Causal trees

Philippe Darondeau, Pierpaolo Degano

Rapports de Recherche

N° 952

Programme 1

CAUSAL TREES

Philippe DARONDEAU
Pierpaolo DEGANO

Décembre 1988

RR.0952

CAUSAL TREES

Philippe DARONDEAU

Pierpaolo DEGANO

Publication Interne n° 442

Décembre 1988

IRISA INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
35042-RENNES CÉDEX
FRANCE
Téléphone: 99 36 20 00
Télex: UNIRISA 950 473 F
Télécopie: 99 38 38 32

Publication Interne n°442 - Décembre 1988 - 44 Pages

CAUSAL TREES *

Philippe Darondeau [1] & Pierpaolo Degano [2]

Abstract. The intent of the paper is to reconcile two antagonist views on bisimulation semantics for concurrency: the interleaving approach and the approach by partial orders. The so-called causal trees are a variant of Milner's synchronisation trees with enriched action labels which supply indication of the observable causes of observable actions. Concerning CCS, we construct an algebra of causal trees with two byproducts: a complete axiomatization of weak causal bisimulation between finite terms and a fully abstract model of recursive programs.

ARBRES CAUSAUX *

Résumé. Le propos de cet article est de réconcilier deux approches antagonistes quant à la sémantique des systèmes concurrents par bisimulations: l'approche par entrelacements et l'approche par ordre partiels. Les arbres causaux sont une variante des arbres de synchronisation de Milner, dans lesquels l'étiquetage indique les causes observables des actions observables. Nous définissons pour CCS une algèbre d'arbres causaux dans laquelle nous construisons un modèle pleinement abstrait des programes récursifs et une axiomatisation complète de la bisimulation causale faible entre termes finis.

[1]Université de Rennes, IRISA, Campus de Beaulieu, F35042 Rennes-Cedex, FRANCE

[2]Universita di Pisa, Dipartimento di Informatica, Corso Italia 40, I56100 Pisa, ITALIA

**Work undertaken during a stay of the second author as a visiting professor in Rennes.*

I. Introduction

The purpose of the present note is to reconcile two antagonist views on bisimulation semantics for concurrency: the interleaving approach (see, e.g., [AB84,BHR84,BK84,DG87,Hen88,Mil80,Mil85,Niv82]) and the approach by partial orders (see,e.g., [BC88a,BC88b,BS87,Dar80,DM87a,DM87b,Lam78,Maz77, NPW81,Pet80,Pra86,REX88,Win80,Win82]). The interleaving approach, probably best illustrated in the expansion theorem of Milner's Calculus of Communicating Systems [Mil80], takes as a basic principle reduction from asynchrony to concurrency, and has the main advantage of simplicity. The approach by partial orders, as taken in Winskel's Event Structures [NPW81], aims at a faithful treatment of causal dependencies and avoids carefully any confusion between concurrency and independence. The drawback in the latter approach is a general trend to deal with non syntactic entities, such as partial orders and their homomorphisms, despite recent effort to adhere more tightly to operational definitions. But the resulting models are not so handy, essentially because they are not treelike and have therefore no obvious syntactic description.

Our intent is to recast partial ordering semantics in the framework of trees and draw all possible benefits from the technical facilities afforded in this way. The so-called causal trees are a variant of Milner's Synchronization Trees [Mil80], with enriched action labels which supply indications of the visible causes of visible actions. As regards CCS, our privileged field of application, we construct an algebra of causal trees with two byproducts: a complete axiomatization of weak causal bisimulation between finite terms, and an abstract model of recursive programs. Only a minor effort was needed for achieving these goals: most of the technical steps are adaptions of similar steps for interleaved semantics. This is true for the τ-laws, which are carried on unchanged, and also for a modified form of the expansion theorem which holds for causal trees, although this theorem may read as interleaving! Indeed, causal trees are an interleaved but faithful representation of partial orders.

There already exist in the litterature several papers where semantics expressing causality are defined for CCS-like languages. Some representative instances are [BC88a,BC88b,DDM87b,DDM88a,DDM88b,DDM88c, DGM87,GV87,Old87]. However, these papers provide neither a complete axiomatization for finitary agents of the considered languages (with communication), nor fully abstract models for them (including the recursive case), which are in our opinion the two main achievements of our work. Actually, Boudol and Castellani give in [BC88a] a complete axiomatization for a language *without communication* .

We think the use of causal trees and the like may bring further clarifications and new developments in many fields of concurrency, beginning with Petri nets [Pet80,Rei85]. For instance, causal trees supply representations for labelled event structures [MS81,NPW81,Win82,Win87] and for non deterministic measurement systems [DDM87a], amenable to algebraic laws for bisimulation semantics. Some efforts are needed for investigating this direction. A comparison between our causal equivalence and other equivalences preserving other forms of causality (e.g.[BC88b,CH87,DDM88b]) is also in order.

The paper is organized as follows. Causal trees are smoothly introduced on top of labelled event structures (II); causal trees induced by CCS programs viewed as transition systems are then defined (III). A general basis is provided for building up algebras of causal trees (IV); a specific algebra is constructed for CCS (V), and shown a fully abstract model (VI). Weak causal bisimulation is finally studied and axiomatized for finitary agents (VII).

II. Causal trees and labelled event structures

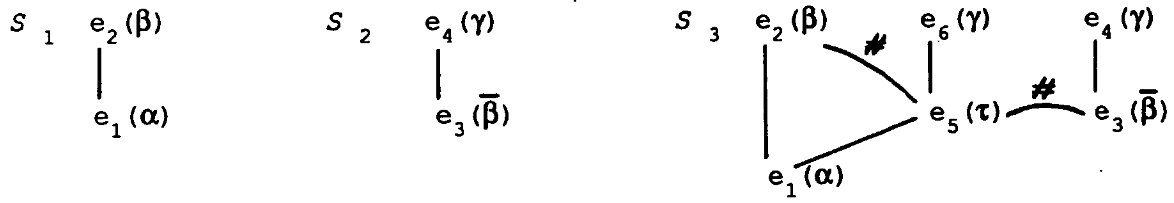
In order to help the intuition, we offer rules for translating labelled event structures into 'naive' causal trees and show some simple examples. Owing to synchronized events, we encounter serious difficulties for defining inductively the parallel composition of naive causal trees; we then introduce an improved version of causal trees.

Let E be a set (of events). An *event structure* on E is a structure $S = (E,\leq,\#)$ or $(E,\leq,\#,\wedge)$ where $\{\leq,\#,\wedge\}$ is a partition of ExE, built up from a transitive *causality* relation ($\leq$), an $\leq$-hereditary *conflict* relation ($\#$), and an *independence* relation ($\wedge$). The computations of S are the $\leq$-left closed and conflict free subsets of E. A *trace* of S is a total order on some computation of S, compatible with $\leq$ (i.e. included in $\leq$). A trace may be represented as a word $e_1 \ldots e_n$ with no letter $e_i \in E$ occuring twice. Note that we consider a proper subset of the event structures defined in [NPW81] where # need not be hereditary. Let L be a set (of labels). A *labelled event structure* (or LES) on E and L is a structure $S = (E,\leq,\#,l)$ where $(E,\leq,\#)$ is an event structure, and $l: E \dashrightarrow L$ is a labelling function. In the sequel, L is taken as $\Lambda \cup \{\tau\}$, where Λ is a set of labels for *observable* events and τ ($\notin\Lambda$) is a symbol for *unobservable* events. For any observable event e, the *direct observable causes* of e are the $\leq$-maximal events in $C(e) = \{ e' \in E \mid e' \leq e \ \& \ e' \neq e \ \& \ l(e') \neq \tau\}$.

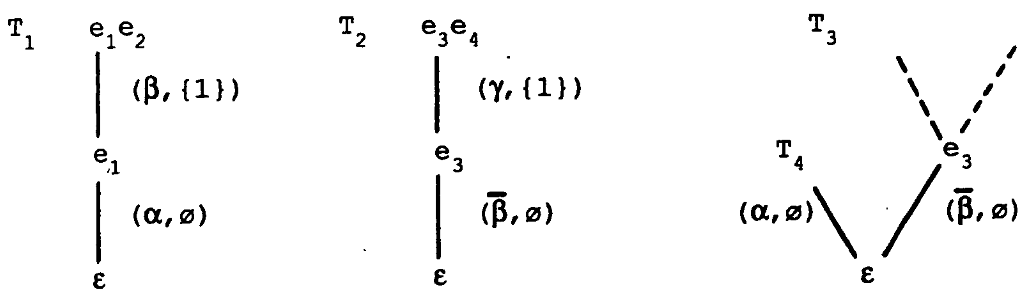
A LES S gives rise to a *causal tree* $T = (V,E,L)$ as follows:

- vertices in V are the traces of S ,
- edges in E are pairs (v,ve) in $V \times V$ such that $e \in E$,
- $L : E \dashrightarrow \{\tau\} \cup (\Lambda \times P(N_+))$ is the labelling function $L(v,ve) = \tau$ if $l(e) = \tau$ else $(l(e),C(e,v))$, letting $C(e,v)$ be the set of references for the direct observable causes of e relative to v,
- the *relative reference* for e_i in $e_1...e_i...e_n$ is the number of observable events in the right factor $e_i...e_n$ (and thus may be thought of as a backwards pointer to e_i)

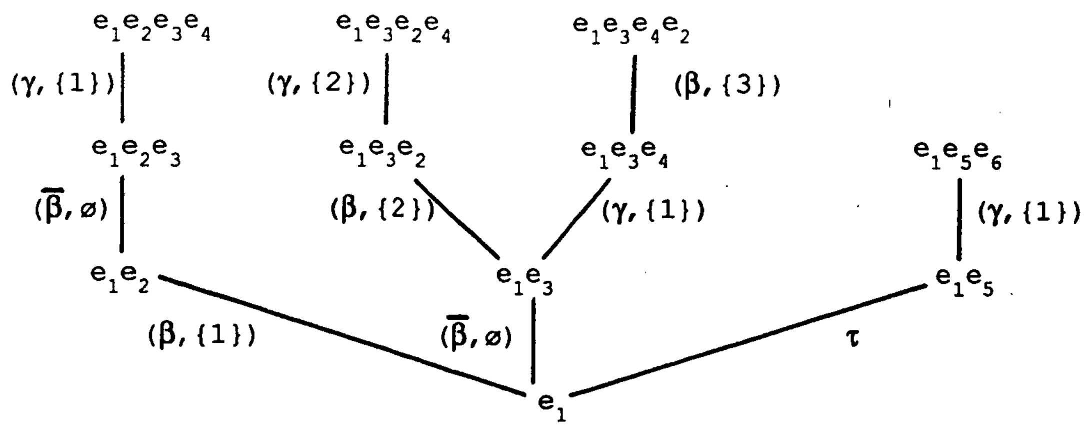
For the sake of illustration, let us consider the following LES's S_1, S_2 and S_3, reminiscent of CCS agents $A_1 = \alpha.\beta.nil$, $A_2 = \bar{\beta}.\gamma.nil$ and $A_3 = (\ \alpha.\beta.nil \mid \bar{\beta}.\gamma.nil\ )$. Those LES's are represented as Hasse diagrams (growing upwards), supplemented with #-arcs indicating their minimal pairs in conflict. Labels of events are written aside within parentheses.



The associated causal trees are respectively:



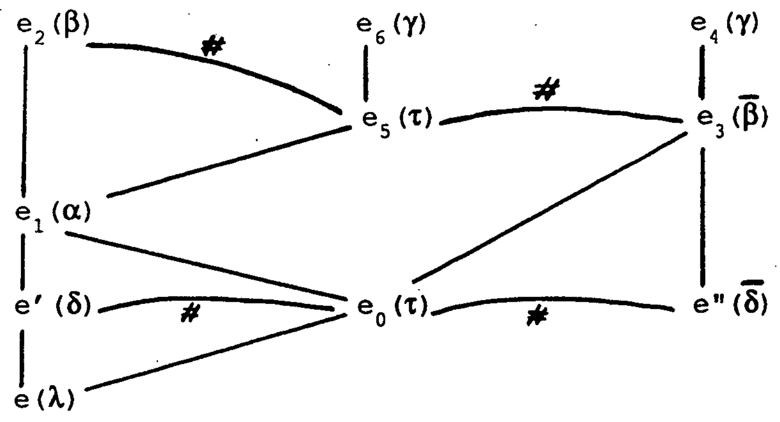
where T_4 stands for the following causal tree (we leave the reader fill in the dots above e_3):



Remark For each trace in an event structure S_i - e.g. $e_1e_3e_4e_2$ in S_3- there exists a unique corresponding path in the causal tree T_i -e.g. $(\alpha,\varnothing)(\overline{\beta},\varnothing)(\gamma,\{1\})(\beta,\{3\})$ in T_3. A path allows to recover causality relation $\leq_i$ restricted to observable events occuring in that trace. Furthermore, causal tree T_i ($i\leq 3$) may be considered up to isomorphism, thus ignoring that vertices are traces: the label of a path still determines the partially ordered multiset of observable actions generated along that path. This is not true for the causal tree T_4, which has dangling references such as 3 in $(\beta,\{3\})$.

The apparent difficulty to define inductively the parallel composition of naive causal trees stems from synchronized communications which induce cross inheritance of causes. In S_3 above, e_6 inherits cause e_1 from e_5 which represents synchronization between e_2 and e_3, but this side effect makes no problem because e_2 and e_3 have no $\leq$-comparable causes. Let us consider the following event structure S which augments S_3 and can be derived from the CCS agent $(\lambda.\delta.\alpha.\beta.\text{nil} \mid \overline{\delta}.\overline{\beta}.\gamma.\text{nil})$:

S



The unique path with trace $ee_0e_1e_5e_6$ in the causal tree T induced by *S* is labelled $(\lambda,\emptyset)\tau(\alpha,\{1\})\tau(\gamma,\{1\})$ and results from synchronized composition of paths $ee'e_1e_2$ and $e''e_3e_4$. If the parallel composition of naive causal trees was defined inductively that sequence would arise from the inductive composition of sequences $(\lambda,\emptyset)(\delta,\{1\})(\alpha,\{1\})(\beta,\{1\})$ and $(\bar{\delta},\emptyset)(\bar{\beta},\{1\})(\gamma,\{1\})$, neverminding the exact definition. In the resulting sequence, a single cause is picked up for γ from the set $\{1,2\}$ (or $\{\alpha,\lambda\}$) formed by amalgamating the respective causes of β and $\bar{\beta}$ ($\bar{\beta}$ inherits cause λ from δ via communication e_0). Nevertheless, at the time $(\lambda,\emptyset)\tau(\alpha,\{1\})$ is produced along the induction, one is left with the pair of residuals $(\beta,\{1\})$ and $(\bar{\beta},\{2\})(\gamma,\{1\})$ -remind that τ's are skipped by backwards pointers- where nothing indicates that 2 refers to the cause λ of the cause α of β! Hence the resulting sequence would certainly be $(\lambda,\emptyset)\tau(\alpha,\{1\})\tau(\gamma,\{1,2\})$!

The simplest way out is to introduce redundancy by joining hereditary causes to direct causes in the labelling of causal trees. Since hereditary causes were already defined unambiguously in naive causal trees, the neat effect of the extension is to make this information available from subtrees, and thus accessible in inductive statements on trees.

Technically, the modification amounts to revise the definition of observable causes C(e,v) in the statement of the labelling function *L*. In the revised version, we set $C(e,v)= \{(k_1,K_1),\dots,(k_n,K_n)\}$ where $\{k_1\dots k_n\}$ is the set of references for the immediate causes e_i of e relative to v, and the K_i are sets of references for the hereditary causes of e_i, relative to v and excluding e_i itself. Partially ordered sets K_i could of course have been chosen, but the extra cost in complexity is useless. Thus, function *L* is valued in the set $\{\tau\} \cup (\Lambda \times P(N_+ \times P(N_+)))$ and the following is verified for each label $L(e)= (\lambda, \{(k_1,K_1),\dots,(k_n,K_n)\})$:

$\forall i \forall j\ (k_i \notin K_j)$ and $\forall i \forall k\ (k \in K_i \Rightarrow k_i < k)$ **is-cause**.

In the sequel, *K* denotes the set of pairs (k,K) with finite K satisfying $k' \in K \Rightarrow k<k'$, *KK* is the family of finite sets $KK \in P(K)$ satisfying **is-cause**, and **L** stands for set of labels $\{(\lambda,KK) \mid \lambda \in \Lambda$ and $KK \in K\}$.

Generalizing on the above, we call a *causal tree* ($\in$ CT) any synchronization tree T labelled on $\{\tau\} \cup$ **L** . We consider causal trees up to strong bisimulation in Milner's sense [Mil80]. We thus obtain a syntax for causal trees with operations $\tau.$ and $(\lambda,KK).$ of *prefixing* (by τ or by some pair (λ,KK) in **L**) and *sum* + (associative, commutative, idempotent, and with neutral element *nil* = $\Sigma\{T_i \mid i \in \emptyset\}$). Next section specifies causal trees induced by CCS programs.

The following notations and definitions apply to causes $KK \in$ *KK* :

- $\mathrm{dom}(KK) = \{k \mid \exists K\ (k,K) \in KK\}$ and $KK(k)=K$ for $(k,K) \in KK$,
- KK+n means the increment by n of all integers occuring in KK, and similarly for K+n, $\delta(KK)=$ KK+1 and $\delta(K)=$ K+1,
- *KK is the set of all integers occuring in KK, thus $\delta(*KK)=*(\delta(KK))$,

- The fusion of causes KK′,KK″ is the set KK = KK′ | KK″ defined as follows. Let *KK = *KK′ $\cup$ *KK″ and let $\leq$ be the least order relation on *KK such that $i \leq j$ if $\exists I$ [$j \in I$ & $(i,I) \in$ (KK′ $\cup$ KK″)]. Then KK′ | KK″ = { (k,K) | (k is $\leq$-minimal in *KK) & (K= {k′$\in$ *KK| $k \leq k'$ & $k \neq k'$}) }. The reader may verify readily that | operates on *KK* , is associative, commutative and idempotent.

III.A causal calculus of communicating systems

The goal of this section is to specify causal trees induced by CCS via the method of structural operational semantics. To this purpose, we embed CCS into a wider set of terms CCCS (a Causal Calculus of Communicating Systems). The transition system defined on CCCS is a direct extension of the original system for CCS: the definition of CCS is retrieved from the definition of CCCS by merely erasing all indications about causes.

Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ be the signature of CCS, defined hereafter with the usual meaning from the set of actions $\Lambda = \Delta \cup \bar{\Delta}$:
Σ_0={**nil**}, Σ_1= {μ | $\mu \in \Lambda \cup \{\tau\}$ } $\cup$ { $\backslash\alpha$ |$\alpha \in \Delta$ } $\cup$ { β/α | $\alpha,\beta \in \Delta$ }, Σ_2= {+, |}. Let REC(Σ,X) be the set of recursive Σ-terms over X (a set of variables) which satisfy the Greibach condition of well-guardedness (w.r.t. guarding operators μ). Terms have possible forms x ($\in$X) or **rec** x.t or $f(t_1,\ldots t_n)$ for $f \in \Sigma_n$. The CCS programs are the elements of CREC(Σ,X), the subset of the closed terms in REC(Σ,X).

Let $\Sigma' = \Sigma_2 \cup \{\backslash\alpha | \alpha \in \Delta\} \cup \{\beta/\alpha | \alpha,\beta \in \Delta\}$ and Σ''= {KK$\Rightarrow$ | KK$\in$ *KK* }. In CCCS derivations, operators KK$\Rightarrow$ will be used to indicate, at each step, the activating causes of all the active terms and subterms, given by backwards references to the past of the derivation. Provided a CCS term t is represented in the initial step as $(\varepsilon \Rightarrow t)$ where $\varepsilon = \{(1,\emptyset)\}$, the set of CCS terms is embedded in CCCS = TERM(Σ',Σ''(CREC(Σ,X))), the family of Σ' terms over generators (KK$\Rightarrow$t). A typical CCCS term, with causes KK_i attached to all outermost occurences of guarding operators and recursion symbols, is $((KK_1 \Rightarrow \lambda.\mathbf{nil}) \mid (KK_2 \Rightarrow \mathbf{rec}\ x.t))$. We assume henceforth that all operators (KK$\Rightarrow$) in Σ'' distribute over all operators in Σ', so that all CCCS terms are reducible to that canonical form. We also let δ operate on CCCS terms and distribute over all operators in Σ', and put $\delta(KK \Rightarrow t) = (\delta(KK) \Rightarrow t)$. The intuition behind δ is that an agent $\delta(KK \Rightarrow t)$ has lost one turn, hence its backwards pointers have been incremented.

The CCCS transition system, defined in SOS style [Plo81], is made of three species of labelled transitions between terms, namely $\{ \xrightarrow{\tau}, \xrightarrow{\lambda,KK}, \xrightarrow{\lambda,KK,KK'} \}$ where $\lambda \in \Lambda$ and KK,KK'$\in$ *KK* . In arrows of the third type, KK' represents the cause of some complementary action $\bar{\lambda}$, whereas arrows of the second type are reserved for non synchronized events. The axioms and rules of inference for transitions are the following.

Notations e,e',e" range over CCCS and t over CCS terms,

$\xrightarrow{z}$ stands for $\xrightarrow{\tau}$ or $\xrightarrow{\lambda,KK}$ or $\xrightarrow{\lambda,KK,KK'}$,

$z\backslash\alpha$ is defined if $z = \tau$ or $\lambda \notin \{\alpha,\bar{\alpha}\}$,

$\xrightarrow{z[\beta/\alpha]} = \xrightarrow{z}$ if $z=\tau$ else $\xrightarrow{\lambda[\beta/\alpha],KK}$ or $\xrightarrow{\lambda[\beta/\alpha],KK,KK'}$

letting $\lambda[\beta/\alpha]=\lambda$ if $\lambda \notin \{\alpha,\bar{\alpha}\}$, β if $\lambda=\alpha$, $\bar{\beta}$ if $\lambda=\bar{\alpha}$.

AXIOMS

$$(KK\Rightarrow \tau(t)) \xrightarrow{\tau} (KK\Rightarrow t) \qquad \textbf{tauact}$$

$$(KK\Rightarrow \lambda(t)) \xrightarrow{\lambda,KK} (\{(1,\delta(*KK))\}\Rightarrow t) \qquad \textbf{isolact}$$

$$(KK\Rightarrow \lambda(t)) \xrightarrow{\lambda,KK,KK'} ((KK|KK')\Rightarrow t) \qquad \textbf{comact}$$

RULES

$$\frac{e' \xrightarrow{z} e}{e'+e'' \xrightarrow{z} e} \;,\; \frac{e'' \xrightarrow{z} e}{e'+e'' \xrightarrow{z} e} \qquad \textbf{plus}$$

$$\frac{e \xrightarrow{z} e'}{e\backslash\alpha \xrightarrow{z} e'\backslash\alpha} \;,\quad z\backslash\alpha \text{ defined} \qquad \textbf{restrict}$$

$$\frac{e \xrightarrow{z} e'}{e[\beta/\alpha] \xrightarrow{z[\beta/\alpha]} e'[\beta\backslash\alpha]} \qquad \textbf{relab}$$

$$\frac{e' \xrightarrow{\tau} e}{e'|e'' \xrightarrow{\tau} e|e''} \;,\; \frac{e'' \xrightarrow{\tau} e}{e'|e'' \xrightarrow{\tau} e'|e} \qquad \textbf{partau}$$

$$\frac{e' \xrightarrow{\lambda,KK} e}{e'|e'' \xrightarrow{\lambda,KK} e|\delta(e'')} \;,\; \frac{e'' \xrightarrow{\lambda,KK} e}{e'|e'' \xrightarrow{\lambda,KK} \delta(e')|e} \qquad \textbf{parisol}$$

$$\frac{e' \xrightarrow{\lambda,KK,KK'} e}{e'|e'' \xrightarrow{\lambda,KK,KK'} e|e''} \;,\; \frac{e'' \xrightarrow{\lambda,KK,KK'} e}{e'|e'' \xrightarrow{\lambda,KK,KK'} e'|e} \qquad \textbf{parcom}$$

$$\frac{e' \xrightarrow{\lambda,KK,KK'} e''' \qquad e'' \xrightarrow{\overline{\lambda},KK',KK} e''''}{e'|e'' \xrightarrow{\tau} e'''|e''''} \qquad \textbf{synch}$$

$$\frac{(KK\Rightarrow (t\ [\textbf{rec}\ x.t/x])) \xrightarrow{z} e}{(KK\Rightarrow \textbf{rec}\ x.t) \xrightarrow{z} e} \qquad \textbf{rec}$$

Comments Axiom **tauact** means essentially that τ's are skipped by backwards references to the past (i.e. references to observable events in the trace of the derivation). Axiom **isolact** allows for the autonomous firing of a guard λ: the direct cause of activation of the residual term t is 1 (referring to λ), and the hereditary cause is incremented by 1). Axiom **comact** serves to fill in the premisses for **synch**. In this case, the action λ which appears in $\xrightarrow{\lambda,KK,KK'}$ is meant to be synchronized in further steps of deduction with some independent action $\bar{\lambda}$, hence it should be considered unobservable as regards backwards chaining. For that reason, the cause of t is simply the fusion of the respective causes KK of λ and KK' of $\bar{\lambda}$ with no additional delay. The same argument explains why **parcom** induces no updating of causes for the inactive process, to the contrary of **parisol** which contributes premisses to **synch**.

For e in CCCS, the operational meaning of e is the causal tree $[e]_{op}$ formed by unfolding from root e the transition system $\{e' \xrightarrow{z} e'' \mid z \in \mathbf{L} \cup \{\tau\}\}$. It may be observed that CCCS transitions $e \xrightarrow{\lambda,K} e'$ resp. $e \xrightarrow{\tau} e'$ are connected to CCS transitions $t \xrightarrow{\lambda} t'$ resp. $t \xrightarrow{\tau} t'$ by the following implications, where ψ:CCCS$\to$CCS is the 'cause erasing morphism' $\psi(KK \Rightarrow t) = t$:

$e \xrightarrow{\lambda,K} e' \Rightarrow \psi e \xrightarrow{\lambda} \psi e'$

resp. $e \xrightarrow{\tau} e' \Rightarrow \psi e \xrightarrow{\tau} \psi e'$,

$t \xrightarrow{\lambda} t' \Rightarrow (\psi e = t \Rightarrow \exists e' (\psi e' = t' \;\&\; \exists KK (e \xrightarrow{\lambda,KK} e')))$

resp. $t \xrightarrow{\tau} t' \Rightarrow (\psi e = t \Rightarrow \exists e' (\psi e' = t' \;\&\; e \xrightarrow{\tau} e'))$.

The last implication relies on the free choice of KK' in **comact**. It may also be observed that causal trees $[\varepsilon \Rightarrow t]_{op}$ do not coincide exactly with causal trees induced by LES's. For instance, the causal tree $[\varepsilon \Rightarrow A_1]_{op}$ derived from $A_1 = \alpha.\beta.$**nil** bears labels $(\alpha,\{(1,\emptyset)\})(\beta,\{(1,\{2\})\})$, slightly different from $(\alpha,\emptyset)(\beta,\{(1,\emptyset)\})$ in the causal tree induced by LES S_1 (see

II). The new labelling is better suited to the definition of algebras of causal trees, equipped with operations of external product such as KK**(1)**T.

Let us establish right now some useful properties of causal trees induced by CCCS expressions. A causal tree is *n-bounded* if for any sequence w.(λ,KK) labelling some initial path in the tree, $k\leq n+|w|$ for any $k\in$ *KK letting $|w|$ mean the length of w minus the number of τ's in w. A causal tree is *well formed* if (n,K')∈ KK' ⇒ K'= *(KK+n) for any sequence w.(λ,KK).w'.(λ',KK') labelling some initial path with $|w'|$= n-1.

Fact 1. $[e]_{op}$ *is always a bounded causal tree* : the upper bound for integers occuring in expressions is increased by 1 through derivations $\xrightarrow{\lambda,KK}$(owing to δ) and preserved through derivations $\xrightarrow{\tau}$(owing to the matching condition of causes in rule **synch** and in spite of **comact**). Hence, any cause KK in $[e]_{op}$ is finite.

Fact 2. $[e]_{op}$ *is always well formed* : in any derivation sequence e $\xrightarrow{w.(\lambda,KK).w'}$e' such that $|w'|$=n , (n,K')∈ KK' ⇒ K'= *(KK+n) for any operator KK'⇒ in e' . In view of **isolact** and **parisol,** this is true when w' is empty, and this remains true when the derivation is extended by one step, say in e $\xrightarrow{w.(\lambda,KK).w''}$e". If w"= w'.τ and the τ-transition was proved from **tauact,** then all operators KK"⇒ in e" already occured in e'. If w"= w'.τ and the τ-transition was proved from **synch,** then all operators KK"⇒ in e" either are of the form $(KK_1|KK_2)$⇒ with the KK_i already occuring in e' (from **comact**), or are inherited from e' (due to **parcom**). If w"= w'.(λ',KK') and KK"⇒ occurs in e" then KK"=(1,*KK'+1) or KK"=1+XX and XX⇒ occured in e' (from **parisol**).

Let $\mathbf{L}_+ = \{(\lambda,KK,KK') \mid \lambda \in \Lambda \ \& \ \{KK,KK'\} \subset \mathit{KK}\}$. Synchronization trees on $\{\tau\} \cup \mathbf{L} \cup \mathbf{L}_+$ are called *extended causal trees* ($\in CT_+$) and are considered from now on up to strong bisimulation in Milner's sense. Extended causal trees are equipped with operations of prefixing $(\tau., (\lambda,KK)., (\lambda,KK,KK').)$ and sum (+). The operation of sum is associative, commutative, idempotent, and has the empty sum *nil* as neutral element. For e in CCCS, let $[e]_{op+}$ be the extended causal tree formed by unfolding from root e the transition system $\{e' \xrightarrow{z} e'' \mid z \in \{\tau\} \cup \mathbf{L} \cup \mathbf{L}_+\}$. In fact, the extended causal tree $[e]_{op+}$ may be retrieved from the causal tree $[e]_{op}$, from which it differs only by redundant information. Redundancy is just needed for the inductive definition of transitions via structural inferences. Let us clarify a little more the relations between $[e]_{op}$ and $[e]_{op+}$. A connection between sets CT and CT_+ may be established by the following pair of functions EXP (expand): $CT \to CT_+$ and SHR (shrink): $CT_+ \to CT$. For T_+ in CT_+, $SHR(T_+)$ is T_+ minus all arrows $\xrightarrow{\lambda,KK,KK'}$ and corresponding subtrees. For KK in *KK* and T in CT, let $KK(\mathbf{1})T$ mean the substitution of KK for reference 1 in T (see IV for more precisions), then EXP is the unique function from CT to CT_+ satisfying the well-guarded recursive formula:

$$EXP((\sum_{i\in I} \tau.T_i)+(\sum_{j\in J} (\lambda_j,KK_j).T_j)) = (\sum_{i\in I} \tau.EXP(T_i)) + (\sum_{j\in J} (\lambda_j,KK_j).EXP(T_j)) +$$
$$(\sum_{j\in J} \sum_{KK} (\lambda_j,KK_j,KK).EXP((KK_j \mid KK)(\mathbf{1})T_j)$$

Now for any e∈CCCS and T∈CT, T=SHR(EXP(T)) obviously, ($[e]_{op} = SHR[e]_{op+}$, by definition, and ($[e]_{op+} = EXP[e]_{op}$) as will emerge from section VI. Most importantly, causal trees $[e]_{op}$ are *finitely branching*, whereas extended causal trees $[e]_{op+}$ are not. Although KK' is left free in axiom **comact**, which is the unique cause of infinite branching, this apparent freedom is reduced to naught by the matching condition of causes in **synch**.

*IV.**Combinatory operators on causes and causal trees***

We introduce via axiomatic definitions three indexed families of basic combinatory operators on causes and causal trees, subsequently used for upgrading the set CT of causal trees into a $(\Sigma\cup\dot{\Sigma}")$-algebra. Let us give some hints before we state precise definitions. Operators **(n)** are a direct generalization of operator **(1)** in section *III* and thus KK**(n)**T means substitution of cause KK+(n-1) for backwards reference n in tree T. Operators **[n]** serve to increment by 1 all backwards references greater than n in a tree. Finally, operators **<n>** set up the indirect cause n+1 wherever n occurs in a tree, meaning that n+1 is the unique cause of n.

In all definitions below, we let $T \equiv (\sum_{i\in I} (\lambda_i, KK_i).T_i) + (\sum_{j\in J} \tau.U_j)$, $k,n,m \in N_+$, $K \in \mathit{K}$ and $KK \in \mathit{KK}$. Index sets for sums are left implicit. All operators result in the empty sum when they are applied to the empty sum *(nil)*. As a general rule, summands with undefined labels are ignored, possibly inducing the empty sum. Undefined labels (λ, KK) may arise from fusion, which produces undefined causes from undefined causes. Fusion plays a crucial role in all statements, where it is needed to enforce axiom **is-cause** in the definition of *KK* (see II).

The family of operators **<n>** $(n\in N_+)$ is the unique family of unary operators on CT satisfying the well-guarded recursive formula:

$\mathbf{<n>}\ T \equiv ((\sum (\lambda_i, \mathbf{<n>}\ KK_i).\mathbf{<n+1>}\ T_i) + (\sum \tau.\mathbf{<n>}\ U_j))$

where $\mathbf{<n>} : \mathit{KK} \rightarrow \mathit{KK}$ is defined as follows:

$\mathbf{<n>}\{(k_1,K_1),\ldots,(k_p,K_p)\} = (\mathbf{<n>}(k_1,K_1) \mid \ldots \mid \mathbf{<n>}(k_p,K_p))$,

$\mathbf{<n>}(n,\emptyset) = (n,\{n+1\})$,

$\mathbf{<n>}(n\pm m,K) = \mathbf{<n>}(n\pm m,K)$ if $n\notin K$,

$\mathbf{<n>}(n\pm m,K\cup\{n\}) = (n\pm m,K\cup\{n,n+1\})$.

The reason why **<n>**(n,K) is **undefined** for non empty K, thus leading to equalities such as **<1>** $(\lambda,(1,\{2,3\})).T = nil$, will appear in the definition of guarding operators on causal trees (see axioms **lambda** and $\overset{\circ}{\textbf{lambda}}_+$ in section V).

The family of operators **[n]** ($n \in \overset{\circ}{N}_+$) is the unique family of unary operators on CT satisfying the well-guarded recursive formula:

$$[\mathbf{n}]\, T \equiv ((\sum (\lambda_i, [\mathbf{n}]\, KK_i).[\mathbf{n+1}]\, T_i) + (\sum \tau.[\mathbf{n}]\, U_j))$$

where **[n]** : *KK* → *KK* is defined as follows:

$[\mathbf{n}]\{(k_1,K_1),\ldots,(k_p,K_p)\} = ([\mathbf{n}](k_1,K_1)|\ldots|[\mathbf{n}](k_p,K_p))$,

$[\mathbf{n}](k,K) = ([\mathbf{n}]k,\{[\mathbf{n}]k' \mid k' \in K\})$,

$[\mathbf{n}](n-m) = n-m$, $[\mathbf{n}](n) = n+1$, $[\mathbf{n}](n+m) = n+m+1$.

The family of unary operators KK**(n)**, KK∈ *KK* and $n \in N_+$, is the unique family of operators on CT satisfying the well-guarded recursive formula:

$$KK(\mathbf{n})(T) = ((\sum (\lambda_i, KK(\mathbf{n})KK_i).\ KK(\mathbf{n+1})T_i) + (\sum \tau.\ KK(\mathbf{n})U_j)$$

where KK**(n)** : *KK* → *KK* is defined as follows:

$KK(\mathbf{n})\ \{(k_1,K_1),\ldots,(k_p,K_p)\} = (KK(\mathbf{n})(k_1,K_1)|\ldots|KK(\mathbf{n})(k_p,K_p))$,

$KK(\mathbf{n})(n-m,K) = \{(n-m,\ KK(\mathbf{n})K)\}$,

$KK(\mathbf{n})(n,K) = (KK+(n-1))$,

$KK(\mathbf{n})(n+m,K) = \{(n+m-1,\ KK(\mathbf{n})K\ )\}$,

$KK(\mathbf{n})K = \cup \{\ KK(\mathbf{n})k \mid k \in K\ \}$,

$KK(\mathbf{n})(n-m) = \{n-m\}$, $KK(\mathbf{n})(n) = {}^*KK + (n-1)$, $KK(\mathbf{n})(n+m) = \{n+m-1\}$.

The reason why n+m-1 occurs in place of n+m will appear in the definition of the asynchronous composition of causal trees (section *V*).

Let us state some algebraic laws satisfied by operators **(n)**,**[n]** and **<n>**. Each law is shaped as an indexed family of equations over causal trees. Since causal trees may be infinite, we cannot in general apply induction on trees to prove the validity of equations. However, we can still use a very general principle of recursive proofs, justified by the metric properties of trees:

in order to prove $F_n(T) = F'_n(T)$ simultaneously for all $n \in N_+$ and $T \in CT$, show that $F_n(T) = \sum_{j \in J} A_j.F_{nj}(T_j)$ and $F'_n(T) = \sum_{j \in J} A_j.F'_{nj}(T_j)$ for some $A_j \in (\{\tau\} \cup L)$, $n_j \in N_+$ and $T_j \in CT$, where $j \in J$.

A very similar principle will be used for proofs on extended causal trees. The following statements, where KK ranges over *KK* and T over CT, are all proved in this way (see appendix 1).

*Laws of **(n)***

α. KK**(n)**(KK′ | KK") = (KK**(n)**KK′ | KK**(n)**KK")

β. (KK′ | KK")**(n)**KK = (KK′**(n)**KK | KK"**(n)**KK)

γ. *if* (n,K′)∈ KK′ *implies* K′⊂ *(KK+n) *then*

KK**(n+p-1)**(KK′ **(p)** KK") = (KK**(n)**KK′)**(p)**(KK**(n+p)** KK") *and*

KK**(n+p-1)**(KK′ **(p)**T) = (KK**(n)**KK′)**(p)**(KK**(n+p)**T)

*Laws of **(n)** and **[m]***

δ. KK**(n)**(**[n]**KK′) = KK′ *and* KK**(n)**(**[n]**T) = T

ε. KK**(n+1)**(**[1]**KK′) = **[1]**(KK**(n)**KK′) *and* KK**(n+1)**(**[1]**T) = **[1]**(KK**(n)**T)

*Laws of **(n)** and **<m>***

ζ. KK**(n+1)** (**<n>**KK′) = (1,*KK+1)**(n)**KK′ *for* KK′ bounded by n, *and*

KK**(n+1)** (**<n>**T) = (1,*KK+1)**(n)**T *for* T an n-bounded causal tree.

The above laws extend obviously to the combinatory operators KK**(n)**,**[n]** and **<n>** : $CT_+ \rightarrow CT_+$ defined as follows:

KK**(n)**T_+ = EXP(KK**(n)** SHR(T_+)), **[n]**T_+ = EXP(**[n]** SHR(T_+)),

and **<n>**T_+ = EXP(**<n>** SHR(T_+)).

Laws γ and ε are carried on unchanged, whereas δ and ζ evolve into:

δ_+. $KK(\mathbf{n})([\mathbf{n}]T_+) = EXP(SHR(T_+))$,

ζ_+. $KK(\mathbf{n+1})(\langle\mathbf{n}\rangle T_+) = (1,{}^*KK+1)(\mathbf{n})T_+$ *for* $SHR(T_+)$ an n-bounded tree.

V. An algebra of causal trees

We turn now the set CT of causal trees into an interpretation for CCCS terms and take a step towards showing the full adequacy of the induced model. For that purpose we define an alternative, highly redundant, interpretation of CCCS terms in the set CT_+ of extended causal trees. Both interpretations are connected by the pair 'expand' and 'shrink'. We shall prove later on that CT_+ captures exactly the extended causal trees generated from CCS programs, whereby the adequacy of the (irredundant) model CT will be established.

The $(\Sigma\cup\Sigma'')$-algebra of causal trees is the unique model of the following axioms on carrier CT with + interpreted as sum of trees. For the sake of readability, pairs (k,K) in *K* are figured with sharp brackets, and set braces are usually omitted for singleton sets. These conventions hold in the remaining part of the paper, including appendices. We remind the reader that $\varepsilon=\langle 1,\emptyset\rangle$. We let $T \equiv (\Sigma_i\tau.T_i + \Sigma_j(\lambda_j,KK_j).T_j)$ in all axioms. Index sets are assumed disjoint.

init $KK\Rightarrow T = KK(\mathbf{1})T$

nil $\mathbf{nil} = nil$

tau $\tau(T) = \tau.T$

lambda $\lambda(T) = (\lambda,\varepsilon).\langle\mathbf{1}\rangle T$

relab $T[\beta/\alpha] = (\Sigma_i\tau.(T_i[\beta/\alpha]) + \Sigma_j(\lambda_j[\beta/\alpha],KK_j).(T_j[\beta/\alpha]))$

restrict $T\backslash\alpha = (\Sigma_i\tau.(T_i\backslash\alpha) + \sum_{(\lambda_j\backslash\alpha \text{ defined})} (\lambda_j,KK_j).(T_j\backslash\alpha)$

interleave *let* $U \equiv (\Sigma_m \tau.U_m + \Sigma_n (v_n, KK_n).U_n)$ *then* $T|U =$

$$\Sigma_i \tau.(T_i|U) + \Sigma_j (\lambda_j, KK_j).(T_j|\mathbf{[1]}U) +$$
$$\Sigma_m \tau.(T|U_m) + \Sigma_n (v_n, KK_n).(\mathbf{[1]}T|U_n) +$$
$$\Sigma_{\lambda_j = \overline{v}_n} \tau.(\ ((KK_j|KK_n)\mathbf{(1)}T_j)\ |\ ((KK_j|KK_n)\mathbf{(1)}U_n)\ )$$

Some comments are in order. The first axiom is used to replace backwards reference 1 by cause KK in tree T, and the second axiom interprets **nil** as the empty tree. The third and fourth axioms concern prefixing. When the prefixed action is τ, no further change is made in the prefixed tree since τ's are totally ignored in the coding of causes. Axiom **lambda** expresses the prefixing of a tree by a visible action. Causes must be updated so that, in the resulting tree, λ is the cause of all and only the actions which had empty cause ε in T. This is precisely reflected by letting **<n>**(n,K) be **undefined** for non empty K! Axiom **relab** relabels actions without changing the structure of the tree, while **restrict** prunes branches labelled by actions cut by restriction. Finally, **interleave** merges two trees by interleaving their actions (the first four summands) or by synchronizing them when complementary (the last summand). In the first case, if the action is τ, then T_i(resp.U_i) is composed with U(resp.T) left unchanged (see axiom **tau**). If the action is λ, then the causal tree U(resp.T) is 'delayed' by operator **[1]** , which increments by 1 all backwards references pointing outside U(resp.T). If complementary actions λ_j and v_n are synchonized, resulting in τ, the fusion of their respective causes KK_j,KK_n must pass down to their descendants. Moreover, both T_j and U_n must be 'advanced' by 1, since τ is invisible while λ_j and v_n were visible. Hence, the merge proceeds inductively with $(KK_j|KK_n)\mathbf{(1)}T_j$ and $(KK_j|KK_n)\mathbf{(1)}U_n$. This may be seen as the causal counterpart of Milner's expansion theorem.

The ($\Sigma \cup \Sigma''$)-algebra of extended causal trees is defined as the unique model of the following axioms on carrier CT_+, letting + be interpreted as sum of trees . We set $T \equiv (\Sigma_i \tau.T_i + \Sigma_j (\lambda_j, KK_j).T_j + \Sigma_k (\lambda_k, KK_k, KK'_k))$ everywhere, and assume disjoint index sets.

init$_+$ $KK \Rightarrow T = KK\mathbf{(1)}T$

nil$_+$ **nil** = *nil*

tau$_+$ $\tau(T) = \tau.T$

lambda$_+$ $\lambda(T) = (\lambda,\varepsilon).\mathbf{<1>}T + \Sigma_{KK} (\lambda,\varepsilon,KK).(\varepsilon | KK)\mathbf{(1)}(\mathbf{<1>}T)$

relab$_+$ $T[\beta/\alpha] = (\Sigma_i \tau.(T_i[\beta/\alpha]) + \Sigma_j (\lambda_j[\beta/\alpha], KK_j).(T_j[\beta/\alpha]) +$
$\Sigma_k (\lambda_k[\beta/\alpha], KK_k, KK'_k).(T_k[\beta/\alpha]))$

restrict$_+$ $T\backslash\alpha = (\Sigma_i \tau.(T_i\backslash\alpha) +$
$\sum_{(\lambda_j\backslash\alpha \text{ def.})} (\lambda_j, KK_j).(T_j\backslash\alpha) + \sum_{(\lambda_k\backslash\alpha \text{ def.})} (\lambda_k, KK_k, KK'_k).(T_k\backslash\alpha))$

interleave$_+$ *let* $U \equiv (\Sigma_l \tau.U_l + \Sigma_m (v_m, KK_m).U_m + \Sigma_n (v_n, KK_n, KK'_n).U_n)$ *then* $T|U =$
$\Sigma_i \tau.(T_i|U) + \Sigma_j (\lambda_j, KK_j).(T_j|\mathbf{[1]}U) + \Sigma_k (\lambda_k, KK_k, KK'_k).(T_k|U) +$
$\Sigma_l \tau.(T|U_l) + \Sigma_m (v_m, KK_m).(\mathbf{[1]}T|U_m) + \Sigma_n (v_n, KK_n, KK'_n).(T|U_n) +$
$\sum_{kRn} \tau.(T_k|U_n)$,

where: $kRn \Leftrightarrow (\lambda_k = \overline{v}_n)$ & $(KK_k = KK'_n)$ & $(KK'_k = KK_n)$.

Most comments made about the algebraic axioms for causal trees carry over the above axioms with obvious adaptations. The metamorphosis of the interleaving axiom, and specially the symmetry of causes set as a condition for synchronization, may help in better understanding the operational rules **comact** and **parcom**. Essential to the forthcoming results is the fact that well-formedness of causal trees is preserved by all operators in $\Sigma \cup \Sigma''$. This property is patent for all operators but guarding and parallel composition. For guarding, the trick is the undefinedness of **<n>**(n,K) for non empty K. In the case of parallel composition, there suffices to show the property for stringlike trees, i.e. for finite or infinite words on $\{\tau\}\cup\mathbf{L}$. If wt,wu are well-formed, then the interleaving axiom implies that $w\mu t'$,$w\mu u'$ are well-formed when $\mu(t'|u')$ is a summand of $(t|u)$. Hence we can make the following statement.

proposition 1. The algebra WCT (of well-formed causal trees) is a subalgebra of the algebra CT (of causal trees).

We are now ready to state the connection between algebras CT and CT_+.

proposition 2. Function EXP acts as a morphism of $(\Sigma\cup\Sigma")$-algebras from the algebra WCT to the algebra CT_+ (of extended causal trees).

This proposition is established by recursive proof (see appendix 2), using the combinatory law δ and a variant form of law α for causal trees, which we introduce below as proposition 3. First, we need a definition.

definition . Cause KK is *p-compatible* with tree T if, for any initial path $w.(\lambda,KK')$ in T, $<p+|w|,K'>\in KK' \Rightarrow K'\subset *(KK+p+|w|)$. (We remind the reader that τ's occuring in w do not contribute to the length |w| of w.)

proposition 3 . If KK is p-compatible with both T and U then:
$KK(\mathbf{p})(T|U) = (KK(\mathbf{p})T)|(KK(\mathbf{p})U)$.

Proposition 3 is in turn established by recursive proof (appendix 3), using the combinatory laws α,γ and ε.

Let the set WCT_+ (of well-formed extended causal trees) be defined as EXP(WCT), then WCT_+ is a subalgebra of CT_+ (proposition 2). Both algebras WCT and WCT_+ may be used as interpretations for CCCS, letting **rec**x.t be interpreted as the fixed point at x of the functional interpretation [t] of t. Existence and unicity of fixed points are guaranteed by the assumption of well-guardedness of recursive definitions (they induce contracting operators) and by the metric continuity of all operators in $\Sigma\cup\Sigma''$ (those operators are distance preserving, due to the form of the defining axioms, letting the distance between pairs of trees be the maximal distance between trees). Henceforth, we let $[e]_{ct}$, $[e]_{wct}$, $[e]_{ct+}$, and $[e]_{wct+}$ denote the respective interpretations of expression e in the algebras CT, WCT, CT_+, and WCT_+. Since WCT is a subalgebra of CT, $[e]_{ct}$ and $[e]_{wct}$ must coincide (by the uniqueness of fixed points in CT). Since WCT_+ is a subalgebra of CT_+, $[e]_{ct+}$ and $[e]_{wct+}$ must coincide (by the uniqueness of fixed points in CT_+). Finally, $[e]_{wct+}= EXP([e]_{wct})$ since EXP acts as a morphism of algebras between WCT and WCT_+ (and by the uniqueness of fixed points in WCT_+). We suggest as an exercise to verify the equality $[\mathbf{rec}x.\lambda(x)]_{ct}= (\lambda,\varepsilon).(\lambda,<1,\{2\}>).(\lambda,<1,\{2,3\}>)\ldots$

VI. Full adequacy

We establish here the equality between the operational meaning $[e]_{op+}$ and the algebraic meaning $[e]_{wct+}$ of an arbitrary CCCS expression . The full adequacy of the model CT, i.e. relation $[e]_{op}= [e]_{ct}$, then follows from equalities $[e]_{op}$=SHR$[e]_{op+}$ and $[e]_{ct}$=SHR$[e]_{ct+}$. Furthermore, we get for free a proof of relation $[e]_{op+}$=EXP$[e]_{op}$ (from $[e]_{wct+}$=EXP(SHR$[e]_{wct+}$)) and another proof of the boundedness and well-formedness of causal trees $[e]_{op}$.

A quick comparison between the algebraic axioms for CT_+ and the logical axioms set for CCCS transitions shows that all we have to establish, for proving full adequacy of WCT_+, is the following series of propositions which imply the existence of normal forms $\Sigma_i(z_i.[e_i]_{ct+})$ for extended causal trees $[e]_{ct+}$. Let us introduce tree-transitions $[e]_{ct+} \xrightarrow{z} [e']_{ct+}$ if $([e]_{ct+}=\Sigma_i(z_i.[e_i]_{ct+})) \Rightarrow (\exists i)(z=z_i \& [e']_{ct+}=[e_i]_{ct+})$, then indeed it is clear from propositions 4 to 7 below that both transitions $e \xrightarrow{z} e'$ between terms and transitions $[e]_{ct+} \xrightarrow{z} [e']_{ct+}$ between trees obey the rules stated for CCCS in section III.

proposition 4. $\forall(\mathbf{rec}x.t)\in \mathrm{CREC}(\Sigma,X)$ $\forall KK\in \mathit{KK}$:

$[KK\Rightarrow \mathbf{rec}x.t]_{wct+} = [KK\Rightarrow t[\mathbf{rec}x.t/x]]_{wct+}$.

proposition 5. $\forall t\in \mathrm{CREC}(\Sigma,X)$ $\forall KK\in \mathit{KK}$:

$[KK\Rightarrow\tau(t)]_{wct+} = \tau.[KK\Rightarrow t]_{wct+}$ and

$[KK\Rightarrow\lambda(t)]_{wct+}= (\lambda,KK).[<1,{*}KK+1>\Rightarrow t]_{wct+} + \Sigma_{KK'}(\lambda,KK,KK').[(KK|KK')\Rightarrow t]_{wct+}$.

proposition 6. $\forall t_i\in \mathrm{CREC}(\Sigma,X)$ $\forall KK\in \mathit{KK}$:

$[KK\Rightarrow(t_1+t_2)]_{wct+} = [KK\Rightarrow t_1]_{wct+} + [KK\Rightarrow t_2]_{wct+}$,

$[KK\Rightarrow(t\backslash\alpha)]_{wct+} = [(KK\Rightarrow t)\backslash\alpha]_{wct+}$,

$[KK\Rightarrow(t[\beta/\alpha])]_{wct+} = [(KK\Rightarrow t)[\beta/\alpha]]_{wct+}$,

$[KK\Rightarrow(t_1|t_2)]_{wct+} = [KK\Rightarrow t_1]_{wct+} \mid [KK\Rightarrow t_2]_{wct+}$.

proposition 7. $\forall e\in$ CCCS: $\mathbf{[1]}[e]_{wct+} = [\delta e]_{wct+}$.

The above propositions (proved in appendix 4) rely upon the combinatory law ζ, proposition 3 and the following lemma (proved ibidem).

lemma 1. $\forall t\in \mathrm{CREC}(\Sigma,X)$: $[t]_{wct}$ is 1-bounded and equal to $[\varepsilon\Rightarrow t]_{wct}$.

The section may be summed up in the following theorem:

The algebraic models CT and WCT are fully adequate for CCCS.

VII. Weak causal bisimulation and observational congruence.

At the present time, we dispose of an algebraic model of causal trees, in which programs are identified when their associated transition systems are strongly bisimilar. We noted above that causal trees are synchronization trees on a modified alphabet $\{\tau\} \cup \mathbf{L}$, where non-τ symbols bear indication of their observable causes. Up to now, we contrived to let the unobservable action τ play the same role here as in Milner's CCS. Our motivation for preserving validity of Milner's results on synchronization trees was to make easier the study of weak bisimulation between programs in the framework of causal trees. Let us adapt the definition of observational congruence to that modified setting.

For $n\geq 0$ and $s=(\lambda_1,KK_1)\dots(\lambda_n,KK_n)$ in $(\Lambda x KK)^*$, let s be the binary relation $e \overset{s}{\Longrightarrow} e'$ if $e(\xrightarrow{\tau})^* \xrightarrow{(\lambda_1,KK_1)} (\xrightarrow{\tau})^* \dots \xrightarrow{(\lambda_n,KK_n)} (\xrightarrow{\tau})^* e'$. *Weak causal equivalence* $\sim$ is the largest bisimulation on CCCS compatible with relations $\overset{s}{\Longrightarrow}$. Equivalently, $\sim$ is the largest symmetric relation on CCCS such that $e_1 \sim e_2$ entails the following for any s in $(\Lambda x KK)^*$: if $e_1 \overset{s}{\Longrightarrow} e'_1$ then $(\exists e'_2)$ $e_2 \overset{s}{\Longrightarrow} e'_2$ & $e'_1 \sim e'_2$. By way of extension, *observational equivalence* on CCS programs is the equivalence $t \sim t'$ iff $(\varepsilon \Rightarrow t) \sim (\varepsilon \Rightarrow t')$, and *observational congruence* on CCS programs is the largest equivalence $\sim^c$ included in $\sim$ and preserved by all combinators of CCS, including recursion.

Following [Mil-85], let us introduce equivalence $\sim^+$ over CCS programs by setting $t \sim^+ t'$ iff $t+t'' \sim t'+t''$ for all programs t". Similarly, let $\sim^+$ be the relation over CCCS expressions defined as $e \sim^+ e'$ iff $e+e'' \sim^+ e'+e''$ for all expressions e". Then $\sim^c$ is included in $\sim^+$, and the next statement is proved exactly like in Milner's paper.

proposition 8. The following are equivalent for all programs t_1, t_2:

(1) $t_1 \sim^+ t_2$

(2) $(\varepsilon \Rightarrow t_1) \sim^+ (\varepsilon \Rightarrow t_2)$

(3) For all μ in $(\Lambda x KK) \cup \{\tau\}$

(i) if $(\varepsilon \Rightarrow t_1) \xrightarrow{\mu} e_1$ then, for some e_2, $(\varepsilon \Rightarrow t_2) \Longrightarrow \xrightarrow{\mu} \Longrightarrow e_2$ and $e_1 \sim e_2$

(ii) if $(\varepsilon \Rightarrow t_2) \xrightarrow{\mu} e_2$ then, for some e_1, $(\varepsilon \Rightarrow t_1) \Longrightarrow \xrightarrow{\mu} \Longrightarrow e_1$ and $e_1 \sim e_2$

The next two propositions show that equivalence $\sim^+$ is in fact a congruence and coincides therefore with observational congruence $\sim^c$. (Proofs of propositions may be found in appendix 5.)

proposition 9. The equivalence $\sim^+$ is preserved by all combinators in Σ.

proposition 10. The equivalence $\sim^+$ is preserved by all program contexts, including recursive contexts.

By way of conclusion, let us mention the availability of a complete system of equational axioms for observational congruence over finite i.e. non recursive CCS programs t identified with CCCS expressions $(\varepsilon \Rightarrow t)$. The considered axioms are equalities between CCCS expressions. The first seven axioms are copies of Milner's axioms A1-A4 for strong bisimulation and τ-laws A5-A7, where label μ ranges over $(\Lambda x KK) \cup \{\tau\}$. The remaining axioms are **init, nil, tau, lambda, relab, restrict, interleave** and the defining equations for combinators **<n>, [n]**, and **(n)** operating on causal trees specified by sum expressions. Let us recall Milner's axioms A1-A7.

A1	$x+(y+z) = (x+y)+z$	**A2**	$x+y = y+x$
A3	$x+x = x$	**A4**	$x+\mathbf{nil} = x$
A5	$x+\tau.x = \tau.x$	**A6**	$\mu.(x+\tau.y) = \mu.(x+\tau.y)+\mu.y$
A7	$\mu.\tau.y = \mu.y$		

The correctness and completeness of the resultant axiomatisation emerge from the following remarks:

(i) A1-A4 are valid because strong bisimulation implies equivalence $\sim^{+}$,

(ii) axioms **init,nil,tau,lambda,relab,restrict,interleave** may be used to derive from any non recursive CCCS expression an equivalent expression on combinators **nil**, $\tau.$, (λ,KK). and $+$,

(iii) since $\sim^{+}$ is a copy of Milner's relation $\sim^{+}$, the τ-axioms A5-A7 are valid (letting $\mu\in (\Lambda x KK) \cup \{\tau\}$),

(iv) the normal forms of [Mil-85] may therefore be used in a remake of the original proof of completeness for Hennessy-Milner's axioms.

The reader may for instance derive from the above axioms relation $\alpha(\beta.\gamma.\mathbf{nil}\,|\,\bar{\beta}.\mathbf{nil})\backslash\beta \sim^{c} \alpha.\gamma.\mathbf{nil}$, but some form of mechanical help would certainly be welcomed in less immediate situations.

VIII. Summary

We have introduced Causal Trees, defined up to strong bisimulation, which extend Synchronization Trees in that they record observable causes of observable events. This makes Causal Trees a suitable semantic domain for concurrency, describing the full interplay between nondeterminism and partial ordering in a concurrent language.

In order to examplify Causal Trees, we have shown their informal derivation from Labelled Event Structures. The other example dealt with in full detail is Milner's CCS for which a new operational semantics has been defined, respecting causality and yet consistent with the original one. Then, Causal Trees have been equipped through axiomatic definition with a structure of algebra providing an interpretation for CCS. The axiom for parallel composition may be read as an 'expansion theorem' reducing parallelism to a combination of interleaving and nondeterminism, but the considered form of *interleaving preserves causality* . The interpretation for CCS has been proved fully abstract w.r.t. the operational semantics. Finally, the notion of *causal observational congruence* has been introduced in a completely standard way (from weak bisimulation), and a complete axiomatization of congruence has been given for finitary CCS. To our knowledge, no complete axiomatization was ever defined before for a language including communication. We do not know for the moment whether causal congruence is preserved by action refinement.

Remarkably enough, we used here after simple adaptation all the stuff developped for CCS in the classical interleaving approach. Similar adaptations might probably work for extended forms of testing equivalences coping with causality.

References

[AB84] Austry,D. and Boudol,G. Algèbre de Processus et Synchronisation, *Theoret. Comput. Sci.* **30**,1 (1984) 91-131.

[BC88a] Boudol,G. and Castellani,I. Concurrency and Atomicity, *Theoret. Comput. Sci.* **59**,1-2 (1988) 25-84.

[BC88b] Boudol,G. and Castellani,I. Permutation of Transitions: an Event Structure Semantics for CCS and SCCS, to appear in [REX878]

[BHR84] Brookes,S.D., Hoare C.A.R. and Roscoe A.D. A Theory of Communicating Sequential Processes, *Journal of A.C.M.*, **31**,3 (1984) 560-599

[BK84] Bergstra,J.A. and Klop,J.W. Process Algebra for Synchronous Communication, *Info. and Control* **61** (1984) 109-137

[BS87] Broy,M. and Steicher,T. Views of Distributed Systems, Proc. Advanced School on Math. Models for the Semantics of Parallelism, Springer-Verlag L.N.C.S. **280** (1987) 114-143

[CH87] Castellani,I. and Hennessy,M. Distributed Bisimulations, Research Report 5/87, Computer Science Department, University of Sussex (1987)

[Dar80] Darondeau,Ph. Processus non séquentiels et leurs observations en univers non centralisé, Proc. Int. Symp. on Programming, Springer-Verlag L.N.C.S. **83** (1980) 92-107

[DG87] Darondeau,Ph. and Gamatié,B. Modelling Infinitary Behaviours of Communicating Systems, INRIA-Rennes, Research Report 749 (1987)

[DDM87a] Degano,P., De Nicola,R. and Montanari,U. Observational Equivalences for Concurrency Models, in *Formal Decrription of Programming Concepts III* (M.Wirsing ed.), North-Holland (1987) 105-132

[DDM87b] Degano,P., De Nicola,R. and Montanari,U. A Distributed Operational Semantics for CCS based on Condition/Events Systems, Nota interna B4-21, IEI (1987). To appear in *Acta Informatica*

[DDM88a] Degano,P., De Nicola,R. and Montanari,U. Partial Ordering Semantics for CCS, Internal Report 88-3, Dipartimento di Informatica, Univ. Pisa (1988)

[DDM88b] Degano,P., De Nicola,R. and Montanari,U. Partial Ordering Description of Nondeterministic Concurrent Systems, to appear in [REX88]

[DDM88c] Degano,P., De Nicola,R. and Montanari,U. On the Consistency of Truly Concurrent Operational and Denotational Semantics, Proc. LICS'88, Edinburgh (1988)

[DGM87] Degano, P., Gorrieri,R. and Marchetti,S. An Exercise in Concurrency: A CSP Process as a Condition/Event System, Proc. 8th European Workshop on Applications and Theory of Petri Nets, Zaragoza (1987)

[DM87a] Degano,P. and Montanari,U. A Model of Distributed Systems based on Graph Rewriting, *Journal of A.C.M.*, **34** (1987) 411-449

[DM87b] Degano,P. and Montanari,U. Concurrent Histories: A Basis for Observing Distributed Systems, *J.C.S.S.*, **34** (1987) 442-461

[GV87] van Glabbeek,R. and Vaandrager,F. Petri Net Models for Algebraic Models of Concurrency, Proc. PARLE Conf., Springer-Verlag L.N.C.S. **259** (1987) 224-242

[Hen88] Hennessy,M. *An Algebraic Theory of Processes*, MIT-Press (1988)

[Lam78] Lamport,L. Time, Clocks and the Ordering of Events in a Distributed System, *Comm. of A.C.M.*, **12** (1978) 558-564

[Maz77] Mazurkiewicz,A. Concurrent Program Schemas and their Interpretation , Proc. Aarhus Workshop on Verification of Parallel Programs (1977)

[Mil80] Milner,R. *A Calculus of Communicating Systems,* Springer-Verlag L.N.C.S. **92** (1980)
[Mil85] Milner,R. Lectures on a Calculus for Communicating Systems, NATO ASI Series, Vol. F14, Springer-Verlag (1985) 205-228
[MS81] Montanari,U. and Simonelli,C. On distinguishing between concurrency and nondeterminism, Proc. Ecole de Printemps on Concurrency and Petri Nets, Colleville sur mer (1980)
[NPW81] Nielsen,M., Plotkin,G. and Winskel,G. Petri Nets, Event Structures and Domains, Part 1, *Theoret. Comput. Sci.* **13** (1981) 85-108
[Niv82] Nivat,M. Behaviours of Processes and Synchronized Systems of Processes, in *Theoretical Foundations of Programming Methodology* , Dordrecht Reidel (1982) 473-550
[Old87] Olderog,E.R. Operational Petri Net Semantics for CCSP, in *Advances in Petri Nets 1987* , Springer-Verlag L.N.C.S. **266** (1987) 196-223
[Pet80] Petri,C.A. Concurrency, in *Net Theory and Applications,* Springer-Verlag L.N.C.S. **84** (1980) 1-19
[Pra86] Pratt V.R. Modelling Concurrency with Partial Orders, *Intern. Journal of Parallel Programming* **15** (1986) 33-71
[Plo81] Plotkin,G. A Structural Approach to Operational Semantics, DAIMI Report FN-19, Department of Computer Science, Aarhus (1981)
[Rei85] Reisig,W. *Petri Nets: an Introduction,* EATCS Monographs on Theoretical Computer Science, Springer-Verlag (1985)
[REX88] Proc. REX School/Workshop on Linear Time, Branching Time and Partial Orders in Logic and Models for Concurrency, Springer-Verlag L.N.C.S., to appear
[Wink80] Winkowski,J. Behaviours of Concurrent Systems, *Theoret. Comput. Sci.* **12** (1980) 39-60
[Win82] Winskel,G. Events in Computation, Ph.D. Thesis, Univ. of Edinburgh, CST-10-80 (1980)
[Win87] Winskel,G. Event Structures, in *Advances in Petri Nets 1987* , Springer-Verlag L.N.C.S. **266** (1987) 196-223

Appendix 1. Proofs for laws on combinatory operators.

Relations α to ζ are patently valid on causal trees $T \in CT$, if they are valid on causes $KK \in \mathit{KK}$. Causal trees are therefore ignored in the subsequent proofs, where operators and relations on integers are extended elementwise to work on sets $K \in \mathit{K}$ and pairs $(k,K) \in \mathit{KK}$. In the following, we let $\pi_i(KK_i)$ be an abbreviation for $KK_1 | \ldots | KK_m$.

----------------------**Laws of (n)**--------------------

proof of α.

In view of the algebraic properties of fusion, relation α is a straightforward consequence of relation $KK\,\mathbf{(n)}\,(\pi_i \langle k_i, K_i \rangle) = \pi_i(KK\,\mathbf{(n)}\langle k_i, K_i \rangle)$ which is true by definition of **(n)**.

proof of β.

In the above notations, relation β is a straightforward consequence of relation $(\pi_i \langle k_i, K_i \rangle)\,\mathbf{(n)}\;KK' = \pi_i(\langle k_i, K_i \rangle\mathbf{(n)}\;KK')$, which we establish hereafter. By virtue of law α, we can restrict ourselves to the simple case when $KK' = \langle k', K' \rangle$ and thus show $(\pi_i \langle k_i, K_i \rangle)\,\mathbf{(n)}\langle k', K' \rangle = \pi_i(\langle k_i, K_i \rangle\mathbf{(n)}\langle k', K' \rangle)$. If $k' = n$, this amounts to $\pi_i KK_i = \pi_i KK_i$ with $KK_i = \langle k_i, K_i \rangle + (n-1)$. If $n \in K'$, the above amounts to the equality $\langle k', [{*}\pi_i \langle k_i, K_i \rangle]\,\mathbf{(n)} K' \rangle = \pi_i \langle k', [{*}\langle k_i, K_i \rangle]\,\mathbf{(n)} K' \rangle$, whose truth emerges from the obvious relation $(\cup_i K'_i)\,\mathbf{(n)}\;K' = \cup_i (K'_i \mathbf{(n)}\;K')$ by setting $K'_i = \{k_i\} \cup K_i$. If $k' \neq n$ and $n \notin K'$ the relation of interest is of the form $(KK = \pi_i KK)$ and therefore holds by idempotence of fusion. All cases have been dealt with.

proof of γ.

Owing to the distributive laws α and β, there suffices to establish γ in the restricted case when KK'= {<k',K'>} and KK"= {<k",K">}. So, let L and R stand respectively for the left and right members of the following equality, to prove:

KK**(n+p-1)**(<k',K'>**(p)**<k",K">) = (KK**(n)**<k',K'>) **(p)** (KK**(n+p)**<k",K">).

We proceed by case to case verification. Two obvious lemmas are used intensively:

ijk+ (KK**(i)**j)+k = KK**(i+k)**(j+k)

ijk- (KK**(i)**j)-k = KK**(i-k)**(j-k)

case k"<p

L= <k",KK**(n+p-1)**(KK' **(p)**K")>

R= <k",((KK**(n)**KK') **(p)** (KK**(n+p)**K")>

Let us prove KK**(n+p-1)**(KK' **(p)**q) = (KK**(n)**KK') **(p)** (KK**(n+p)**q) for q in N_+.

We proceed by case analysis.

q<p : the relation to prove amounts to q=q,

q=p : we get KK**(n+p-1)**(*KK'**+**(p-1)) = *(KK**(n)**KK')**+**(p-1).

By ijk+ the left member is equal to (KK**(n)***KK')**+**(p-1).

Now, *(KK**(n)**KK') = (KK**(n)***KK') by the hypothesis

<n,K'>∈KK' ⇒ K'⊂ *(KK+n).

p<q<n+p : the relation to prove amounts to q-1=q-1,

q=n+p : it amounts to *KK**+**(n+p-2)=(*KK**+**(n+p-1))-1,

q>n+p : the relation reduces to q-2=q-2.

Hence the proof is complete for the case k"<p.

case k"=p

L= KK**(n+p-1)**<k'+p-1,K'+p-1>

R= (KK**(n)**<k',K'>)**+**(p-1)

If k′=n then L=KK**+**(n+p-2)=(KK**+**(n-1))**+**(p-1)=R.

If k′<n or k′>n then accordingly: L= <k′+p-1,X> or <k′+p-2,X>, and R= <k′+p-1,Y> or <k′+p-2,Y> for X= KK**(n+p-1)**(K′**+**(p-1)) and Y= (KK**(n)**K′)**+**(p-1). Now, X=Y by ijk+.

case p<k"<n+p

L= <k"-1,KK**(n+p-1)**(K"-1)>

R= <k"-1,(KK**(n+p)**K")-1>

and L=R by ijk-.

case k"=n+p

L= KK**+**(n+p-2)

R= KK**+**(n+p-1)-1

whence L=R.

case k">n+p

The relation to prove is <k"-2,K"-2> = <k"-2,K"-2>.

Hence the proof of validity of law γ is complete.

----------------------Laws of (n) and [m]------------------------

proof of δ.

In view of the idempotence of fusion and by laws α,β there suffices to establish the simplified relation <k,K>**(n)**(**[n]**<k′,K′>) = <k′,K′>. We proceed by case analysis. If k′<n then **[n]**<k′,K′> = <k′,**[n]**K′>, and since $n \notin$ **[n]**K′, the left member of the relation evaluates to <k′,K"> for K" = {i| $i \in$ K′ & i<n} $\cup$ {i+1-1| $i \in$ K′ & $i \geq$ n}, hence K" = K′. If k′=n then **[n]**<k′,K′> = <k′+1,K′**+**1> and both members of the relation evaluate to the pair <k′,K′>. Finally, if k′>n then <k,K>**(n)**(**[n]**<k′,K′>) = <k′,K′>+1-1 which is still <k′,K′>.

proof of ε.

For the same reasons as above, there suffices to establish the simplified relation $<k,K>\mathbf{(n+1)}(\mathbf{[1]}<k',K'>) = \mathbf{[1]}(<k,K>\mathbf{(n)}<k',K'>)$. Now, $\mathbf{[1]}<k',K'> = <k'+1,K'+1>$, and we proceed by case analysis. If $k'=n$ the left member of the relation amounts to $<k,K>\mathbf{+}n$ and the right member to $<k,K>\mathbf{+}(n-1)\mathbf{+1}$ hence they are equal. If $k'\neq n$ then both members evaluate to $<k",K">$ for $k"=k'$ or $k"=k'+1$ according to $k'>n$ or $k'<n$, and $K"= \{i+1 \mid i\in K' \ \& \ i<n\} \cup \{i+n \mid n\in K' \ \& \ (i=k \vee i\in K)\} \cup \{i \mid i\in K' \ \& \ i>n+1\}$.

----------------------Laws of (n) and <m>-----------------------

proof of ζ.

In view of laws α,β and since $(1,*(\pi_i KK_i)+1) = \pi_i(1,*KK_i+1)$, it suffices to show the simplified relation $<k,K>\mathbf{(n+1)}(\mathbf{<n>}<k',K'>) = <1,\{k+1\}\cup(K+1)>\mathbf{(n)}<k',K'>$ under the hypotheses $k'\leq n$ and $K'\leq n$ ($i\leq n \ \ \forall i\in K'$). The case by case verification follows.

If $k'=n$ then $K'=\emptyset$ by the hypothesis $K'\leq n$, and the leftmost expression evaluates to $<k,K>\mathbf{(n+1)}<n,\{n+1\}>$ and therefrom to $<n,\{n+k\}\cup(K+n)>$, while the rightmost expression evaluates to $<1,\{1+k\}\cup(K+1)>\mathbf{+}(n-1)$ which amounts clearly to $<n,\{n+k\}\cup(K+n)>$. If $n\in K'$ then $k'<n$ and the relation to establish reduces to $<k,K>\mathbf{(n+1)}<k',K'\cup\{n+1\}>= <k',K">$ for $K"$ defined as $\{i \mid i\in K' \& i\neq n\}\cup\{n\}\cup\{k+n\}\cup(K+n)$. Since $n\in K'$, $K"$ is equal to $K'\cup\{k+n\}\cup(K+n)$, whence $<k,K>\mathbf{(n+1)}<k',K'\cup\{n+1\}> = <k',K">$. Finally, if $k'<n$ and $K'<n$, relation δ reduces to $<k,K>\mathbf{(n+1)}<k',K'> = <1,\{k+1\}\cup(K+1)>\mathbf{(n)}<k',K'>$, which is nothing but $<k',K'>=<k',K'>$.

Appendix 2. Proof of proposition 2

The proposition results from seven elementary propositions, established below. In all statements T,U $\in$ WCT with $T \equiv (\Sigma_i \tau.T_i + \Sigma_j (\lambda_j, KK_j) T_j)$ and $U \equiv (\Sigma_m \tau.U_m + \Sigma_n (\nu_n, KK_n) U_n)$.

1. **KK⇒EXP(T) = EXP(KK⇒T)**

   this reads as KK**(1)**EXP(T)=EXP(KK**(1)**T), which holds since

   KK**(1)**EXP(T) = EXP(KK**(1)**SHR(EXP(T))) and SHR(EXP(T)) = T.

2. **nil = nil**

3. **τ(EXP(T)) = EXP(τ(T))**

   τ(EXP(T))= τ.EXP(T)= EXP(τ.T)= EXP(τ(T))

4. **λ(EXP(T)) = EXP(λ(T))**

   λ(EXP(T)) = (λ,ε).**<1>**EXP(T) + Σ_{KK}(λ,ε,KK).(ε|KK) **(1)** (**<1>**EXP(T)),

   while EXP(λ(T)) = EXP((λ,ε).**<1>**T) =

   (λ,ε).EXP(**<1>**T) + Σ_{KK}(λ,ε,KK).EXP((ε|KK) **(1)** (**<1>**T)),

   now by definition **<1>**EXP(T) = EXP(**<1>**T) and (ε|KK) **(1)** (EXP(**<1>**T)) =

   EXP((ε|KK) **(1)** (SHR(EXP(**<1>**T)))) = EXP((ε|KK) **(1)** (**<1>**T)).

5. **(EXP(T))[β/α] = EXP(T[β/α])**

   (EXP(T))[β/α] = $\Sigma_i\tau$.(EXP(T_i))[β/α] + Σ_j(λ_j[β/α],KK_j).(EXP(T_j))[β/α] +

   $\Sigma_j\Sigma_{KK}$(λ_j[β/α],KK_j,KK).(EXP((KK_j|KK) **(1)** T_j))[β/α],

   EXP(T[β/α]) = $\Sigma_i\tau$.EXP(T_i[β/α]) + Σ_j(λ_j[β/α],KK_j).EXP(T_j[β/α]) +

   $\Sigma_j\Sigma_{KK}$(λ_j[β/α],KK_j,KK).EXP((KK_j|KK) **(1)** (T_j[β/α])),

   and the desired result is proved by recursive proof from relation:

   (KK_j|KK) **(1)** (T_j[β/α])=((KK_j|KK) **(1)** T_j) [β/α],

   which in turn may be easily established by recursive proof on CT.

6. **(EXP(T))\\$\alpha$ = EXP(T\\$\alpha$)**

   is proved exactly like proposition 5.

7. **EXP(T|U) = EXP(T)|EXP(U)**

By definition, EXP(T|U) =

$$\Sigma_i \tau.\mathrm{EXP}(T_i|U) + \Sigma_j(\lambda_j,KK_j).\mathrm{EXP}(T_j|[\mathbf{1}]U) +$$
$$\Sigma_m \tau.\mathrm{EXP}(T|U_m) + \Sigma_n(v_n,KK_n).\mathrm{EXP}([\mathbf{1}]T|U_n) +$$
$$\Sigma_j\Sigma_{KK}(\lambda_j,KK_j,KK).\mathrm{EXP}((KK_j|KK)\,(\mathbf{1})\,(T_j|[\mathbf{1}]U)) +$$
$$\Sigma_n\Sigma_{KK}(v_n,KK_n,KK).\mathrm{EXP}((KK_n|KK)\,(\mathbf{1})\,([\mathbf{1}]T|U_n)) +$$
$$\sum_{\lambda_j=\overline{v}_n} \tau.\mathrm{EXP}(\ (KK_j|KK_n)\,(\mathbf{1})T_j\ \ |\ \ (KK_j|KK_n)\,(\mathbf{1})U_n\ ),$$

while EXP(T)|EXP(U) =

$$\Sigma_i \tau.(\mathrm{EXP}(T_i)|\mathrm{EXP}(U)) + \Sigma_j(\lambda_j,KK_j).(\mathrm{EXP}(T_j)|[\mathbf{1}]\mathrm{EXP}(U)) +$$
$$\Sigma_m \tau.(\mathrm{EXP}(T)|\mathrm{EXP}(U_m)) + \Sigma_n(v_n,KK_n).([\mathbf{1}]\mathrm{EXP}(T)|\mathrm{EXP}(U_n)) +$$
$$\Sigma_j\Sigma_{KK}(\lambda_j,KK_j,KK).(\mathrm{EXP}((KK_j|KK)\,(\mathbf{1})T_j)|\mathrm{EXP}(U)) +$$
$$\Sigma_n\Sigma_{KK}(v_n,KK_n,KK).(\mathrm{EXP}(T)|\mathrm{EXP}((KK_n|KK)\,(\mathbf{1})U_n)) +$$
$$\sum_{\lambda_j=\overline{v}_n} \tau.(\ \mathrm{EXP}((KK_j|KK_n)\,(\mathbf{1})T_j)\ \ |\ \ \mathrm{EXP}((KK_j|KK_n)\,(\mathbf{1})U_n)\ ).$$

By hypotheses, T and a fortiori $(\lambda_j,KK_j).T_j$ are well-formed , hence KK_j and a fortiori $(KK_j|KK)$ are 1-compatible with T_j. Since any cause is n-compatible with **[n]**U for any tree U, $(KK_j|KK)$ is 1-compatible with **[1]**U. We are in the right conditions to apply proposition 3, which leads to the equality $((KK_j|KK)\,(\mathbf{1})\,(T_j|[\mathbf{1}]U))$ = $((KK_j|KK)\,(\mathbf{1})T_j)\,|\,(KK_j|KK)\,(\mathbf{1})\,([\mathbf{1}]U))$. The validity of the proposition follows by recursive proof from the defining equality **[1]**EXP(T)=EXP(**[1]**T) and the combinatory law δ, since the trees $(KK_j|KK)\,(\mathbf{1})T_j$ and $(KK_n|KK)\,(\mathbf{1})U_n$ are well formed.

Appendix 3. Proof of proposition 3

Let $T = (\Sigma_i \tau . T_i + \Sigma_j (\lambda_j, KK_j) . T_j)$ and $U = (\Sigma_m \tau . U_m + \Sigma_n (v_n, KK_n) . U_n)$.

Then $KK(\mathbf{p}) T = (\Sigma_i \tau . KK(\mathbf{p}) T_i + \Sigma_j (\lambda_j, KK(\mathbf{p}) KK_j) . KK(\mathbf{p+1}) T_j)$,

and $KK(\mathbf{p}) U = (\Sigma_m \tau . KK(\mathbf{p}) U_m + \Sigma_n (v_n, KK(\mathbf{p}) KK_n) . KK(\mathbf{p+1}) U_n)$.

By definition, $KK(\mathbf{p}) T \mid KK(\mathbf{p}) U =$

$$\Sigma_i \tau . (KK(\mathbf{p}) T_i \mid KK(\mathbf{p}) U) + \Sigma_j (\lambda_j, KK(\mathbf{p}) KK_j) . (KK(\mathbf{p+1}) T_j \mid \mathbf{[1]} (KK(\mathbf{p}) U)) +$$

$$\Sigma_m \tau . (KK(\mathbf{p}) T \mid KK(\mathbf{p}) U_m) + \Sigma_n (v_n, KK(\mathbf{p}) KK_n) . (\mathbf{[1]} (KK(\mathbf{p}) T) \mid KK(\mathbf{p+1}) U_n) +$$

$$\sum_{\lambda_j = \overline{v}_n} \tau . [(KK(\mathbf{p}) (KK_j \mid KK_n)) \mathbf{(1)} (KK(\mathbf{p+1}) T_j) \mid (KK(\mathbf{p}) (KK_j \mid KK_n)) \mathbf{(1)} (KK(\mathbf{p+1}) U_n)]$$

while $KK(\mathbf{p}) (T \mid U) =$

$$\Sigma_i \tau . KK(\mathbf{p}) (T_i \mid U) + \Sigma_j (\lambda_j, KK(\mathbf{p}) KK_j) . KK(\mathbf{p+1}) (T_j \mid \mathbf{[1]} U) +$$

$$\Sigma_m \tau . KK(\mathbf{p}) (T \mid U_m) + \Sigma_n (v_n, KK(\mathbf{p}) KK_n) . KK(\mathbf{p+1}) (\mathbf{[1]} T \mid U_n) +$$

$$\sum_{\lambda_j = \overline{v}_n} \tau . KK(\mathbf{p}) [((KK_j \mid KK_n) \mathbf{(1)} T_j) \mid ((KK_j \mid KK_n) \mathbf{(1)} U_n)].$$

From the hypotheses, KK is p-compatible with both T and U.

Hence $K' \subset *(KK+p)$ for any pair (p, K') occuring in $(KK_j \mid KK_n)$.

We are in the right conditions to apply law γ, which yields the equality $KK(\mathbf{p}) ((KK_j \mid KK_n) \mathbf{(1)} T_j) = (KK(\mathbf{p}) (KK_j \mid KK_n)) \mathbf{(1)} (KK(\mathbf{p+1}) T_j)$.

On the other hand, $KK(\mathbf{p+1}) (\mathbf{[1]} T) = \mathbf{[1]} (KK(\mathbf{p}) T)$ by direct application of ε.

The proposition follows by recursive proof, seeing that p-compatibility (of KK) with T and U entails (p+1)-compatibility with **[1]**T and **[1]**U, p-compatibility with T_i and U_m, (p+1)-compatibility with T_j and U_n, and finally p-compatibility with $(KK_j \mid KK_n) \mathbf{(1)} T_j$ and $(KK_j \mid KK_n) \mathbf{(1)} U_n$.

Appendix 4. Complements to section VI.

We establish here propositions 4 to 7 and the underlying lemma 1, thereby completing the proof of full adequacy of WCT_+ .

proof of lemma 1. The second assertion in the lemma is an immediate consequence of the main statement, which expresses 1-boundedness of $[t]_{wct}$ for $t \in CREC(\Sigma, X)$. Suppose for contradiction that t has been chosen with minimal complexity in the set of terms for which $[t]_{wct}$ is not 1-bounded . From the algebraic axioms of CT, t must be of the form $\lambda(t')$ or of the form $t'|t''$. If $t= \lambda(t')$ and $[t']_{wct}$ is 1-bounded then $[t]_{wct}$ is 1-bounded (by **lambda**). If $t= t'|t''$ and both $[t']_{wct}$, $[t'']_{wct}$ are 1-bounded, then $[t]_{wct}$ is 1-bounded (by lemma 2 below). Hence t cannot be minimal and we have got the desired contradiction.

lemma 2. Let causal trees T',T" be n-bounded then T'|T" is n-bounded. (This property comes directly from the interleaving axiom for CT.)

proof of proposition 4. Since **rec** is interpreted as a fixed point combinator, $[\mathbf{rec}x.t]_{wct+} = [t[\mathbf{rec}x.t/x]]_{wct+}$, whence $[KK\Rightarrow \mathbf{rec}x.t]_{wct+} = KK(\mathbf{1})[\mathbf{rec}x.t]_{wct+} = KK(\mathbf{1})[t[\mathbf{rec}x.t/x]]_{wct+} = [KK\Rightarrow t[\mathbf{rec}x.t/x]]_{wct+}$, by applying axiom **init**.

proof of proposition 5.

The first half of the proposition is straightforward:

$[KK\Rightarrow\tau(t)]_{wct+} = KK(\mathbf{1})[\tau(t)]_{wct+} = EXP(KK(\mathbf{1})[\tau(t)]_{wct}) = \tau.EXP(KK(\mathbf{1})[t]_{wct}) = \tau.(KK(\mathbf{1})[t]_{wct+}) = \tau.[KK\Rightarrow t]_{wct+}$.

The second half of the proposition is more involved.

$[KK\Rightarrow\lambda(t)]_{wct+} = KK(\mathbf{1})[\lambda(t)]_{wct+} = EXP(KK(\mathbf{1})[\lambda(t)]_{wct}) = EXP(KK(\mathbf{1})[(\lambda,\varepsilon).\mathbf{<1>}[t]_{wct}] = EXP[(\lambda,KK).(KK(\mathbf{2})(\mathbf{<1>}[t]_{wct}))]$.

Lemma 1 enables the combinatory law ζ which in this case amounts to:

$(KK\mathbf{(2)}(\mathbf{<1>}[t]_{wct})) = <1,*KK+1>\mathbf{(1)}[t]_{wct}$.

Now, $((KK|KK')\mathbf{(1)}(<1,*KK+1>\mathbf{(1)}[t]_{wct})) = ((KK|KK')\mathbf{(1)}([t]_{wct}))$

because the causal tree $[t]_{wct}$ is 1-bounded.

(The reader may for instance verify the equality:

$(KK|KK')\mathbf{(2)}(<1,*KK+1>\mathbf{(2)}<1,\{2\}>) = (KK|KK')\mathbf{(2)}<1,\{2\}>$.)

As a consequence, $[KK\Rightarrow\lambda(t)]_{wct+} = EXP[(\lambda,KK).(<1,*KK+1>\mathbf{(1)}[t]_{wct})] =$

$(\lambda,KK).[<1,*KK+1>\mathbf{(1)}EXP[t]_{wct}] + \Sigma_{KK'}(\lambda,KK,KK').EXP((KK|KK')\mathbf{(1)}([t]_{wct})) =$

$(\lambda,KK).[<1,*KK+1>\Rightarrow t]_{wct+} + \Sigma_{KK'}(\lambda,KK,KK').[(KK|KK')\Rightarrow t]_{wct+}$.

proof of proposition 6. Only the last assertion deserves a proof.

$[KK\Rightarrow(t_1|t_2)]_{wct+} = KK\mathbf{(1)}[t_1|t_2]_{wct+} = EXP(KK\mathbf{(1)}([t_1]_{wct}|[t_2]_{wct})$.

Since causal trees $[t_i]_{wct}$ are 1-bounded, they are 1-compatible with any cause KK, and proposition 3 yields us the equality:

$KK\mathbf{(1)}[t_1|t_2]_{wct} = (KK\mathbf{(1)}[t_1]_{wct})|(KK\mathbf{(1)}[t_2]_{wct})$.

Hence, $[KK\Rightarrow(t_1|t_2)]_{wct+} = EXP[(KK\mathbf{(1)}[t_1]_{wct})|(KK\mathbf{(1)}[t_2]_{wct})] =$

$EXP(KK\mathbf{(1)}[t_1]_{wct}) \mid EXP(KK\mathbf{(1)}[t_2]_{wct})$ by proposition 2.

We are finished because the last line is the definition of:

$[KK\Rightarrow t_1]_{wct+} \mid [KK\Rightarrow t_2]_{wct+}$.

proof of proposition 7. If e is an expression of the form $(KK\Rightarrow t)$ then $\delta(KK\Rightarrow t)=[(KK+1)\Rightarrow t]$ and the relation to prove is $\mathbf{[1]}(KK\mathbf{(1)}[t]_{wct+}) = ((KK+1)\mathbf{(1)}[t]_{wct+})$. Now $\mathbf{[1]}(KK\mathbf{(1)}[t]_{wct}) = ((KK+1)\mathbf{(1)}[t]_{wct})$ because $[t]_{wct}$ is 1-bounded (lemma 1), and the above relation follows from symmetric application of EXP. There remains to check the case when e is an expression of the compound form e'|e". We reason by induction on expressions. By definition: $\mathbf{[1]}[e'|e"]_{wct+} = EXP(\mathbf{[1]}([e']_{wct}|[e"]_{wct}))$

$= EXP(\mathbf{[1]}([e']_{wct})|\mathbf{[1]}([e"]_{wct}))$ -by lemma 3 below-

$= \mathrm{EXP}([\mathbf{1}]([e']_{wct}))|\mathrm{EXP}([\mathbf{1}]([e'']_{wct}))$ -by proposition 2-

$= ([\mathbf{1}]([e']_{wct+})|[\mathbf{1}]([e'']_{wct+}))$ -by definition-

$= ([\delta e']_{wct+}|[\delta e'']_{wct+})$ -by the inductive hypothesis-

$= [\delta e'|\delta e'']_{wct+} = [\delta e]_{wct+}$.

lemma 3. $[\mathbf{1}](T'|T'') = ([\mathbf{1}]T'|[\mathbf{1}]T'')$ for 1-bounded causal trees. (This property comes directly from the interleaving axiom for CT.)

Appendix 5. Complements to section VII

We intend to prove that relation $\sim^+$ is a congruence over CCS programs. The alternative characterisation of $\sim^+$ offered in proposition 8 may be used for that purpose, because full adequacy of the model CT entails correspondence between CCCS transitions $e \xrightarrow{\mu} e'$ and tree-transitions $[e]_{wct} \xrightarrow{\mu} [e']_{wct}$ (for all μ in $\mathbf{L} \cup \{\tau\}$).

In the sequel, we let $\sim^+$ denote the equivalence over CT defined as $T_1 \sim^+ T_2$ iff for all $\mu \in \mathbf{L} \cup \{\tau\}$:

(i) if $T_1 \xrightarrow{\mu} T'_1$ then for some T'_2, $T_2 \Rightarrow \xrightarrow{\mu} \Rightarrow T'_2$ and $T'_1 \sim T'_2$,

(ii) if $T_2 \xrightarrow{\mu} T'_2$ then for some T'_1, $T_1 \Rightarrow \xrightarrow{\mu} \Rightarrow T'_1$ and $T'_1 \sim T'_2$,

where $\sim$ is the largest bisimulation on causal trees compatible with tree-transitions s, $s \in (\Lambda \times KK)^*$.

proof of proposition 9.

$t \sim^+ t'$ iff $(\varepsilon \Rightarrow t) =^+ (\varepsilon \Rightarrow t')$ -by proposition 8-

iff $[\varepsilon \Rightarrow t]_{wct} \sim^+ [\varepsilon \Rightarrow t']_{wct}$ - by proposition 8 and full adequacy of WCT-

iff $[t]_{wct} \sim^+ [t']_{wct}$ - by lemma 1-

In order to prove the proposition, it suffices therefore to show that relation $\sim^+$ is a congruence over the algebra WCT of well-formed causal trees. From axioms **tau,lambda,relab,restrict** it is patent that relation $\sim^+$ is preserved by operators $\tau, \lambda, \backslash\alpha, [\alpha/\beta]$. The case for parallel composition is a little more complex and may be dealt with as follows. Let relation R over WCT be defined as T R T′ if $T=T_1|T_2$ and $T'=T'_1|T'_2$ with $T_1 \sim T'_1$ and $T_2 \sim T'_2$. Owing to implications $T\sim T' \Rightarrow$ **[1]**T $\sim$ **[1]**T′ and $T\sim T' \Rightarrow$ KK**(1)**T $\sim$ KK**(1)**T′, the interleaving axiom shows that relation R is a bisimulation w.r.t. tree-transitions $\xrightarrow{\mu}$ (remind that *nil* is a neutral element for parallel composition). Relation R is thus included in the largest bisimulation $\sim$; compatibility relations $T\sim^+T' \Rightarrow (T|U)\sim^+(T'|U)$ and $T\sim^+T' \Rightarrow (U|T)\sim^+(U|T')$ then follow as easy consequences of the interleaving axiom. Assume for instance $T\sim^+T'$ and $T|U \xrightarrow{\mu} V$ then, from **interleave**, $T'|U \Longrightarrow \xrightarrow{\mu} \Longrightarrow V'$ and V R V′ (and symmetrically on U).

proof of proposition 10.

For exactly the same reasons as in the proof of the above proposition 9, it suffices to show that relation $\sim^+$ is preserved by all elements in the fixed point clone of operators on WCT defined on top of Σ. Let us establish $T\sim^+T' \Rightarrow$ **recx**.f(x,T)$\sim^+$**recx**.f(x,T′) for non recursive operators f(x,y) defined by usual Σ-terms, well-guarded in x. The case for recursively defined operators f(x,y) is similar: Bekic's principle for simultaneous fixed points is valid in WCT since any system of well-guarded equations has a unique solution in that algebra. Let relation S over WCT be defined as V S V′ if V = g(**recx**.f(x,T),$T_1,\ldots T_n$) for some g in the $(\Sigma\cup\Sigma")$-clone of operators on WCT and V′ = g(**recx**.f(x,T′),$T'_1,\ldots T'_n$) for corresponding trees T'_i satisfying $T_i\sim^+T'_i$. Since V S V′ $\Rightarrow$ (**[1]**V S **[1]**V′) and V S V′ $\Rightarrow$ ((KK**(1)**V) S (KK**(1)**V′)), an adaptation of the proof for proposition 9 shows that relation S is a weak bisimulation over WCT, hence

S is included in $\sim$. In view of the algebraic axioms for WCT, the implication $T\sim^{+}T' \Rightarrow$ **rec**x.f(x,T)$\sim^{+}$**rec**x.f(x,T') then follows from the assumption of well-guardedness of f(x,y) in x.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique