Automata for Partially Ordered and Partially Reliable Connections.

J.Fanchon. fanchon@laas.fr

April 18, 2001

LAAS-CNRS, 7 avenue du Colonel Roche, 31077 Toulouse Cedex 4 - France

Abstract

We present a formal model for the specification of a transport protocol with partially ordered and partially reliable data delivery. An abstract syntax is defined to specify order and reliability constraints in data flows, and is also used to specify the particular automata which recognise exactly the sets of deliverable streams. This new class of automata extends the Asynchronous Cellular Automata for Pomsets with state variables such as to recognize languages of partial orders labelled by (tuples of) integers. This extension is used for the specification of a multimedia data transport protocol with a new type of highly scalable QoS parameters.

Keywords: automata, concurrency, multimedia streams, partial orders, protocols, QoS.

1 Introduction

Partial order and partial reliability connections (POC) have been introduced to optimise multimedia streams from the point of view of time and space requirements ([1],[2]). While UDP and TCP protocols are implementations of respectively no-order no-reliability and total-order total-reliability connections, POC are intended to cover a complete range of specifications between these two extrema. POC take into account the fact that insides a data stream:

1. The use of some item by the receiver may depend or not on the reception of another item. Desequencing of data items in delivery, following some specification is a standard behaviour of POC.

2. Some items may not be lost during the transport while some others may be, when specific conditions are fullfilled. Losses of data items following some specification is also a standard behaviour of POC.

We present a formal framework which includes the specification of multimedia data streams with partial order delivery and partial reliability, and the specification of the main part of the protocol handling them, by means of specific automata recognizing languages of labelled partial orders. One of the problems which we face when dealing with partial order and reliability is the complexity of the specification and the validation of the protocols. The model presented here proposes a compositional specification of the streams, and a compositional specification and validation of the corresponding protocols. It is based on an abstract syntax for streams, with operators to specify the partial order, like sequence, parallel composition, iteration, etc.., and operators to specify the admitable losses, as defined below. A term of the specification language is then given two compositional semantics as

- two languages (sets) of labelled partially ordered sets called the emitted and the deliverable languages. The first represents the emitted stream and the second the set of its substreams (subsets of data) which satisfy the reliability conditions.

- an automaton which recognises exactly the deliverable language, and which is compositional w.r.t. the stream structure.

The terms are sorted as finite patterns, elementary streams and synchronised streams. The specification language is structured as follows:

- **A synchronised stream** is defined as a **set of parallel elementary streams** with **synchronisation constraints** between them.
- **Each elementary stream is structured as an iterated finite pattern**, i.e. is composed of consecutive occurrences (or instances) of a particular finite labelled partial order called pattern (see MPEG and its frame structure).
- **Synchronisations may occur between two elementary streams**, and they constraint an order of delivery between patterns of the two streams, following the specification of a **synchronisation expression.**
- **Losses can be specified on each elementary stream** in two ways:

1. A **'mandatory' minimal "sub-pattern"** which must be delivered at each iteration of the pattern.
2. Maximal **losses per windows of patterns** on a specific subset of labels can be defined. This mechanism extends to patterns and labels the losses on windows of fixed length.

The automata defined in this paper have extended states (with variables or counters), together with communication and composition features. Two classes of automata corresponding to the different stream sorts, are defined : to handle patterns and elementary streams, an Elementary Automaton, EA for short, reads labelled inputs from the environment, and may "accept" them depending on a condition on their internal "local" state and the label of the input. To handle synchronised streams, a Synchronised Automaton, SA, is composed of a set of elementary automata, each one reads its labelled inputs from a specific channel and accepts them depending not only on its local state and the label of the input, but also on the local states of any subset of the other components. The difference between SA components and EA is thus an extended testing capacity for the former, which implements the synchronisation constraints between elementary streams. Different operators like synchronised merge and iteration are defined on such automata to model the operations of the signature. In particular some partial reliability operators are defined by means of automata compositions. Synchronised Automata are very close to Asynchronous Cellular Automata for Pomsets as defined in [7][12]. The main difference consists in the presence of counters. They can be viewed as the application to ACA's of the extensions as defined in [5].

The three types of terms, i.e. lossy patterns, elementary streams and synchronised streams, are denoted by T_{lp}, T_{es} and T_{st}. The stream semantics associates to each term t two languages of labelled posets, the emitted language denoted $EL(t)$ and the deliverable language $DL(t)$, the latter being composed of the sub-partial orders of the former which satisfy the reliability constraints. Different sets of labels are used for the items belonging to these languages, depending if we consider terms in T_{lp}, T_{es} or T_{st}. The labels are tuples composed by elements of a set A of SDU (i.e. single data units) labels and elements of $\aleph$, the set of integers. The set A may model data types, like audio, video, etc.. or have a more precise meaning, like in MPEG frames. The items of lossy patterns are labelled by pairs $(a, i) \in A \times \aleph$. A labelled poset "belonging" to the emitted or deliverable languages of a pattern is then a tuple $(D, \leq, l \times \nu)$ where D is the emitted data set, $l : D \longrightarrow A$ is a typing and $\nu : D \longrightarrow \aleph$ is a numbering map which identifies uniquely each item inside the pattern. Throughout the paper, by abuse of notation, the expression $l \times \nu$ for label mappings denotes the mapping $l \times \nu : D \longrightarrow A \times \aleph$ such that $(l \times \nu)(e) = (l(e), \nu(e))$. The items of elementary streams belong to some pattern with the labelling defined above, and moreover have an iteration number: they are labelled on triples in $A \times \aleph \times \aleph$. Composed streams are composed

of elementary streams each one being mapped on a specific channel. They are thus labelled on $C \times A \times \aleph \times \aleph$, where C is a set of channel names.

Let T be an algebra in T_{lp}, T_{es} or T_{st} , and Σ the corresponding set of labels, the automata semantics is defined as a map $Aut : T \longrightarrow AUT(\Sigma)$ where $AUT(\Sigma)$ is the class of automata associated to T, with inputs labelled by Σ. The semantic maps are the following: (as defined in section 2, $PL(\Sigma)$ denotes the set of languages of posets labelled on Σ) :

$DL : T_{lp} \longrightarrow PL(A \times \aleph)$ and $Aut : T_{lp} \longrightarrow EA(A \times \aleph)$

$DL : T_{es} \longrightarrow PL(A \times \aleph \times \aleph)$ and $Aut : T_{es} \longrightarrow EA(A \times \aleph \times \aleph)$

$DL : T_{st} \longrightarrow PL(C \times A \times \aleph \times \aleph)$ and $Aut : T_{st} \longrightarrow SA(C \times A \times \aleph \times \aleph)$.

The coherence property states that the deliverable language $DL(t)$ of a term t is the language $L(Aut(t))$ accepted by the automaton $Aut(t)$. More precisely $L(Aut(t))$ is the set of linearizations of $DL(t)$, and we have $\forall t \in T$, $Lin(DL(t)) = L(Aut(t))$. This is due to the interleaving definition of the runs of the automata. A concurrent semantics can also be defined but the presented one is sufficient, in that it reflects the adequacy of the automaton with the partial order delivery constraints.

In the implementations, the protocols have more complex behaviours than the automata. First they do not just accept (or reject) the inputs, but can also bufferise those which may be accepted later. Furthermore a retransmission mechanism must be implemented in case of loss of some minimal sub-patterns elements. More generally a mechanism of (non)acknowledge must be used due to the unreliable nature of the media, induced by the hypothesis of partial reliability. Nevertheless, they can be implemented in a way which preserves the properties ensured by the model.

Related Works: The Multimedia streams with partial order and partial reliable delivery with timing constraints composed of a single iterated pattern have been modelled by (Hierarchical) Time Stream Petri Nets ([6],[15]). On another hand, Finite State Machines Communicating through lossy channels have been defined [4], but besides parallel composition, no other algebraic structure, and in particular no possibility to define acceptable losses, have been developped.

2 Partial orders.

Let Σ be a set of labels, a $\Sigma-$**labelled poset** is a tuple $(D, \leq, \lambda)$ such that D is a set, $\leq \subseteq D \times D$ is an order relation, and $\lambda : D \longrightarrow \Sigma$ is the labelling map. The set of $\Sigma-$labelled posets is denoted $P(\Sigma)$. The **partial word or pomset** $[D, \leq, \lambda]$ is the isomorphism class of $(D, \leq, \lambda)$ as labelled poset, i.e. the set of all $(D', \leq', \lambda') \in P(\Sigma)$ such that there is an order isomorphism $\varphi : (D, \leq) \longrightarrow (D', \leq')$ s.t. $\lambda = \lambda' \circ \varphi$. The **set of partial words** labelled on Σ is denoted $PW(\Sigma)$. The set of languages of $\Sigma-$labelled posets, i.e. the set of subsets of $P(\Sigma)$, is denoted $PL(\Sigma)$. All the languages considered here are closed for isomorphism, in particular the emitted and deliverable languages $EL(t)$ and $DL(t)$.

The **sub-word** relation $\triangleright$ on $\Sigma-$labelled posets is defined by $(D, \leq, \lambda) \triangleright (D', \leq', \lambda') \Longleftrightarrow D' \subseteq D \bigwedge \leq' = \leq /D' \bigwedge \lambda' = \lambda/D'$.The relation $\triangleright$ is also defined on partial words by $\rho \triangleright \rho' \Longleftrightarrow \exists (D, \leq, \lambda) \in \rho, (D', \leq', \lambda') \in \rho' : (D, \leq, \lambda) \triangleright (D', \leq', \lambda')$In both cases sets of subwords are defined by $Sub(\sigma) = \{\sigma', \sigma \triangleright \sigma'\}$ or $\sigma \triangleright \sigma' \Longleftrightarrow \sigma' \in Sub(\sigma)$.

Let $\sigma = (D, \leq, \lambda) \in P(\Sigma)$ a labelled poset, the set $Ext(\sigma)$ of **order extensions** of σ is $Ext(\sigma) = \{(D, \leq', \lambda) \in P(\Sigma), \leq \subseteq \leq'\}$.The set of **order extensions** of a poset language $L \subseteq P(\Sigma)$ is $Ext(L) = \cup_{\sigma \in L} Ext(\sigma)$. The set of **linear extensions** $Lin(\sigma)$ is the subset of $Ext(\sigma)$ composed of totally ordered extensions of σ.

The sequential and parallel compositions of two $\Sigma-$labelled posets $s_1 = (D_1, \leq_1, l_1)$ and $s_2 = (D_2, \leq_2, l_2)$, are defined iff $D_1 \cap D_2 = \emptyset$, by $s_1 \bullet s_2 = (D_1 \uplus D_2, \leq_1 \cup \leq_2 \cup (D_1 \times D_2), l_1 \cup l_2)$ and $s_1 \parallel s_2 = (D_1 \uplus D_2, \leq_1 \cup \leq_2, l_1 \cup l_2)$

We denote the set of naturals by $\aleph$ and by $\leq_n$ (resp.$<_n$) the usual total (resp. strict) ordering on $\aleph$. Let be $n \in \aleph$, we note $[n]$ the initial segment of integers $[n] = \{1, ..n\}$.

3 Patterns and Elementary streams.

3.1 Numbered patterns.

Numbered patterns are defined by the syntax

$$\boxed{m := (a,i) \mid m \parallel m \mid m.m}$$

where $a \in A$, $i \in \aleph$.

Let us denote $I(m)$ the set of integers occurring in the numbered pattern m, defined inductively by $I(a,i) = \{i\}$ and $I(m \parallel m') = I(m.m') = I(m) \cup I(m')$. **Well formed** numbered patterns are defined as follows: for any $a \in A$, $i \in \aleph$, (a,i) is well formed and $m * m'$ is well formed ($*$ standing for $\parallel$ and .) if m and m' are well formed and $I(m) \cap I(m') = \emptyset$. We furthermore require for $m.m'$ that $i \in I(m) \bigwedge j \in I(m') \Longrightarrow i <_n j$. The set of **well formed** numbered patterns is denoted T_{np}

To each well formed numbered pattern m we associate a $A \times \aleph-$labelled poset $P(m) = (I(m), \leq_m, l \times Id_{I(m)})$ defined inductively as follows:

$P(a,i) = (\{i\}, \emptyset, \{(i,(a,i))\})$, $P(m.m') = P(m) \bullet P(m')$ and $P(m \parallel m') = P(m) \parallel P(m')$ where the operators $\parallel, \bullet$ on labelled posets are defined above .

3.1.1 Emitted and deliverable languages.

The emitted language $EL(m) \subseteq P(A \times \aleph)$ of a numbered pattern m, is the set of $A \times \aleph-$labelled posets $(D, \leq, l \times \nu)$ isomorphic to $P(m)$: $EL(m) = [P(m)] = \{(D, \leq, l \times \nu) \in P(A \times \aleph), [D, \leq, l \times \nu] = [P(m)]\}$. The **deliverable language** $DL(m) \subseteq P(A \times \aleph)$ is the **set of sub-words** of its emitted language: $DL(m) = Sub(EL(m)) = \{\sigma' \in P(A \times \aleph), \exists \sigma \in EL(m) : \sigma \triangleright \sigma'\}$

3.2 Lossy patterns

The syntax of lossy patterns is

$$\boxed{\rho := m \mid \rho / \triangleright m}$$

where $m \in T_{np}$ is a numbered pattern. The expression $\rho / \triangleright m$ defines m as a minimal unloosable subpattern of ρ. We denote by T_{lp} the set of lossy pattern terms. The **root pattern** $R(\rho) \in T_{np}$ of a lossy pattern $\rho \in T_{lp}$ is defined by $R(\rho) = \rho$ if $\rho \in T_{np}$ and $R(\rho / \triangleright m) = R(\rho)$, otherwise. A lossy pattern $\rho / \triangleright m$ is **well-formed** iff the root pattern $R(\rho)$ is well-formed and m is a subword of $R(\rho)$.

3.2.1 Emitted and Deliverable languages.

The **emitted language** $EL(\rho)$ is equal to the one of the root $R(\rho)$, $EL(\rho) = EL(R(\rho))$.

The **deliverable language** $DL(\rho / \triangleright m)$ is the set of elements of $DL(\rho)$ which have $[P(m)]$ as sub-word: $DL(\rho / \triangleright m) = \{(D, \leq, l \times \nu) \in DL(\rho), [D, \leq, l \times \nu] \triangleright [P(m)]\}$

Due to the property $DL((m / \triangleright m_1) \triangleright m_2) = DL((m / \triangleright m_2) \triangleright m_1)$ we can use the notation $m / \triangleright \{m_j,\ j = 1..n\} = ((..(m / \triangleright m_1) / \triangleright m_2)../ \triangleright m_n$

Proposition 1: $DL(w / \triangleright \{m_j,\ j = 1..n\}) = \bigcap_{j \in [n]} DL(w / \triangleright m_j)$

3.3 Elementary automata for patterns.

We present first the definition of elementary automata, to be used in the following.

3.3.1 Automata definition.

Let Σ be an alphabet, a $\Sigma-$elementary automaton is a tuple

$$\Gamma = (Q, \Sigma_\gamma, T, q_0, F)$$

Q is the **set of states**, $q_0 \in Q$ is the initial state and $F \subseteq Q$ a set of final states, $\Sigma_\gamma \subseteq \Sigma$ is the **set of input**s, $T \subseteq G \times O$ is a set of **abstract transitions** , where G is a set of **boolean guards** on pairs (state, input), i.e. mappings $g : Q \times \Sigma_\gamma \longrightarrow \{0,1\}$ and O is a set of "**operations**" , i.e. mappings $O \subseteq Q \times \Sigma_\gamma \longrightarrow Q$ yielding a new state from an old one and an input. The class of $\Sigma-$elementary automata is denoted $EA(\Sigma)$.

The **transition relation** $\longrightarrow \subseteq Q \times \Sigma_\gamma \times Q$, denoted $q \overset{l}{\longrightarrow} q'$ for $(q,l,q') \in \longrightarrow$, is defined by $q \overset{l}{\longrightarrow} q' \iff \exists (g,o) \in T : g(q,l) = 1 \bigwedge q' = o(q,l)$. The relation $\Longrightarrow \subseteq Q \times \Sigma_\gamma^* \times Q$ is the smallest one containing $\overset{u}{\longrightarrow}$ and such that $(q \overset{u}{\Longrightarrow} q'' \bigwedge q'' \overset{l}{\longrightarrow} q') \Longrightarrow q \overset{ul}{\Longrightarrow} q'$. The (sequential) **language** $L(\Gamma) \subseteq \Sigma^*$ of an automaton Γ is the set $L(\Gamma) = \{u \in \Sigma^*, \exists q \in F : q_0 \overset{u}{\Longrightarrow} q \}$

Let $\Gamma_i = (Q_i, \Sigma_i, T_i, q_{i0}, F_i)$ $i = 1,2$, be two automata, then the **synchronised product** $\Gamma = \Gamma_1 \parallel \Gamma_2 = (Q, \Sigma, T, q_0, F)$ is defined by $Q = Q_1 \times Q_2$, with $q_0 = (q_{10}, q_{20})$, $F = F_1 \times F_2$, $\Sigma = \Sigma_1 \cup \Sigma_2$, T is the set of pairs (g,o),$g : Q \times \Sigma \longrightarrow \{0,1\}$,$o : Q \times \Sigma \longrightarrow Q$ corresponding to one of the cases below:

1. $\exists (g_1,o_1) \in T_1$, $\exists (g_2,o_2) \in T_2$, $g((q_1,q_2),l) = (l \in \Sigma_1 \cap \Sigma_2) \bigwedge g_1(q_1,l) \bigwedge g_2(q_2,l)$ and $o((q_1,q_2),l) = (o_1(q_1,l), o_2(q_2,l))$
2. $\exists (g_1,o_1) \in T_1$, $g((q_1,q_2),l) = (l \in \Sigma_1 - \Sigma_2) \bigwedge g_1(q_1,l)$ and $o((q_1,q_2),l) = (o_1(q_1,l), q_2)$
3. Like 2) exchanging indices.

The following proposition is straightforward

Proposition 2: $L(\Gamma_1 \parallel \Gamma_2) = \{u \in (\Sigma_1 \cup \Sigma_2)^* : u/_{\Sigma_i} \in L(\Gamma_i), i = 1,2\}$
$\Sigma_1 = \Sigma_2 \Longrightarrow L(\Gamma_1 \parallel \Gamma_2) = L(\Gamma_1) \cap L(\Gamma_2)$
$\Sigma_1 \cap \Sigma_2 = \emptyset \Longrightarrow L(\Gamma_1 \parallel \Gamma_2) = L(\Gamma_1) ||| L(\Gamma_2)$ where $|||$ is the shuffle operation on languages.

3.3.2 Automata for numbered patterns.

Let m be a numbered pattern with its associated poset $P(m) = (I, \leq_m, l \times Id_I)$.

A state of the pattern automaton $Aut(m)$ is a map $state : I \longrightarrow \{x,r,t\}$ which maps the integers of the pattern on different tags, defined as follows: if the element $i \in I$ has been (received and) accepted, then $state(i) = r$; if i has not been received but belongs to the causes of an accepted element, and thus its loss has been accepted, $state(i) = t$. Initially and until i or any of its futures has been accepted, $state(i) = x$.

We first define for each pattern two automata: the first one denoted $Aut_r(m)$, called reliable, accepts exactly the set of linearisations of the pattern, the other one denoted $Aut_u(m)$, accepts (the linearizations of) all the sub-words of m.

For each $i \in I$ we denote by $p(i)$ the set of **immediate predecessors** of i in the relation $\leq_m$, i.e. $p(i) = \{j \in I : j <_m i \bigwedge (j \leq_m k \leq_m i \Longrightarrow k = j \bigvee k = i)\}$.

In the **reliable automaton** $Aut_r(m)$, the unique final state is the one where all the elements have been received, there is a single transition $T_r = \{(g,o)\}$. The guard $g(state, a, i)$ tests if all the immediate predecessors of the element being processed have been received.

$Aut_r(m) = (S, A \times I, T_r, state_0, F_r)$ where
$S = I \longrightarrow \{x,r,t\}$ and $state_0 \in S$ is defined by $\forall i \in I : state_0(i) = x$.

$F_r = \{state \in S,\ state(I(m)) \subseteq \{r\}\}$

$T_r = \{(g,o)\}$ is $g(state,a,i) = (state(i) = x \bigwedge j \in p(i) \Longrightarrow state(j) = r)$

The new state $state' = o(state,a,i)$ is $state'(j) = state(j)$ for all $i \neq j$ and $state'(i) = r$.

The language of the reliable automaton $L(Aut_r(m))$ is the set of linearisations of the emitted language $EL(m)$:

Proposition 3: $\forall m \in T_{np},\ L(Aut_r(m)) = Lin(EL(m))$

In the **unreliable automaton** $Aut_u(m)$ any state is a final state, there is a single transition and the guard jusst tests that the item has not been neither received nor declared lost:

$Aut_u(m) = (S, A \times I, T_u,\ state_0,\ F_u)$ where

$S = I \longrightarrow \{x,r,t\}$ and $state_0 \in S$ is defined by $\forall i \in I :\ state_0(i) = x$.

$F_u = S$.

$T_u = \{(g,o)\}$ is $:g(state,a,i) = (state(i) = x)$

The new state $state' = o(state,a,i)$ is $state'(j) = t$ if $j <_m i \bigwedge state(j) = x$, $state'(j) = r$ if $i = j$, $state'(j) = state(j)$ otherwise

The language of the unreliable automaton $L(Aut_u(m))$ is the set of linearisations of the delivered language $DL(m)$, which coincides with the set of subwords of $EL(m)$:

Proposition 4: $\forall m \in T_{np},\ L(Aut_u(m)) = Lin(DL(m))$

3.3.3 Automata for lossy patterns.

For any numbered pattern $m \in T_{np}$, $Aut_u(m)$ is the automaton recognizing $DL(m)$ and we define $Aut(m) = Aut_u(m)$. Let m and m' be two numbered patterns with $m \triangleright m'$, we have $I(m') \subseteq I(m)$, and then by proposition 2, $L(Aut_u(m) \parallel Aut_r(m')) = \{v \in L(Aut_u(m)),\ v/_{A \times I(m')} \in L(Aut_r(m'))\}$, and thus $Aut_u(m) \parallel Aut_r(m')$ is the automaton recognizing the elements of $Lin(DL(m))$ which "contain" $Lin(EL(m'))$: $L(Aut_u(m) \parallel Aut_r(m')) = Lin(DL(m \triangleright m'))$. We define inductively the automaton $Aut(\rho \triangleright m)$ by $Aut(\rho/ \triangleright m) = Aut(\rho) \parallel Aut_r(m)$.

If $m,\ m_j \in T_{np},\ j = 1..n$, have $Aut(m/ \triangleright \{m_j,\ j = 1..n\}) = Aut_u(m) \parallel_{j \in [n]} Aut_r(m_j)$.

Proposition 5: $\forall \rho \in T_{lp},\ L(Aut(\rho)) = Lin(DL(\rho))$

3.4 Elementary streams.

Elementary streams are caracterised by an iterated lossy pattern and a set of loss specifications per windows of adjacent patterns. They are defined by the syntax

$$\boxed{f := \rho^* \mid f/[B,n,\lambda]}$$

where $\rho \in T_{lp}$ is a lossy pattern and $[B,n,l] \in 2^A \times \aleph \times \aleph$ is a **loss expression**, $B \subseteq A$ is a subset of labels, and $n, l \in \aleph$ are integers. It defines a window of n patterns, for which no more than l elements labelled by B may be lost in the deliverable language. We denote by T_{es} the set of elementary stream terms. The root pattern $R(f) \in T_{np}$ of an elementary stream $f \in T_{es}$ ds the root definition on T_{lp} by $R(\rho^*) = R(\rho)$ for $\rho \in T_{lp}$and $R(f/[B,n,l]) = R(f)$.

3.4.1 Semantics of iteration.

The languages $EL(f)$ and $DL(f)$ for $f \in T_{es}$ are sets of streams labelled on $A \times \aleph \times \aleph$. A labelled poset belonging to an elementary stream is then a tuple $(D, \leq, l \times \nu \times \mu)$ with $l \times \nu \times \mu : D \longrightarrow A \times \aleph \times \aleph$ where $\mu : D \longrightarrow \aleph$ defines the (pattern) iteration number at which the item occurs.

Let be $s = (D, \leq, l \times \nu) \in P(A \times \aleph)$ and $n \in \aleph$, then $s \times n$ is the $A \times \aleph \times \aleph$–labelled poset $s \times n = (D, \leq, l \times \nu \times \mu)$ with $\mu(D) = \{n\}$, i.e. all elements of D are mapped on n by μ . Let $L \subseteq P(A \times \aleph)$ be a language, and $n \in \aleph$ then $L \times [n] \subseteq P(A \times \aleph \times \aleph)$ is the language

$L \times [n] = \{(s_1 \times 1) \bullet (s_2 \times 2) \bullet .. \bullet (s_n \times n), \{s_i, i = 1, n\} \subseteq L\}$, the concatenation being naturally restricted to all nuples $(s_i)_{i \in [n]} \in L^n$ with disjoints event sets.

The emitted and deliverable languages $EL(\rho^*)$ and $DL(\rho^*)$ are defined by: $EL(\rho*) = \bigcup_{n \in \aleph} EL(\rho) \times [n]$ and $DL(\rho*) = \bigcup_{n \in \aleph} DL(\rho) \times [n]$.

3.4.2 Semantics of loss expressions.

Let $f/[B, n, \lambda]$ be a stream term and $m = |R(f)|_B$ the number of items in the root pattern $R(f)$ which are labelled by a letter of B. For each consecutive n emitted patterns, we have to deliver at least $(n \times |R(f)| - \lambda)$ items labelled by a letter of B.

Let be $\sigma = (D, \leq, l \times \nu \times \mu) \in DL(f)$, and let us note $D_i^n = \mu^{-1}(\{i, i+1, ...i+n-1\})$ the set of data units in the window of size n starting at pattern number i, then σ satisfies the constraint $[B, n, \lambda]$ iff $\forall i \in \aleph, |D_i^n \cap l^{-1}(B)| \geq n \times |R(f)| - \lambda$. We can then define the **deliverable language** of $f/[B, n, \lambda]$: $DL(f/[B, n, \lambda]) = \{\sigma = (D, \leq, l \times \nu \times \mu) \in DL(f), \forall i \in \aleph, |D_i^n \cap l^{-1}(B)| \geq n \times |R(f)| - \lambda\}$

An elementary stream can be written $f = \rho^*/\{[B_i, n_i, \lambda_i], i = 1, k\}$

Proposition 6: $DL(\rho^*/\{[B_i, n_i, \lambda_i]\}_{[k]}) = \bigcap_{[k]} DL(\rho^*/[B_i, n_i, \lambda_i])$

3.5 Elementary stream automata.

3.5.1 Iteration.

Let ρ a lossy pattern, with the root poset $P(R(\rho)) = (I, \leq_\rho, l \times Id_I)$, a state of the elementary stream automaton $Aut(\rho^*)$ is a pair $(state, cur)$ where $cur \in \aleph$ denotes the current pattern number being processed, and $state$ is a state of $Aut(\rho)$ as defined above. Inputs are triples (a, i, h) where (a, i) is an input of $Aut(\rho)$, and h is the iteration pattern number to which the input belongs. Let $\Gamma = Aut(\rho) = (Q, A \times I, T, state_0, F)$ be the pattern automaton, then we define the automaton $Aut(\rho^*)$ by

$$Aut(\rho^*) = (Q \times \aleph, A \times I \times \aleph, T^*, (state_0, 1), F \times \aleph)$$

The transitions are defined by

$(g, o) \in T^* \iff \exists (g', o') \in T$ such that one of the following cases occur:

1. $g(state, cur, a, i, h) = ((cur = h) \bigwedge g'(state, a, i))$ and $o(state, cur) = (o'(state), cur)$
2. $g(state, cur, a, i, h) = ((state \in F) \bigwedge (h = cur + 1) \bigwedge g'(state_0, a, i))$ and $o(state, cur) = (o'(state_0), cur + 1)$

The first case deals with inputs from the current pattern $(cur = h)$, while the second defines the conditions for a jump to the next iteration $(h = cur + 1)$.

Proposition 7: $\forall \rho \in T_{lp},\ L(Aut(\rho^*)) = Lin(DL(\rho^*))$

3.5.2 Single Loss.

We define the automaton $Aut(\rho^*/[B, n, \lambda])$ of an elementary stream with a single loss expression. It is defined using the automaton $Aut(\rho)^*$ defined above and the root pattern $P(R(\rho)) = (I, \leq_\rho , l \times Id_I)$. Let be the pattern automaton $Aut(\rho) = (Q, A \times I, T, state_0, F)$ and its iteration $Aut(\rho)^* = (Q \times \aleph, A \times I \times \aleph, T^*, (state_0, 1), F \times \aleph)$. Let be $[B, n, \lambda]$ the loss triple, we use a vector of n integers $\lambda[n] = \lambda_1..\lambda_n$ recording the number of lost elements labelled by B on the n last patterns, with the required property that $\sum_{k=1}^{n} \lambda_k \leq \lambda$. The value of λ_n is the number of lost elements in the current pattern. Then the automaton $Aut(\rho^*/[B, n, \lambda])$ is defined by

$$Aut(\rho^*/[B, n, \lambda]) = (Q \times \aleph \times \aleph^n \times \aleph,\ A \times I \times \aleph, T_\pi,\ (state_0, 1, 0^n, 0),\ F \times \aleph \times \aleph^n \times \aleph)$$

Transitions are pairs (g,o) such that $g : (Q \times \aleph \times \aleph^n \times \aleph) \times (A \times I(\rho) \times \aleph) \longrightarrow \{0,1\}$, and $o : (Q \times \aleph \times \aleph^n \times \aleph) \times (A \times I(\rho) \times \aleph) \longrightarrow Q \times \aleph \times \aleph^n \times \aleph$.

$(g,o) \in T_\pi \iff \exists (g',o') \in T^*$ such that one of the following occurs:

1- Current pattern evolution

guard $g(state, cur, \lambda[], \kappa, a, i, h) = g_{cr}(state, cur, \lambda[], \kappa, a, i, h) \bigwedge g'(state, cur, a, i, h))$ such that

$g_{cr} = (cur = h) \bigwedge (\lambda_n + |\{j,\ j <_\rho i \bigwedge state(j) = x \bigwedge l(j) \in B\}| \leq \kappa)$,

action $o(state, cur, \lambda[], \kappa, a, i, h) = (o'(state, cur), \lambda'[], \kappa')$ is such that:

$\lambda'_n = \lambda_n + |\{j,\ j <_\rho i \bigwedge state(j) = x \bigwedge l(i) \in B\}|$

$\lambda'_i = \lambda_i$ for $i \neq n$

$\kappa' = \kappa$

2- Next pattern evolution

guard $g(state, cur, \lambda[], \kappa, a, i, h) = g_{nx}(state, cur, \lambda[], \kappa, a, i, h) \bigwedge g'(state, cur, a, i, h))$ such that

$g_{nx} = (h = cur + 1) \bigwedge (\lambda_n + J(j, state, B) \leq \kappa) \bigwedge (\sum_{k=2}^{n} \lambda_k + J(i, state, B) + K(i, B) \leq \lambda)$ where $J(i, state, B)$ and $K(i, B)$ are auxiliary integer values defined by:

$J(i, state, B) = |\{j,\ j \neq i \bigwedge state(j) = x \bigwedge l(j) \in B\}|$ and $K(i, B) = |\{j,\ j <_\rho i \bigwedge l(j) \in B\}|$

action $o(state, cur, \lambda[], \kappa, a, i, h) = (o'(state_0, cur + 1), \lambda'[], \kappa')$ such that

$\lambda'_i = \lambda_{i+1}$ for $i = 1, n-2$

$\lambda'_{n-1} = \lambda_n + J(i, state, B)$

$\lambda'_n = K(i, B)$

$\kappa' = \lambda - \sum_{k=1}^{n-1} \lambda'_k$

Proposition 8: $\forall (\rho^*/[B, n, \lambda]) \in T_{es} :\ L(Aut(\rho^*/[B, n, \lambda])) = Lin(DL(\rho^*/[B, n, \lambda]))$

3.5.3 Multiple losses.

Let be $f = \rho^*/\{[B_i, n_i, \lambda_i], i = 1, k\}$ then we define $Aut(\rho^*/\{[B_i, n_i, \lambda_i], i = 1, k\}) = \|_{i \in [k]} (Aut(\rho^*)[B_i, n_i, \lambda_i])$

This definition yields the duplication of the states of the automaton $Aut(\rho^*)$, but in an implementation they can be merged in a single object.

The following result is the consequence of the propositions 1 and 8.

$L(Aut(\rho^*/\{[B_i, n_i, \lambda_i], i = 1, k\})) = \cap_{i \in [k]} L(Aut(\rho^*/[B_i, n_i, \lambda_i])) = \cap_{i \in [k]} Lin(DL(\rho^*/[B_i, n_i, \lambda_i]))$

Proposition 9: $\forall f \in T_{es},\ L(Aut(f)) = Lin(DL(f))$

4 Synchronised streams and automata.

4.1 Synchronised streams.

We consider now a set C of channel names (which can model concrete connections, but not exclusively) . A synchronised stream is a finite set of elementary streams, each associated with a specific channel name, and synchronised by expressions. The location operator denoted $c :: f$ associates the channel $c \in C$ to the elementary stream $f \in T_{es}$. The syntax of synchronised streams is then:

$$\boxed{\psi := c :: f \mid \psi \parallel \psi \mid \psi[\xi]}$$

where $c \in C$, $f \in T_{es}$, and $\xi \in SE$ where SE is the set of synchronisation expressions defined below. We denote by T_{st} the set of stream terms.

4.1.1 Synchronisation expressions.

A synchronisation expression is a pair $((c_1,k),(c_2,h)) \in (C\times\aleph)\times(C\times\aleph)$ with $c_1 \neq c_2$. The expression $((c_1,k),(c_2,h))$ constraints the pattern numbered k on the channel c_1 to be delivered before the pattern numbered h on the channel c_2. **Parametric expressions** allow the specification of unbounded sets of pairs by mean of integer functions. Let Θ be the set of linear integer functions $\Theta = Linear(\aleph \longrightarrow \aleph)$, the pair $((c_1,\eta_1),(c_2,\eta_2)) \in (C\times\Theta)\times(C\times\Theta)$, constraints for each integer i, the pattern numbered $\eta_1(i)$ on the channel $c_1$to be delivered before the pattern numbered $\eta_2(i)$ on the channel c_2. The syntax of synchronisation expressions is then

$$\boxed{\xi := ((c_1,k),(c_2,h))|((c_1,\eta_1),(c_2,\eta_2))}$$

where $((c_1,k),(c_2,h)) \in (C\times\aleph)\times(C\times\aleph)$ and $((c_1,\eta_1(i)),(c_2,\eta_2(i))) \in (C\times\Theta)\times(C\times\Theta)$. The set of synchronisation expressions is $SE = ((C\times\aleph)\times(C\times\aleph)) \cup ((C\times\Theta)\times(C\times\Theta))$. Clearly an element $\eta \in \Theta$ can be represented by a pair of integers (a,b)s.t. $\eta(i) = ai+b$. The **set of channels** $Ch(\xi)$ of an expression ξ is defined by $Ch((c_1,k),(c_2,h)) = Ch((c_1,\eta_1),(c_2,\eta_2)) = \{c_1,c_2\}$

Compound synchronisation expression.

To synchronise two elementary streams periodically, with two different periods, we define the expression (c_1,k_1) <> (c_2,k_2) where $(c_i,k_i) \in C\times\aleph$, $i = 1,2$, which is equivalent to two parametric expressions as follows:

$$(c_1,k_1) <> (c_2,k_2) \stackrel{def}{=} \{((c_1,k_1\times i)\,(c_2,k_2\times i+1)),((c_2,k_2\times i),(c_1,k_1\times i+1)),\ i\in\aleph\}$$

Note that (c_1,k_1) <> (c_2,k_2) is a symmetric expression.

The set of channels $Ch(f)$ of a stream is defined by $Ch(c :: f) = \{c\}$, $Ch(\psi_1||\psi_2) = Ch(\psi_1) \cup Ch(\psi_2)$, $Ch(\psi[\xi]) = Ch(\psi) \cup Ch(\xi)$.

4.1.2 Channel mapping and composition's semantics.

An element of $EL(\psi)$ or $DL(\psi)$ is a labelled poset $\sigma = (D,\leq,c\times l\times\nu\times\mu) \in P(C\times A\times\aleph\times\aleph)$ where $c : D \longrightarrow C$ maps data on channels, $l : D \longrightarrow A$ is the type labelling map, $\nu : D \longrightarrow \aleph$ is the inside pattern numbering, $\mu : D \longrightarrow \aleph$ is the pattern iteration number which data belong to. The languages $EL(c :: f)$ and $DL(c :: f)$ are deduced from $EL(f)$ and $DL(f)$ by adding the channel c in the labels:

$DL(c :: f) = \{(D,\leq,ch\times l\times\nu\times\mu) \in P(C\times A\times\aleph\times\aleph),(D,\leq,l\times\nu\times\mu) \in DL(f) \bigwedge ch(D) = \{c\}\}$

The languages $EL(\psi_1 \parallel \psi_2)$ and $DL(\psi_1 \parallel \psi_2)$ are the parallel composition of the component languages: $EL(\psi_1 \parallel \psi_2) = EL(\psi_1) \parallel EL(\psi_2)$ and $DL(\psi_1 \parallel \psi_2) = DL(\psi_1) \parallel DL(\psi_2)$

4.1.3 Synchronisation semantics.

The **emitted language and delivered languages of a stream** $\psi[\xi]$ are defined as the order extensions (see section 2) of streams belonging to $EL(\psi)$ and $DL(\psi)$ which satisfy the expression ξ: $EL(\psi[\xi]) = \{\sigma \in Ext(EL(\psi)),\sigma \models \xi\}$ and $DL(\psi[\xi]) = \{\sigma \in Ext(DL(\psi)),\sigma \models \xi\}$

Let be $\sigma = (D,\leq_\sigma,c\times l\times\nu\times\mu)$ the relation $\models$ is defined by:

$\sigma \models ((c_1,k),(c_2,h)) \Longleftrightarrow (c\times\mu)^{-1}(c_1,k) \leq_\sigma (c\times\mu)^{-1}(c_2,h)$

$\sigma \models ((c_1,\eta_1),(c_2,\eta_2)) \Longleftrightarrow [\forall i \in \aleph : \sigma \models ((c_1,\eta_1(i)),(c_2,\eta_2(i))]$

For the deliverable languages, we have to adapt the definition to the possible loss of the items numbered by the integers of the expression. This entails the following definition:

$\sigma \models ((c_1,k),(c_2,h)) \Longleftrightarrow \forall i \leq_n k, \forall j \geq_n h : (c\times\mu)^{-1}(c_1,i) \leq_\sigma (c\times\mu)^{-1}(c_2,j)$

This latter definition coincides with the previous one for partial words belonging to the emitted language.

Any well formed stream ψ where the channels have been indexed, i.e. $Ch(\psi) = \{c_i,\ i = 1,n\}$ can be written in normalised form, $\psi = (\|_{i\in[n]}\ c_i :: f_i)[\xi_1][\xi_2]...[\xi_k]$ where = stands in particular for both emitted and delivered language equality.

4.2 Synchronised Automata.

A **synchronised automaton** is a pair $\Delta = (C_\Delta, \{\Gamma_c, c \in C_\Delta\})$ composed of a set of channel names $C_\Delta \subseteq C$, and for each $c \in C_\Delta$ an **elementary automaton with global testing** Γ_c . Γ_c is a tuple $\Gamma_c = (Q_c, \Sigma_c, T_c, q_{0c}, F_c)$ for which there is an elementary automaton $Elem(\Gamma_c) = (Q_c, \Sigma_c, Loc(T_c), q_{0c}, F_c)$, such that

$T_c \subseteq G_c^l \times G_c^g \times O_c$ is a set of **abstract global transitions**, i.e.

- G_c^g is a set of **boolean guards** on pairs (global state, input), i.e. $G_c^g \subseteq \bigcup_{D \subseteq C_\Delta} (\prod_{c \in D} Q_c \times \Sigma_c \longrightarrow \{0,1\})$.
- $Loc(T_c) = T_c / G_c^l \times O_c$. The projection of T_c onto $G_c^l \times O_c$ is the set of abstract transitions of $Elem(\Gamma_c)$.

For a subset $D \subseteq C_\Delta$ and a guard $g \in \prod_{c \in D} Q_c \times \Sigma_c \longrightarrow \{0,1\})$, the subset D is called the domain of g and denoted $Dom(g) = D$. In the following lines we abbreviate C_Δ in C due to no possible confusion. The sets Σ_c are defined by $\Sigma_c = \{c\} \times A \times \aleph \times \aleph$ and thus pairwise disjoints. Lets note $\sum_\Delta = \cup_{c \in C} \sum_c$, and $Q_\Delta = \Pi_{c \in C} Q_c$, then the transition relation $\longrightarrow_\Delta \subseteq Q_\Delta \times \sum_\Delta \times Q_\Delta$ is defined by

$(q_d) \xrightarrow{l}_\Delta (q'_d) \Longleftrightarrow \exists c \in C, \exists (g_l, g, o) \in T_c$ such that :

1. $l \in \Sigma_c \bigwedge g_l(q_c, l) = 1 \bigwedge q'_c = o(q_c, l)$ (Local automaton transition). This means that $q_c \xrightarrow{l}_c q'_c$ is a transition triple in $Loc(T_c)$ (for the transition $(g_l, o) \in Loc(T_c)$),
2. $g((q_d)_{d \in Dom(g)}, l) = 1$ (Test global predicate)
3. $q'_d = q_d$ if $d \neq c$ (Other components than Γ_c do not change their local states)

The transition relation is extended to $Q_\Delta \times \sum_\Delta^* \times Q_\Delta$ as for EA, and the definition of the **language** $L(\Delta)$ is done accordingly.

We omit the global predicate g in the triple $(g_l, g, o) \in G_c^l \times G_c^g \times O_c$ when it is identically true. This in particular for an automaton $\Delta = (\{c\}, \{\Gamma_c\})$, i.e. with only one elementary component.

The projection of a behaviour of the synchronised automaton on the inputs of some components is a behaviour of the component. Let be $\Delta = \{\Gamma_i, i = 1, n\}$ a synchronised automaton, we have:

Proposition 10: $\forall c \in C_\Delta : L(\Delta)/\Sigma_c \subseteq L(Loc(\Gamma_c))$

The **parallel composition** of synchronised automata is partial and defined on operands with disjoint channel sets. The channels and automata sets of the resulting automaton are the unions of the ones of components. Let be $\Delta_1 = (C_1, \{\Gamma_c, c \in C_1\})$ and $\Delta_2 = (C_2, \{\Gamma_c, c \in C_2\})$, then: $\Delta_1 \parallel \Delta_2 = (C_1 \uplus C_2, \{\Gamma_c, c \in C_1\} \cup \{\Gamma_c, c \in C_2\})$. The resulting language is the shuffle of the components ones, i.e. $L(\Delta_1 \parallel \Delta_2) = L(\Delta_1) ||| L(\Delta_2)$.

When we compose two streams by the $\parallel$ operator, there is no interaction needed between the automata recognizing the components. Instead when we synchronise a stream by an expression, the two component automata dealing with the channels occurring in the expression have to be modified: what is modified is the guard of some transitions, which do not only depend on the local state, but also on the (local) state of the other component.

4.2.1 Channel mapping and parallel composition.

The synchronised automaton $Aut(c :: f)$ is simply deduced from the elementary automaton $Aut(f) = (Q, \Sigma, T, q_0, F)$ by introducing the channel in the inputs as follows: $Aut(c :: f) = (\{c\}, \{(Q, \{c\} \times \Sigma, T_c, q_0, F)\})$ where $T_c = \{(g, o) | \exists (g', o) \in T, \forall (q, l) \in Q \times \Sigma : g(q, (c, l)) = g'(q, l)\}$

If $\psi_1 \parallel \psi_2$ is a well formed term, the components $Aut(\psi_1)$ and $Aut(\psi_2)$ have disjoint channels, sets, and we can define $Aut(\psi_1 \parallel \psi_2) = Aut(\psi_1) \parallel Aut(\psi_2)$

4.2.2 Synchronisations.

The synchronisation operator $\Delta[\xi]$, to be used in an inductive definition $Aut(\psi[\xi]) = Aut(\psi)[\xi]$, models the synchronisation of the two elementary components of Δ reading data from the channels occurring in the expression ξ. Each component automaton proceeds independently, except when it reaches a pattern number (an iteration number) which should be synchronised with some pattern of the other component. An expression in ES is not symmetric, and two situations may occur: either a stream has to wait for another stream to reach a specific pattern number, or a stream is awaited for.

Let be $\Delta = (C_\Delta, \{\Gamma_c, c \in C_\Delta\})$, then we define $\Delta[\xi]$ by $\Delta[\xi] = (C_\Delta, \{\Gamma_c[\xi], c \in C_\Delta\})$, where $\Gamma_c[\xi]$ is defined below. In the definition which follows, for simplicity's sake, we suppose that $C_\Delta = \{c_i,\ i = 1, n\}$, we note $\Gamma_i = \Gamma_{c_i}$ for $i = 1, n$, and all local states components are indexed accordingly.

For the definition of $\Gamma_i[\xi]$ the changes w.r.t Γ_i occur in the set of transitions, and with three cases . We note $(state_i, cur_i)$ the current state of Γ_i and (c_i, a, u, h) the input label to be tested, the pair (a, u) being irrelevant for the definition below.

Case 1: $c_i \notin Ch(\xi)$, then $\Gamma_i[\xi] = \Gamma_i$

Case 2: $\xi = ((c_i, k_i), (c_j, k_j))$, then $(g_l, g, o) \in T(\Gamma_i[\xi]) \Longleftrightarrow \exists (g_l, g', o) \in T(\Gamma_i)$ s.t.

if $g_l \Longrightarrow (h = cur_i)$ then

$g = g' \bigwedge [(state_i \in F_i) \bigwedge (cur_i = k_i) \Longrightarrow (cur_j < k_j)]$ (global test on $Q_i \times Q_j$)

else $g = g'$

Case 3: $\xi = ((c_j, k_j), (c_i, k_i))$, then $(g_l, g, o) \in T(\Gamma_i[\xi]) \Longleftrightarrow \exists (g_l, g', o) \in T(\Gamma_i)$ s.t.

if $g_l \Longrightarrow [(state_i \in F_i) \bigwedge (h = cur_i + 1)]$ then

$g = g' \bigwedge [(cur_i + 1 = k_i) \Longrightarrow (cur_j > k_j) \bigvee ((cur_j = k_j) \bigwedge (state_j \in F_j))]$ (global test on $Q_i \times Q_j$)

else $g = g'$

Note that the conditions $g_l \Longrightarrow (h = cur_i)$ and $g_l \Longrightarrow [(state_i \in F_i) \bigwedge (h = cur_i + 1)]$ in this definition reflect the two main cases for the guards of elementary automata.

For a parametric synchronisation expression, the definition of $\Gamma_i[\xi]$ is a mere adaptation of the one above. As mentionned above, we can define $Aut(\psi[\xi]) = Aut(\psi)[\xi]$. Applied to a stream $\psi = (\|_{i \in [n]}\ c_i :: f_i)[\xi_1][\xi_2]...[\xi_k]$, this yields the automaton $Aut(\psi) = (\|_{i \in [n]}\ Aut(c_i :: f_i))[\xi_1][\xi_2]...[\xi_k]$

The following theorem asserts the coherence property between the two semantics defined on the stream algebra.

Theorem:

$$\forall t \in T_{st},\ Lin(DL(t)) = L(Aut(t))$$

5 Conclusion.

We have presented a model for multimedia data streams where the order and reliability properties are specified precisely by operators, and in a compositional way which is used to define the part of the transport protocols needed to handle these streams. Different improvements of the signature, in particular a sequencing operator, in order to model renegociation, and a refinement operator, allowing different levels of specification, are easy to include in the framework.

The class of automata defined here extends by the use of counters the existing class of Asynchronous Cellular Automata for Pomsets as defined in [7][12]. The pomsets recognised by this new class of automata are labelled by (vectors of) integers, and constitute an interesting field of investigation for partial order theory. Particular subclasses with restricted computing capacities should be defined allowing a set of tractable problems to be worked out.

Complexity aspects have to be introduced, in particular in the automata behaviours. Particular pattern and stream structures should be worked out for their low or high recognition complexity.

References

[1] Amer.P,Chassot.C,Conolly.T.J,Diaz.M.,Conrad.P. *Partial-order transport service for multimedia and other applications.* IEEE/ACM Transactions on Networking, Vol.2, Nř5, 440-456, Octobre 1994

[2] P. Berthou, J.F. Fanchon, E. Exposito, Model for Partialy Ordered and Partially Reliable Connections (LAAS research report 00456, november 2000) *(http://www.laas.fr/~fanchon/RapportPOC.ps.gz)*

[3] Chassot.C.DDiaz.M,Lozes.A. *From the partial order concept to partial order multimedia connections.* Journal of High Speed Networks, vol. 5, n. 2, 1996.

[4] Cece.G,Finkel.A,Purushotaman.S. *Unreliable channels are easier to verify than perfect channels.* Information and computation, vol.124 (1),20-31,1996.

[5] H.Comon,Y.Jurski, *Multiple counters automata, safety analysis and Presburger arithmetic.* CAV'98. LNCS vol. 1427,(1998) 268-279.

[6] M.Diaz,P.Senac. *Time Stream Petri Nets for Multimedia Information.* ICATPN 94 Zaragoza, June 94.

[7] M.Droste,P.Gastin,D.Kuske *Asynchronous Cellular automata for Pomsets.* Theoretical Computer Science Vol. 247, 1-2, pp. 1-38, September 2000.

[8] Fanchon.J. *A Fifo-net model for processes with asynchronous communication.* Lecture Notes in Computer Science, Vol. 609; Advances in Petri Nets 1992, 152-178. Springer-Verlag, 1992.

[9] Fanchon.J. *Trace Channel Nets.* ICATPN 99 .Lecture Notes in Computer Science, Vol. 1639. Springer-Verlag, 1999.

[10] Gischer,J. *The equational theory of pomsets.* Theoretical Computer Science 61 (1988)199-224.

[11] Grabowski.J. *On partial Languages.* Fundamenta Informaticae IV.2 . (1981).

[12] Kuske,D. *Asynchronous Cellular and Asynchronous Automata for Pomsets.* Lecture Notes in Computer Science, Vol 1466, CONCUR 98, 517-523, Springer -Verlag 1998.

[13] P.Owezarski, M. Diaz, C. Chassot, "*A Time Efficient Architecture for Multimedia Applications*", to appear in IEEE Journal on Selected Areas in Communication, Special Issue on Protocols and Architectures for Applications of the 21st Century, 1998

[14] Pratt.W.*Modeling concurrency with partial orders.* Int.J.Parallel Programming 15 (1987) 33-71.

[15] P.Senac, M.Diaz, A.Leger, P.de Saqui-Sannes. *Modeling logical and temporal synchronisation in Hypermedia Systems.* IEEE Journal on Selected Areas in Communications, 1995.

[16] Thomas.W.*Automata theory on Trees and Partial Orders.* Lecture Notes in Computer Science, Vol. 1214, TAPSOFT 97,Springer -Verlag 1998.