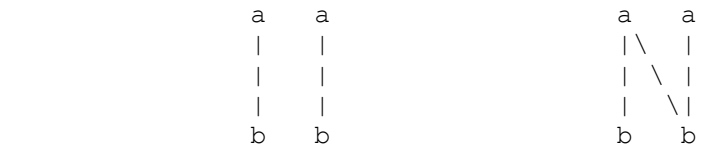
@b[Main definition.] @i[A process on a set E is a set of pomsets on E.]

@b[Intended Interpretation]. E is a set of events. Each pomset of events is one of the possible computations of the process. The order on each pomset is that of necessary temporal precedence; the order of the events in a computation need not be completely specified.

@b[Contrast with Functions]. A function is a set of totally ordered doubletons. This definition exposes three differences between functions and processes: the dropping of the cardinality requirement that each element of a function have two elements, the switch from sets to multisets, and the switch from a total order to a partial order.

The cardinality change is motivated by the ongoing nature of a process: many events may need to be considered as part of a single computation. Multisets are needed because an event may be repeated, e.g. the arrival of the number 3.
Partial orders are preferred over total because it is not always natural to totally order events - consider for example two communicating processes on Earth and Saturn respectively, each running at nanosecond speeds.

@b[Inadequacy of Total Orders]. The use of total rather than partial orders enjoys some currency in modelling parallel processes [H][Pn]. However there does not appear to be a natural way of using total orders to distinguish the following two ways in which two a's might precede two b's.
@begin(verbatim)



@end(verbatim)

Thus not only are total orders unnatural, they are not an expressively adequate substitute for pomsets.

@b[Examples]. The above-drawn pomsets together form a two-element process. Any n-ary relation (hence binary relation, and hence function) is a process if each n-tuple in the relation is regarded as a totally ordered set. A power set is a process if each element is regarded as a set with the empty partial order. The power set C of a power set B is a process if each element of C is regarded as ordered by inclusion on B: event e necessarily follows event d just when e is d with some additional elements - the process makes progress by accumulating elements and distinct accumulations leading to the same subset are (in this case) considered the same event.

@b[Spatial Localization]. In order to put processes in communication with each other it is helpful to know where their events are taking place (cf. [W], p.64). We define an @i[event space] to be a Cartesian product CxD, consisting of @i[spatial events]. The intended interpretation is that C is a set of @i[channels] or @i[places] (cf. [B]) where the events may be found and D the set of @i[data] that may be sent over the channels of C. A @i[spatial process] is a process on an event space.

@b[Nets.] A @i[net] is a process P on CxD having constituent processes P@-[1],...,P@-[n] on C@-[1]xD,...,C@-[n]xD respectively. Process

P@-[i] is a @i[constituent] of P just when there exists a function a@-[i]:C@-[i]@k(rightarrow)C determining a projection A@-[i]:P@k(rightarrow)P@-[i]. (a@-[i] gives the attachment of the channels (i.e. ports) of P@-[i] to the channels of the net.) The projection A@-[i] is determined from a@-[i] by taking A@-[i](p) to be the multiset {(c,d)|(a@-[i](c),d) is in p}. Order is preserved, that is, (c,d) < (c',d') in A@-[i](p) iff (a@-[i](c),d) < (a@-[i](c'),d') in p. (Note that A@-[i] need not be onto, i.e. it is not required that P@-[i] equal A@-[i](P), only that it include it.)

@b[Process Composition.] Processes are composed to form a new process in two steps: given the processes P@-[i] with corresponding attachments a@-[i]:C@-[i]@k(rightarrow)C for i from 1 to n-1, the maximum (under set inclusion) net P having those processes as constituents is formed, and then an additional attachment a@-[n]:C@-[n]@k(rightarrow)C is used to determine the projection A@-[n]:P@k(rightarrow)P@-[n]. The result is A@-[n](P). The n attachments themselves can thus be seen to determine an (n-1)-ary operation on processes.

@b[Example.] Ordinary composition of binary relations on D is determined by C@-[1] = C@-[2] = C@-[3] = {0,1}, C = {0,1,2} with a@-[1](c) = c, a@-[2](c) = c+1, and a@-[3](c) = 2c. In this net P@-[1] and P@-[2] are composed to yield P@-[3]. This is of course a particularly simple example.

@b[Bibliography]

[B] Brauer, W., Net Theory and Applications, Springer-Verlag LNCS 84, 1980.

[BA] Brock, J.D. and W.B. Ackerman, Scenarios: A Model of Non-Determinate Computation. In LNCS 107: Formalization of Programming Concepts, J. Diaz and I. Ramos, Eds., Springer-Verlag, New York, 1981, 252-259.

[H] Hoare, C.A.R., Communicating Sequential Processes, CACM, 21, 8, 666-672, August, 1978,

[K] Kahn, G., The Semantics of a Simple Language for Parallel Programming, IFIP 74, North-Holland, Amsterdam, 1974.

[KM] Kahn, G. and D.B. MacQueen, Coroutines and Networks of Parallel Processes, IFIP 77, 993-998, North-Holland, Amsterdam, 1977.

[M] Milner, R., A Calculus of Communicating Systems, Springer-Verlag LNCS 92, 1980.

[Pn] Pnueli, A., The Temporal Logic of Programs, 18th IEEE Symposium on Foundations of Computer Science, 46-57. Oct. 1977.

[Pr] Pratt, V.R., On the Composition of Processes, Proceedings of the Ninth Annual ACM Symposium on Principles of Programming Languages, Jan. 1982.

[W] Winskel, G., Events in Computation, Ph.D. Th., Dept. Comp. Sci, U. of Edinburgh, Dec. 1980.

From heddaya@cs.bu.edu Wed May 29 16:40:12 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Mon, 26 Nov 90 18:10:59 EST
To: concurrency
Subject: Early pomset paper
From: heddaya@cs.bu.edu
Sender: meyer@theory.lcs.mit.edu
In-Reply-To: Your message of Sun, 25 Nov 90 17:25:09 -0500.
             <9011252225.AA00710@stork>

Date: Mon, 26 Nov 90 12:12:26 -0500

Here's a LaTeX version of Pratt's IFIP-83 workshop position paper that he recently posted to the concurrency mailing list. I did not edit it in any way, but I must take responsibility for any notational erros I may have introduced in the twenty minutes I spent converting it from Scribe format.

```
%================================Cut here================================
% Template for short articles in LaTeX.
%
% Author:   Abdelsalam Heddaya, Boston University
% Created:  1988 at Harvard
% Modified: 1989.07 at BU
% Modified: 1990.10.23 at BU
%=========================================================================
=

\documentstyle[11pt]{article}
\pagestyle{plain}

% ====================================================
% Layout dimensions
% ====================================================

% NOTE: All the variables listed below have default values that depend
% on the document style, so, it is wise to leave as many of them as
% possible untouched.

% ---------------------
% Page layout:
% ---------------------

\textheight 9.0in             % Total height including Head
\textwidth 6.5in              % Total width including (probably) Margin Notes
\topmargin 0.0in              % Point of origin is (1in,1in) from
                              % upper left corner
\oddsidemargin 0.0in          % Left side of odd pages (or all in
                              % case of one sided printing)
\evensidemargin 0.0in         % Left side of even pages (in case of
                              % two sided printing)
\headheight 0.0in
\headsep 0.0in                % Between Head and top of text

%\footheight 0.0in
%\footskip 0.0in              % Between bottom of text and Foot (not
                              % footnotes)
% ---------------------
```

```
% Paragraph layout:
% ----------------------

\parskip 0.5 \baselineskip     % Inter-paragraph space
\parindent 1.3em               % Paragraph indentation

%
==========================================================================
=

% Custom settings of dimensions and other variables go here.
% Check the preamble file for comments on the layout variables.

\parindent     0.0in

\begin{document}

\begin{centering}

{\Large IFIP-83 - Panel on Mathematics of Parallel Processes \\
Position Statement
}\\

\medskip

V.R. Pratt\\
Stanford University

\medskip

{\it September, 1983}

\bigskip

\end{centering}


{\bf Abstract}.  The notion of function as a set of ordered pairs is
mathematically appealing but not quite rich enough for modelling processes.
Our position is that it suffices to generalize ordered pairs to pomsets
(partially ordered multisets) to obtain a satisfactory notion of process.

{\bf Functions}.  A function abstracts the essence of stimulus-response:
it collects all possible stimuli and pairs each with a corresponding
response.  Furthermore functions obey the principle of behavioral
extensionality:  two functions with the same set of stimulus-response
pairs are considered not merely behaviorally equivalent functions but
in fact the same function.  These two attributes are captured
simultaneously in defining a function from $A$ to $B$ to be a subset
of $A \times B$
(with additional conditions when being single-valued and total
matters).

{\bf Processes}.  Processes are like functions in some respects.
Processes accept stimuli and emit responses.  And behavioral extensionality
is just as natural for processes as for functions.
```

A process is not however an ordinary function. It may for example respond to each of a series of numeric inputs with the sum of all inputs to date; this is the behavior of a cumulative ``function,'' which is not really a function since it takes memory to keep a running sum.

{\bf Functions on Histories}. A process can be made a function if the domain is taken to be sequences of stimuli instead of individual stimuli. That is, a process may be defined to be a function from histories. It is natural to then take the codomain to be histories as well, i.e. a process is a function on histories.

This definition is the basis for the semantics of parallel processes given at IFIP 74 by G. Kahn [K], and elaborated on at IFIP 77 by Kahn and D. MacQueen [KM]. This definition works well for deterministic processes.

{\bf The Nondeterminism Anomaly}. In 1978 D. Brock and W. Ackerman exhibited an anomaly demonstrating that the straightforward extension of Kahn-MacQueen semantics to nondeterministic processes, namely relations on histories, did not yield sensible behaviors [BA]. They identified the problem as a lack of information about the relative timing of individual input and output events. The Kahn-MacQueen model did not specify any interleaving information between input and output histories. Brock and Ackerman noted that a little additional information of this sort sufficed to dispose of the anomaly at hand.

{\bf Our Position.} We consider the Brock-Ackerman fix, appropriately formalized [Pr], to provide a very attractive model of processes. Before defining this model we introduce the notion of partially ordered multiset or pomset.

{\bf Pomsets}. A {\em pomset} on a set A is, up to isomorphism, a structure $(U,L,<)$ consisting of an underlying set U, a labelling function $L: U \rightarrow A$, and a partial order $<$ on U.

The labels supply the elements of the pomset. The same label can be reused, hence multiset rather than set. Pomsets are defined only up to isomorphism (of structures) because the identity of the underlying set is unimportant; only the labels (the {\em real} multiset elements) and the order matter.

{\bf Main definition.} {\em A process on a set E is a set of pomsets on E.}

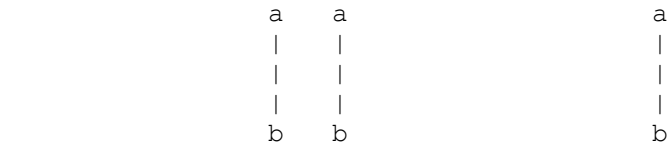
{\bf Intended Interpretation}. E is a set of events. Each pomset of events is one of the possible computations of the process. The order on each pomset is that of necessary temporal precedence; the order of the events in a computation need not be completely specified.

{\bf Contrast with Functions}. A function is a set of totally ordered doubletons. This definition exposes three differences between functions and processes: the dropping of the cardinality requirement that each element of a function have two elements, the switch from sets to multisets, and the switch from a total order to a partial order.

The cardinality change is motivated by the ongoing nature of a process: many events may need to be considered as part of a single computation. Multisets are needed because an event may be repeated, e.g. the arrival of the number 3.

Partial orders are preferred over total because it is not always natural to totally order events - consider for example two communicating processes on Earth and Saturn respectively, each running at nanosecond speeds.

{\bf Inadequacy of Total Orders}. The use of total rather than partial orders enjoys some currency in modelling parallel processes [H][Pn]. However there does not appear to be a natural way of using total orders to distinguish the following two ways in which two a's might precede two b's.



Thus not only are total orders unnatural, they are not an expressively adequate substitute for pomsets.

{\bf Examples}. The above-drawn pomsets together form a two-element process. Any n-ary relation (hence binary relation, and hence function) is a process if each n-tuple in the relation is regarded as a totally ordered set. A power set is a process if each element is regarded as a set with the empty partial order. The power set C of a power set B is a process if each element of C is regarded as ordered by inclusion on B: event e necessarily follows event d just when e is d
with some additional elements - the process makes progress by accumulating elements and distinct accumulations leading to the same subset are (in this case) considered the same event.

{\bf Spatial Localization}. In order to put processes in communication with each other it is helpful to know where their events are taking place (cf. [W], p.64). We define an {\em event space} to be a Cartesian product $C \times D$, consisting of {\em spatial events}. The intended interpretation is that C is a set of {\em channels} or {\em places} (cf. [B]) where the events may be found and D the set of {\em data} that may be sent over the channels of C. A {\em spatial process} is a process on an event space.

{\bf Nets.} A {\em net} is a process P on $C \times D$ having constituent processes $P_1,\ldots ,P_n$ on $C_1 \times D, \ldots ,C_n \times D$ respectively. Process P_i is a {\em constituent} of P just when there exists a function $a_i : C_i \rightarrow C$ determining a projection $A_i : P \rightarrow P_i$. (a_i gives the attachment of the channels (i.e. ports) of P_i to the channels of the net.) The projection A_i is determined from a_i by taking $A_i(p)$ to be the multiset \{$(c,d) \mid (a_i(c),d)$ is in p\}. Order is preserved, that is, $(c,d) < (c',d')$ in A_i(p) iff ($a_i(c),d) < (a_i(c'),d')$ in p. (Note that A_i need not be onto, i.e. it is not required that P_i equal $A_i(P)$, only that it include it.)

{\bf Process Composition.} Processes are composed to form a new process in two steps: given the processes P_i with corresponding attachments $a_i : C_i \rightarrow C$ for i from 1 to $n-1$, the maximum (under set inclusion) net P having those processes as constituents is formed, and then an additional attachment $a_n : C_n \rightarrow C$ is used to

determine the projection $A_n: P \rightarrow P_n$. The result is $A_n(P)$. The n attachments themselves can thus be seen to determine an $(n-1)$-ary operation on processes.

{\bf Example.} Ordinary composition of binary relations on D is determined by C_1 = C_2 = C_3 = \{0,1\}, C = \{0,1,2\} with $a_1(c) = c$, $a_2(c) = c+1$, and $a_3(c) = 2c$. In this net P_1 and P_2 are composed to yield P_3. This is of course a particularly simple example.

{\bf Bibliography}

[B] Brauer, W., Net Theory and Applications, Springer-Verlag LNCS 84, 1980.

[BA] Brock, J.D. and W.B. Ackerman, Scenarios: A Model of Non-Determinate Computation. In LNCS 107: Formalization of Programming Concepts, J. Diaz and I. Ramos, Eds., Springer-Verlag, New York, 1981, 252-259.

[H] Hoare, C.A.R., Communicating Sequential Processes, CACM, 21, 8, 666-672, August, 1978,

[K] Kahn, G., The Semantics of a Simple Language for Parallel Programming, IFIP 74, North-Holland, Amsterdam, 1974.

[KM] Kahn, G. and D.B. MacQueen, Coroutines and Networks of Parallel Processes, IFIP 77, 993-998, North-Holland, Amsterdam, 1977.

[M] Milner, R., A Calculus of Communicating Systems, Springer-Verlag LNCS 92, 1980.

[Pn] Pnueli, A., The Temporal Logic of Programs, 18th IEEE Symposium on Foundations of Computer Science, 46-57. Oct. 1977.

[Pr] Pratt, V.R., On the Composition of Processes, Proceedings of the Ninth Annual ACM Symposium on Principles of Programming Languages, Jan. 1982.

[W] Winskel, G., Events in Computation, Ph.D. Th., Dept. Comp. Sci, U. of Edinburgh, Dec. 1980.

\end{document}

From hewitt@ai.mit.edu Wed May 29 16:40:13 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Sat, 1 Dec 90 12:26:03 EST
To: concurrency
Subject: Early pomset dissertation
From: hewitt@ai.mit.edu
Sender: meyer@theory.lcs.mit.edu

Date: Fri, 30 Nov 90 01:05 EST

Folks,

Since there is a resurgence of interest in partial order of event models of

concurrent computation, I thought to mention Will Clinger's MIT Math Dept. doctoral dissertation "Foundations of Actor Semantics," Technical Report AI-TR-633, MIT A.I. Laboratory, Cambridge, MA, May 1981.

Readers may be particularly interested in Clinger's results on (non-unique) global times since it is related to the questions about total orderings that have been discussed in this forum.

Also of interest is Clinger's results on functions as special cases of Actors which satisfy the criteria mentioned by Pratt in his IFIP-83 paper. In particular Clinger gives an elegant proof to a theorem of Henry Baker and myself to the effect that any function which can be implemented using Actors satisfies Scott's continuity criterion. In this way, the fundamental assumption of Scott's theory of denotational semantics can be proved as a theorem in the Actor model.

Clinger also developed a fixed point semantics for a simple universal Actor programming language.

To those who send me their US Mail address, I will be happy to send excerpts of his thesis that are being reprinted in the book ``Towards Open Information Systems Science'' edited by Hewitt, Manning, Inman, and Agha, MIT Press, December 1990.

Sincerely,

Carl Hewitt

From infhil!eike@relay.eu.net Wed May 29 16:40:14 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Sat, 1 Dec 90 12:28:33 EST
To: concurrency
Subject: Market
From: infhil!eike@relay.eu.net (Eike Best)
Sender: meyer@theory.lcs.mit.edu

Date: Fri, 30 Nov 90 10:35:07 +0100

Dear Participants of the Concurrency Net,

It seems an excellent idea to fill otherwise idly wasted
net bandwidth with the broadcasting of unpublished notes
in duplication. So far, I received:

-- A note by Pratt.

-- This same note in LaTeX form.

-- This same note in LaTeX form.
   (There seems to be a fixpoint here.)

-- A paper by Hewitt.

And I'm looking forward to spending my time reading and re--reading

many more unpublished notes. If I may suggest sending also **published** papers around in triplicate, or at least in duplicate so that people can give the spare copies to their respective history departments (maybe the nonLaTeXed version, as historians tend to be a little antiquated); from what I hear they are just dying to write down the history of concurrency.

In fact I am now encouraged to plan putting my own unpublished work on partially ordered sets (starting from 1976) three times on the net. Maybe somebody could follow suit with respect to all of the work of the Petri Net and related schools, since that seems to have escaped the attention of concurrency theorists and historians alike:

    Holt et al. 1968
    Petri 1973 and 1976
    Grabowski and Starke 1979
    Shields 1978
    Mazurkiewicz 1977
    Goltz and Reisig 1985
    Fernandez, Nielsen and Thiagarajan 1987
    Best and Fernandez 1988
    ... ...

not to forget:

    Hewitt and Baker 1977
    Lamport 1977.

... ...

Books could perhaps be processed chapter by chapter.

Yours: Eike Best (:-)

From hewitt@ai.mit.edu Wed May 29 16:40:14 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Wed, 12 Dec 90 17:10:13 EST
To: Eike Best <infhil!eike@relay.eu.net>
Cc: concurrency
Subject: Those who do not remember are doommed to ...
From: hewitt@ai.mit.edu
Sender: meyer@theory.lcs.mit.edu
In-Reply-To: <9012011728.AA05532@stork>

Date: Wed, 12 Dec 90 16:56 EST

Eike,

I just saw your important message to this list.

In response I would like to note that you can obtain a copy of Clinger's doctoral dissertation from University Microfilms in Ann Arbor Michigan at their standard publication rate. Unfortunately, the Technical Report publication from the MIT Artificial Intelligence Lab has long been out of print thereby limiting the dissemination of knowledge about Clinger's

pioneering work.

Cheers,

Carl (:-)

From saraswat@parc.xerox.com Wed May 29 16:40:15 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Thu, 20 Dec 90 13:54:41 EST
To: concurrency, logic
Subject: Concurrent Constraint Programming
From: Vijay Saraswat <saraswat@parc.xerox.com>
Sender: meyer@theory.lcs.mit.edu

Date: Thu, 20 Dec 1990 04:25:59 PST
I forgot to send out the abstract of our forthcoming POPL 91 paper
earlier. Among other things, it gives a simple analysis of the
notion of a constraint system.

To obtain a copy, send mail to sslreceptionist@parc.xerox.com.
Vijay

Semantic foundations of concurrent constraint programming

Vijay A. Saraswat, Xerox PARC
Martin Rinard, Stanford University
Prakash Panangaden, McGill University

(Preliminary Report)
October 1990

Concurrent constraint programming [Sar89,SR90] is
a simple and powerful model of concurrent computation based on
the notions of *store-as-constraint* and *process as
information transducer*. The *store-as-valuation* conception
of von Neumann computing is replaced by the notion that the store
is a constraint (a finite representation of a possibly infinite
set of valuations) which provides partial information about the
possible values that variables can take. Instead of ``reading''
and ``writing'' the values of variables, processes may now
ask (check if a constraint is entailed by the store) and
tell (augment the store with a new constraint). This is a very
general paradigm which subsumes (among others) nondeterminate
data-flow and the (concurrent)(constraint) logic programming
languages.

This paper develops the basic ideas involved in giving a coherent
semantic account of these languages. Our first contribution is to
give a simple and general formulation of the notion that a
constraint system is a system of partial information (a la
the information systems of Scott). Parameter passing and hiding
is handled by borrowing ideas from the cylindric algebras of
Henkin, Monk and Tarski to introduce diagonal elements and
``cylindrification'' operations (which mimic the projection of

information induced by existential quantifiers).

The second contribution is to introduce the notion of determinate concurrent constraint programming languages. The combinators treated are ask, tell, parallel composition, hiding and recursion. We present a simple model for this language based on the specification-oriented methodology of [OH86]. The crucial insight is to focus on observing the *resting points* of a process---those stores in which the process quiesces without producing more information. It turns out that for the determinate language, the set of resting points of a process completely characterizes its behavior on all inputs, since each process can be identified with a closure operator over the underlying constraint system. Very natural definitions of parallel composition, communication and hiding are given. For example, the parallel composition of two agents can be characterized by just the intersection of the sets of constraints associated with them. We also give a complete axiomatization of equality in this model, present a simple operational semantics (which dispenses with the explicit notions of renaming that plague logic programming semantics), and show that the model is fully abstract with respect to this semantics.

The third contribution of this paper is to extend these modelling ideas to the nondeterminate language (that is, the language including bounded, dependent choice). In this context it is no longer sufficient to record only the set of resting points of a process---we must also record the path taken by the process (that is, the sequence of ask/tell interactions with the environment) to reach each resting point. Because of the nature of constraint-based communication, it turns out to be very convenient to model such paths as certain kinds of closure operators, namely, bounded trace operators. We extend the operational semantics to the nondeterminate case and show that the operational semantics is fully consistent with the model, in that two programs denote the same object in the model iff there is no context which distinguishes them operationally.

This is the first simple model for the cc languages (and ipso facto, concurrent logic programming languages) which handles recursion, is compositional with respect to all the combinators in the language, can be used for proving liveness properties of programs, and is fully abstract with respect to the obvious notion of observation.

From jeffrey@cs.chalmers.se Wed May 29 16:40:16 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Thu, 20 Dec 90 14:23:55 EST
To: concurrency
Subject: Zeno machines
From: Alan Jeffrey <jeffrey@cs.chalmers.se>
Sender: meyer@theory.lcs.mit.edu

Date: Wed, 19 Dec 90 15:01:35 +0100

God Jul everyone,

A short question about timed concurrency... One of the standard counter-examples to throw at any real-timed model is the {\em Zeno machine\/} defined

```
ZENO_{n}  =  WAIT 2^{-n} ; a ; ZENO_{n+1}
```

So ZENO_1 does an a at time 1/2, then at time 3/4, then at time 7/8,...

So despite being a guarded recursion (well, sort of guarded, depending how you define guardedness) it can do an infinite number of actions before time 1.

The question is, who coined the name `Zeno machine'? It seems to have entered the folklore, but I wondered if anyone had a reference to the name being used in the literature.

A suitably merry Christmas and New Year to all,

Alan.

Alan Jeffrey Tel: +46 31 72 10 98 jeffrey@cs.chalmers.se
Department of Computer Sciences, Chalmers University, Gothenburg, Sweden

From jcm@cs.stanford.edu Wed May 29 16:40:16 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Fri, 21 Dec 90 16:35:27 EST
Cc: concurrency@theory.lcs.mit.edu
Subject: Re: Zeno machines
From: John C. Mitchell <jcm@cs.stanford.edu>
Sender: meyer@theory.lcs.mit.edu

To: Alan Jeffrey <jeffrey@cs.chalmers.se>
In-Reply-To: Your message of Thu, 20 Dec 90 14:23:55 -0500.
<9012201923.AA19323@stork>
Date: Thu, 20 Dec 90 14:43:02 -0800

This example seems odd to me. If a programming language has a WAIT statment, then shouldn't the semantics of doing an action include some elapsed time? Otherwise, what can the WAIT be waiting for? If this is so, then the "a" action in

```
ZENO_{n}  =  WAIT 2^{-n} ; a ; ZENO_{n+1}
```

takes some amount of time, and the process can only do a finite number of actions before time 1.

The example reminds me of a standard "trick" question used in MIT oral exams: Suppose a class of Turing machines has the property that the first instruction takes 1/2 secound to execute, the second 1/4 second, the third

1/8 second, and so on. Is the halting problem solvable for this class of machines?

The point of the question is to debug the question, not answer it.

John Mitchell

From jeffrey@cs.chalmers.se Wed May 29 16:40:17 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Fri, 21 Dec 90 16:42:17 EST
Cc: concurrency
Subject: Re: Zeno machines
From: Alan Jeffrey <jeffrey@cs.chalmers.se>
Sender: meyer@theory.lcs.mit.edu
In-Reply-To: John C. Mitchell's message of Thu, 20 Dec 90 14:43:02 -0800
<9012202243.AA03282@Iswim.Stanford.EDU>

Date: Fri, 21 Dec 90 10:51:21 +0100
To: jcm@cs.stanford.edu

>John C. Mitchell <jcm@cs.stanford.edu>

>This example seems odd to me. If a programming language has a WAIT
>statment, then shouldn't the semantics of doing an action include
>some elapsed time? Otherwise, what can the WAIT be waiting for?
>If this is so, then the "a" action in

>   ZENO_{n} = WAIT 2^{-n} ; a ; ZENO_{n+1}

>takes some amount of time, and the process can only do a finite number of
>actions before time 1.

Well, that's certainly one way of modelling the world (c.f. Timed CSP) but it has its problems. Specifically, you can't have a version of the expansion lemma. For example,

   a ||| b = ab + ba

doesn't hold if the actions must take time, but does hold if they're instantaneous. This is why (Muller and Tofts) and (Hennessy and Regan) have got a complete axiomatization for their languages and Timed CSP hasn't.

The problem is that if you allow instantaneous prefixing, you allow an awful lot of very uncomputational processes, like the Zeno Machine. How worried you are about that depends on taste. Personally I can live with uncomputational structures living in a model, if they make the algebra easier to work with---the analogy is with complex numbers, which have no `real world' equivalent but are damn useful.

What's more worrying is that in some models (notably transition systems) there's no concept of behaviour after doing an infinite number of actions (you `drop off' the bottom of the T.S.) so a process like ZENO_1 is a `time stop'. No process placed in parallel with it will ever get to time 1, because ZENO_1 is flooding the T.S. with

actions. Again, how worried you are about this depends on taste.

There seems to be two distinct viewpoints going on here, the Timed CSP viewpoint, where everything in the model (well, almost everything) is computational; and the Timed CCS viewpoint, where there's a lot of uncomputational structures in the model, but they make the algebra easier to deal with.

It's all swings and roundabouts I guess...

>John Mitchell

Cheers,

Alan.

Alan Jeffrey Tel: +46 31 72 10 98 jeffrey@cs.chalmers.se
Department of Computer Sciences, Chalmers University, Gothenburg, Sweden

From jcm@cs.stanford.edu Wed May 29 16:40:17 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Fri, 21 Dec 90 16:50:27 EST
Cc: concurrency
Subject: Re: Zeno machines
In-Reply-To: Your message of Fri, 21 Dec 90 10:51:21 +0100.
             <9012210951.AA21747@birk.cs.chalmers.se>
From: John C. Mitchell <jcm@cs.stanford.edu>
Sender: meyer@theory.lcs.mit.edu

To: Alan Jeffrey <jeffrey@cs.chalmers.se>
Date: Fri, 21 Dec 90 10:45:57 -0800

Here's what I don't understand. If you have a WAIT(n) statement
or guard, where n is supposed to be time in some units, then it
seems that you must have some idea of the passage of time.
If you are waiting, then time should pass while you are doing so.
What are the things that happen as time goes by? If actions are
instantaneous, then of course you could do infinitely many of them
before time 1; the WAIT statement is just a red herring in this
case. On the other hand, if actions happen over time, then
they should have some duration. This would seem the more realistic
case, if you care to think about time. And so what if

   a || b = ab + ba

fails? This equation seems arguable anyway. The main thing is that
I can't see the motivation for you mixed set of assumptions about
time, so the fact that you get a "paradox" seems an indictment of
your assumptions.

John

From pratt@cs.stanford.edu Wed May 29 16:40:18 1991

Return-Path: <meyer@theory.lcs.mit.edu>
Date: Sat, 22 Dec 90 04:04:25 EST
To: concurrency
Subject: Re: Zeno machines
From: pratt@cs.stanford.edu
Sender: meyer@theory.lcs.mit.edu
In-Reply-To: Your message of Fri, 21 Dec 90 16:35:27 EST.
<9012212135.AA20407@stork>

Date: 21 Dec 90 22:35:50 PST (Fri)

The first time I heard the term "Zeno machine" was from Carl Hewitt around 1979, whose office was then next door to mine. He would therefore be a good person to ask concerning its origin.

Carl was concerned about fairness in actor semantics. I believe the concern was that a Zeno machine might be accorded unfairly preferential treatment. A certain axiomatization of actors was purported either to rule out the possibility of such Zeno machines, or to depend for its efficacy on their absence, I forget which. The sense of the axiomatization was that every unblocked machine waited only a finite length of time before taking its next step. In the absence of Zeno machines computation was then supposed or expected to be inherently fair.

Concurrency divides somewhere near the root into real time and ordered time. As I understood actors at that time, their official semantics were of the latter kind, at least in the sense that neither reals nor integers appeared explicitly anywhere in actor theory. Indeed the earliest partial order model of concurrency appears to be Irene Greif's 1975 axiomatization of actors; the following extract starting on p.7 of her thesis gives a very modern motivation for partial orders.

> Partial orders are appropriate for characterization of computer systems since, using them, orderings that must exist among certain operations can be expressed without the hypothesizing of total orders over all operations. Some operations may not be related due, for instance, to lack of physical connection between the processors on which they occur. Other operations, while they can be ordered as they occur, will in fact occur in unpredictable orders each time the computer system is run. Even though quantifying over all possible orders of operations in all parts of a sytem may in some sense be guaranteed to capture all possible properties, it can in fact obscure the important properties. The important properties will generally be the properties of ordering relations which are common to all of the postulated total orders. Such common properties should be abstracted from the set of all possible total orders and expressed directly as a partial order.
>
> Choosing a model. The actor model of computation on which this specification language is based has several properties which relate well to the physical realities of parallel processing. ... In real computer systems the speed at which the messages travel would be a factor. In the actor model that factor cannot be relied on, making it impossible to write specifications for systems which could be realizable only in

> limited kinds of environments (e.g. on particular computer configurations.)
>
> Individual actors are specified by *causal axioms* which are properties that will hold for any behavior of any actor system of which the particular actor is a part. They specify relations among events. These relations are then the properties of the behaviors as partial orders.

This makes very clear that the essence of actors, at least as understood in 1975, was order theoretic.

Now I claim that the concept of a Zeno machine, at least in isolation, is meaningless for a purely ordered theoretic model. Consider the binary numbers .1, .11, .111, ..., that is, 1/2, 3/4, 7/8, ... As a sequence of reals this sequence has limit 1. This is the sequence that arises with Alan Jeffrey's notion of Zeno machine.

However as a linearly ordered set it is isomorphic to the ordered set of natural numbers. Thus on the basis only of order of events, a Zeno machine is indistinguishable from any other machine executing an infinite number of steps.

So if actor semantics is really order theoretic, with no notion of real number or measure, the concept of a Zeno machine would seem inappropriate.

However for ordered time one may capture at least the spirit of Zeno machines in terms of the notion of unfair augment. There is a wealth of literature on fairness, including Nissim Francez's book of that title, and I apologize for citing my work instead in what follows. My excuse is that so little of the prior work on fairness benefits from partial orders in the manner illustrated in the following.

Following Gaifman and Pratt, LICS-87, define the rank of an element v of a well-founded poset to be the set rank(v) = {rank(u)|u<v}. (Identifying the ordinal n with the set of ordinals less than n makes rank an ordinal, with those elements without predecessor having rank 0.) Define a fair event to be one with finite rank, and a fair poset one with all elements fair. (Exercise: Show that requiring a fair element to have only finitely many predecessors yields a strictly stronger notion of fairness, while requiring that it have no infinite chain leading up to it is strictly weaker.)

An augment of a poset is the same set with the same or a stronger ordering. A fair augment of a fair poset p is an augment of p that is a fair poset. In particular a fair linearization of p is a fair linear augment of p.

For a process P to synchronize with other processes it may be necessary to admit augments of posets of P, since the posets being synchronized with may be more ordered (they might be linear, for example). It is at this point that the requirement of fairness is most naturally incorporated into the semantics, namely by permitting only fair augments when synchronizing.

In fact this should happen automatically if one defines synchronization

of two fair posets such that augmentation only sufficient to achieve synchronization be imposed. Thus fairness is obtained as a theorem. The fairness requirement enters in a more essential way when the alternative approach is adopted of keeping all processes "augmented closed" from the outset, so that no additional augmentation is required when synchronizing. It is here that one needs to rule out unfair augments explicitly. Here fairness is obtained by fiat rather than as a theorem.

A Zeno machine could then be taken to be one which when synchronized with a fair non-Zeno machine yields behaviors containing unfair events performed by the latter machine. The axiom forbidding unfair augments can then be interpreted as ruling out the existence of, or at least rendering impotent, such Zeno machines.

Vaughan Pratt

PS. The sequence .1,.11,.111,... suggests suitable pedagogy for introducing higher ordinals: define them as the order type of certain sets of reals all in [0,1]. Problems:

1. Let k be a nonnegative integer. What is the order type, under the standard ordering of the reals, of the set of finite-length binary numerals of the form .(0,1)*1 containing at most k occurrences of 0? (For k=0 it is evidently omega.)

2. Give an ordinal bounding the order type of any well-ordered set of reals in [0,1] representable as a regular set of finite binary strings, interpreted as starting with a binary point. Give corresponding ordinals for the rest of the Chomsky hierarchy.

These sets also afford an opportunity to exercise topological notions. Assume the standard (real) topology on [0,1]. For each of the sets X in the previous problem state whether X is (i) open (ii) closed (iii) a boundary (iv) Hausdorff (v) compact. Which of the answers change with the discrete topology on [0,1]? The coarse topology? The topology whose open sets are all sets X_a = {y|y<a}? All X_a = {y|a<y}?

From infhil!eike@relay.eu.net Wed May 29 16:40:19 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Mon, 24 Dec 90 10:46:26 EST
To: concurrency
Subject: Re: Zeno Machines
From: infhil!eike@relay.eu.net (Eike Best)
Sender: meyer@theory.lcs.mit.edu

Date: Sun, 23 Dec 90 12:19:23 +0100

The discussion on 'Zeno Machines' had been brought up already about a year ago by Mathai Joseph. At that time I sent him a remark which did not make it to the net. So here is an extended version of my comment:

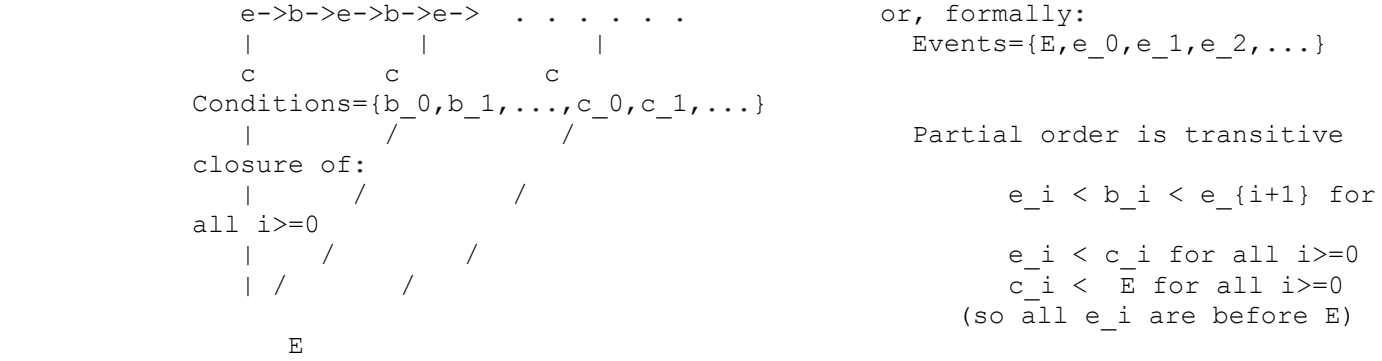
The situation being discussed is typical of what in Petri Net Theory is called the 'Non--discreteness' of a behaviour.

The Petri Net view of behaviour is by means of special partial orders. Every behaviour is simply a partial order of events, with ordering meaning prior/later, and absence of order meaning concurrent. Between any two immediately successive events there must be a so-called condition which describes the local state between the two (i.e. the state in which the first has just ended but the second has not yet begun).

In 1976 Petri invented something he called K--density, which can be stated order--theoretically as 'every maximal chain intersects every maximal antichain', and interpreted as 'every sequential subprocess intersects every global state'.

In 1977 I found a characterisation of K--density which I described in a Technical Report of the University of Newacstle upon Tyne, in a talk at a Workshop in Aarhus (at which I met Carl Hewitt for the first time), and which was finally was published as a paper in Fundamenta Informaticae in 1980 (Vol.III.1, pp.77-94).

One of the consequences of this characterisation is that K--density implies the absence of partial orders which describe 'an infinite sequence of events before another one', like the following one (where | and / mean downward arrows):



or, formally:

Events={E,e_0,e_1,e_2,...}

Partial order is transitive closure of:

e_i < b_i < e_{i+1} for all i>=0

e_i < c_i for all i>=0

c_i < E for all i>=0

(so all e_i are before E)

In my article I discussed this situation using Achilles and the tortoise, which is however the same problem as discussed in connection with Zeno machines. The article may be interesting to read but goes a bit overtop in proposing K--density as an axiom. In the meantime we think that an axiom called 'Discreteness w.r.t. a cut' is the one to require; it is weaker than K--density.

This discussion bears a connection with the interleaving vs. partial orders discussion, since one may observe that the above partial order, while being linear on its events, canNOT be brought into the form of what one normally would describe as an interleaving (which really must be order--isomorphic with the set of integers or a subsection thereof). Discreteness w.r.t. a cut essentially captures this property of being capable of being linearised into an interleaving.

So the bottom line is: Assuming one of these axioms, Zeno's 'process' is not a behaviour.

Our book `Nonsequential processes: a Petri net view', Springer EATCS Monographs No.13 (1988) discusses some of the connections between various properties of this kind of partial orders, refraining however from giving

any interpretations. Amongst others, it contains exact conditions for a partial order to have an interleaving linearisation.

Other, perhaps more easily accessible articles are:

(1) E.Best: Concurrent Behaviour: Sequences, Processes and Axioms. Lecture Notes in Computer Science Vol.197 (eds. Brookes, Roscoe, Winskel), 221-245 (1984).

(2) E.Best and R.Devillers: Sequential and Concurrent Behaviour in Petri Net Theory. TCS (Theoretical Computer Science) Vol.55/1, 87--136 (1987).

Petri's original article is:

Nichtsequentielle Prozesse. Gesellschaft fuer Mathematik und Datenverarbeitung Bonn, ISF-Bericht 76-6; an English translation is available.

Copies can be obtained from GMD or through me:

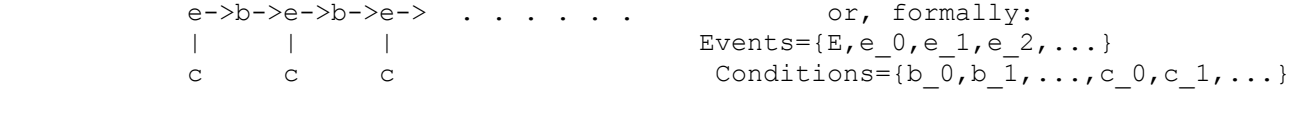
Eike Best, Institut fuer Informatik,
Universitaet Hildesheim, W-3200 Hildesheim.

From pratt@cs.stanford.edu Wed May 29 16:40:20 1991
Return-Path: <meyer@theory.lcs.mit.edu>
Date: Fri, 28 Dec 90 17:55:33 EST
To: concurrency
Subject: Re: Zeno Machines
From: pratt@cs.stanford.edu
Sender: meyer@theory.lcs.mit.edu
In-Reply-To: Your message of Mon, 24 Dec 90 10:46:26 EST.
<9012241546.AA21550@stork>

Date: 24 Dec 90 21:08:49 PST (Mon)

> From: infhil!eike@relay.eu.net (Eike Best)
>
> In 1976 Petri invented something he called K--density, which can be stated order--theoretically as 'every maximal chain intersects every maximal antichain', and interpreted as 'every sequential subprocess intersects every global state'.
> ...
> One of the consequences of this characterisation is that K--density implies the absence of partial orders which describe 'an infinite sequence of events before another one', like the following one

(Eike's diagram appears to have been mangled en route, most likely due to improper expansion of tabs. My guess is that it originally looked like:)



Partial order is transitive closure of:
  e_i < b_i < e_{i+1} for all i>=0
              e_i < c_i for all i>=0
                 c_i < E for all i>=0
                  (so all e_i are before E)

As a minor quibble, even if K-density did rule out such infinite chains, these constitute only the weakest of the three kinds of "event at infinity" I described in my previous message, the next stronger being that of having infinite rank and the strongest being that of having infinitely many predecessors.

A more serious objection is that, contrary to what I understand Eike to be claiming, K-density does not rule out even the weakest of these kinds of "event at infinity."

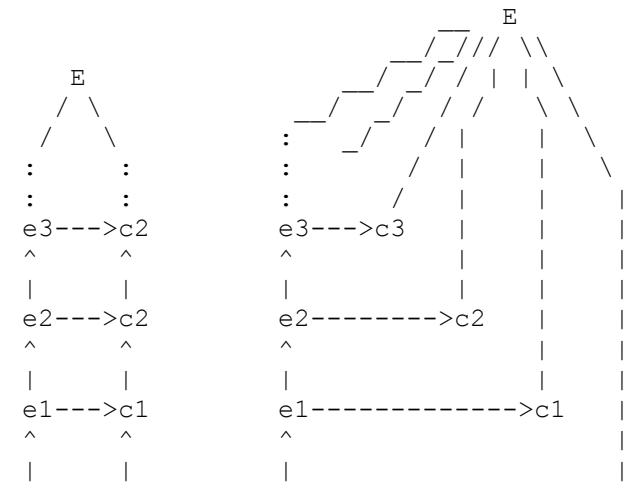
Theorem. Every linear order is K-dense.

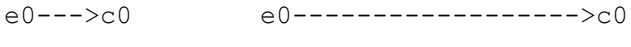
Proof. A linear order contains only one maximal chain, namely itself, but its maximal antichains are all and only its singleton subsets.

Taking E = omega in the ordinal omega+1 then supplies a suitable counterexample of a K-dense poset containing an "event at infinity."

It is however true that Eike's example is not K-dense: the c's constitute a maximal antichain while the e's, b's, and E constitute a maximal chain. Since this seemed a rather unlikely counterexample, I found myself wondering just what this was a counterexample to. After looking up the 1976 paper that Eike referred to and thinking a while about it, I think I understand. This is a counterexample to the converse of a theorem by Petri in that paper, that every K-dense poset is N-dense.

Here are two versions of Eike's counterexample. The one on the right is essentially what Eike gave (the b's contribute nothing and I've dropped them). The one on the left is a plausible variant where the conditions c_i happen in the order caused.



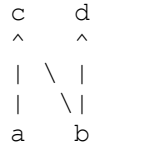


Neither poset is K-dense. However the one on the left already fails K-density within the first three events. Just take the maximal antichain c0,e2 and the maximal chain beginning e0<e1<c1.

The one on the right is more interesting. Here there exists a (unique) infinite maximal antichain, namely c0,c1,c2,... and a maximal chain e0<e1<e2<...<E; this combination of chain and antichain is the only one with an empty intersection.

What is special about the example on the right is that it is N-dense. Here are the details.

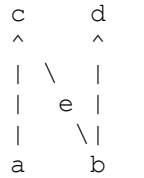
Definition. N is the four-element poset



(Note that N is isomorphic to its order dual, a property holding of exactly half of the 16 4-element posets. Does anyone happen to know whether the self-dual posets are in a minority or majority of the n-element posets as n -> oo ?)

A poset is said to be N-free when it does not contain N as a subposet.

Now at this point one might be inclined to guess that a partial order that contains an N cannot be K-dense. The following supplies an easy counterexample.



The maximal antichains of this poset are a,b; a,e,d; and c,d. These intersect the respective maximal chains a,c (or b,d); b,e,d; and a,c (or b,d). Hence this is a K-dense poset containing an N.

Definition (Petri, IMMD Jubilee Colloq. 1976; GMD Report ISF-77-05, June 15, 1977). P is *N-dense* when for every occurrrence of N as a subposet of P there exists an element x such that b<x<c (where b,c are as in the above definition of N and < is strict.)

Theorem (Petri, op.cit.) (AC). K-dense -> N-dense.

Petri did not supply a proof, and did not mention requiring AC, the Axiom of Choice. Here is an easy proof using AC. My guess is that the theorem fails in the absence of choice, but I do not have a proof of this and would very much like to see one.

Proof. Let P be K-dense, and suppose {a,b,c,d} is any occurrence of N

in P. By AC we may choose a maximal chain containing b and c, and a maximal antichain containing a and d. By K-density there exists an element x of P common to the chain and the antichain. x <= b is impossible since x would then compare with d. Similarly c <= x is impossible since x would then compare with a. Since x lies on the chain it follows that b<x<c. QED

Inspection of Eike's example and my variant of it shows that although both are not K-dense, only the former is N-dense.

What I don't understand is, what is the significance of either? As I argued above, whatever the significance of K-density, it is not what Eike claimed. Moreover I can't think what small change would help. The interesting counterexample to "N-dense -> K-dense" (it seems very likely that any counterexample must be infinite) suggests that the thought was that under the assumption of N-density, the impact of further constraining things by requiring K-density would be to rule out "events at infinity." However this hope is dashed by the theorem that every chain is K-dense.

Something has been left out.

Although W. Reisig's very nice 1985 book on Petri nets defines K-density, it does not use it for anything, nor does it mention N-density.

Vaughan Pratt