Published in "Advances in Petri Nets 1991", G.Rozenberg (Ed.), LNCS 524, pp 1-28

DELAYS IN SYNCHRONIZED ELEMENTARY NET SYSTEMS

Charles ANDRE
Université de NICE - Sophia Antipolis
Laboratoire de Signaux et Systèmes
Equipe de l'I3S (U.R.A 1376 du CNRS)
41 Bd Napoléon III
F-06041 NICE Cédex - FRANCE

ABSTRACT

Synchronized Elementary Net Systems (SENS) are a model for systems consisting of events, conditions and subject to generalized timing constraints. The constraints are quantified and are relative to some "reference events" such as clock ticks, meters, (multiform time). The partial ordering of event occurrences can be altered by the "timing" constraints which induce additional precedences. This paper provides algorithms which compute delays between event occurrences. The "dioid" algebraic structure is the underlying mathematical model. Applications of delays to some reachability problems are given.

KEYWORDS: Time Petri nets, Elementary Net Systems, Synchronized Elementary Net Systems, POset, Process, Delay, Dioid.

CONTENTS

I. INTRODUCTION

The aim of this paper is twofold: **modelling** and **analysis** of **concurrent systems** with **timing constraints**.

The application of Petri nets modelling and analysis techniques to safety-critical real-time systems has been explored in a recent paper [LevSto87]. The central example, proposed by the authors, was "a simple railroad crossing". This example has been taken up again in another paper about modelling, specification and verification of real-time embedded systems [OstWon87]. In this case, Extended State Machines (ESMs) were preferred to Petri nets. As both papers dealt with real-time systems, they had to express some timing constraints. The former used Time Petri nets [Merlin75], the latter chose a "Real-time Temporal Logic". We tackle this problem with **Synchronized Elementary Net Systems** (SENS)[André88].

Chap.II is devoted to **modelling**. Taking the Railroad Crossing System as an example, we show that SENS are suitable for modelling **multiform time**. Few approaches, but some synchronous languages ([BeCo85]), offer such a possibility.

The **dynamics of concurrent systems** can be expressed by "(non-sequential) processes" [Petri77]. In a previous paper ([André88]), we explained how the notion of processes could be extended to SENS. In Chap.IV, we introduce **a quantitative approach** which is complementary to the former presentation. The central problem is **the computation of delays** between event occurrences. According to their values, delays can induce a temporal ordering between some event occurrences. This alteration, due to the "timing constraints" is characteristic of the system behaviour and cannot be ignored.

To cope with multiform time, a richer structure than natural numbers is needed. We introduce **a dioid whose elements are sets of bags**. The technical material about SENS and this algebra, is given in Chap.III.

Delays are related to "longest paths" in cycle-free graphs. But, due to the firing rule of SENS, their computation gives rise to several specific issues, analyzed in Chap.IV. Effective algorithms to compute delays are given. They are applied to the Railroad Crossing System in order to prove its correctness.

The computation of delays is **restricted to conflict-free SENS**. This simplification allows to shed light on the essential problems. The extension to general SENS is beyond the scope of this paper. It is more technical and does not involve new concepts.

II. SYSTEM WITH TIMING CONSTRAINTS

II.1. A SIMPLE RAILROAD CROSSING

We take the system proposed in [LevSto87]. This simplified Railroad Crossing System is made of events and conditions. The system is composed of a railway and a train, a level crossing with its gate and a controller for the gate. The train can be **tr**avelling ("Tr"), **b**efore the **c**rossing ("Bc"), **w**ithin the **c**rossing ("Wc"), **p**ast the **c**rossing("Pc"). The position of the train is given by sensors called pedals which are pushed down by the wheels. "pb" and "pp" are the pedals signaling that the train is approaching the crossing(event "pb↓") or moving away from the crossing(event "pp↑"). "pic" is the event occurring when the locomotive enters the crossing, "poc" occurs when the rear wagon leaves the crossing. The controller receives information from the sensors and emits commands to the "gate". The gate can be up ("Up") or down ("Dn"). Actions on the gate are raising ("rg") or lowering ("lg"). Let Σ1 be the net modelling this system (Fig.1).

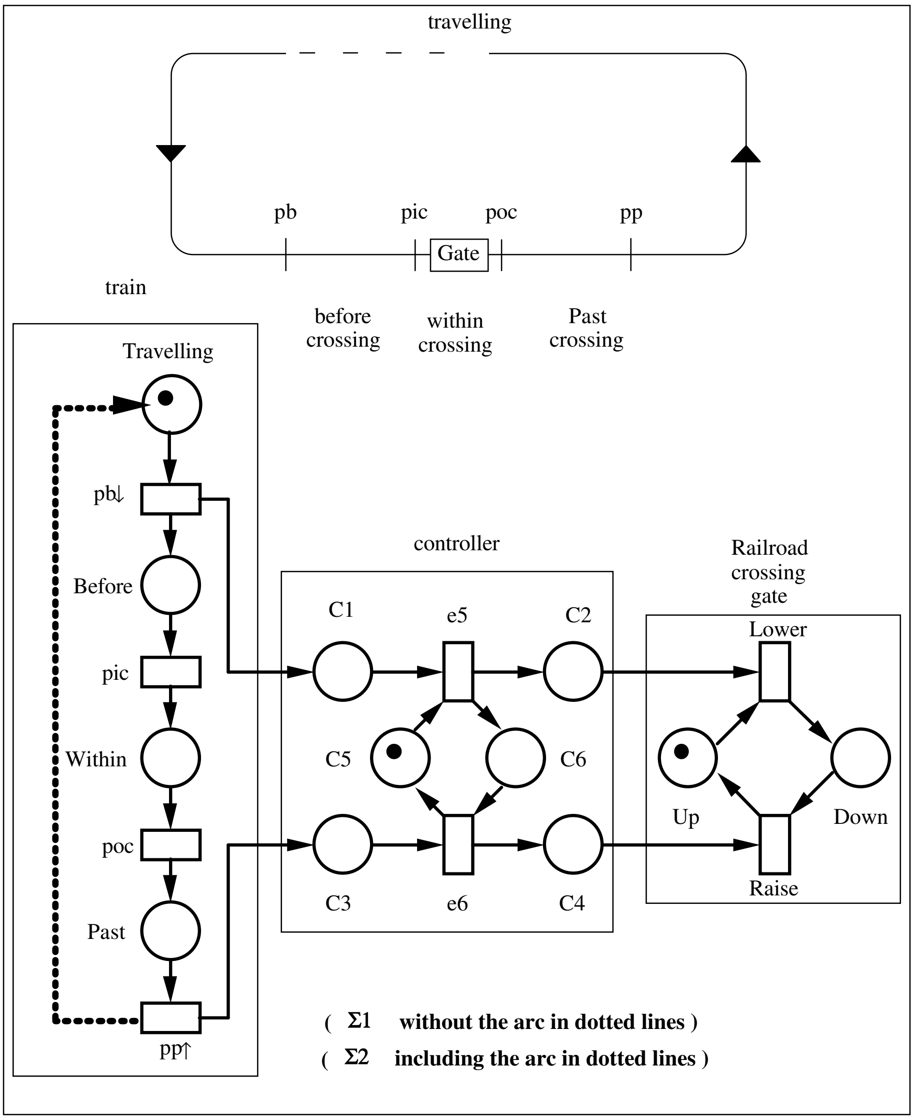

FIGURE 1: Railroad Crossing system.

II.2. PROCESS OF THE RAILROAD CROSSING

The "railroad crossing system" can be considered as an **elementary net system (ENS)** [Thiag86]. A "process" [Rozen86] represents a possible "run" of the system. In what follows, some familiarity with terminology and concepts relative to nets and processes are presumed. Fig.2 is a graphical representation of a process π1 of Σ1. This process is a classical one, taking no account of the timing constraints. In a previous paper [André88], we named it as a "free process". According to π1, {Wc, Up} is a possible subcase of the system, i.e the train may be at the crossing while the gate is Up. This situation is, typically, a dangerous one. In [LevSto87] techniques in order to detect and to avoid such "risky" states are proposed. They are based on the reachability graph and as such, they consider an interleaving semantics of concurrency.

This system is not live. If we suppose that the train comes back to the "travelling" condition after the occurrence of pp↑, then the system becomes live, but other properties are lost. Let Σ2 be the net obtained by adding an extra arc from "pp↑" to "travelling". If Σ2 is taken as an ENS, then it is not contact-free: e.g the case {Tr, C1, C3} is reachable and event "pb↓" is not enable due to a contact. If Σ2 is thought of as a place/transition net, then it is unbounded. These malfunctions are consequences of the fact that the firing of an enable transition can be delayed for any amount of time. They can be avoided by **imposing additional constraints** to the system.

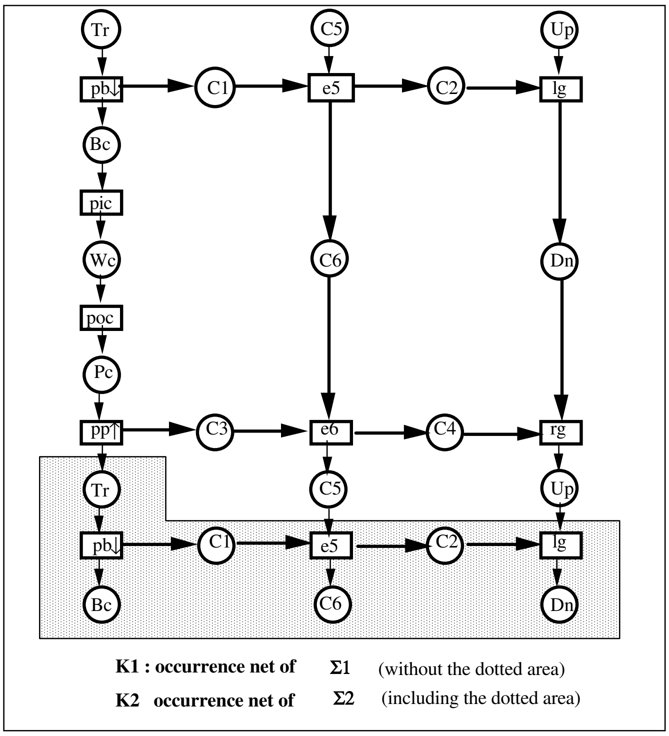

FIGURE 2: Occurrence net of the Railroad crossing system.

II.3. CONSTRAINTS

Generally, some form of "**feedback**" is necessary to control the system. **Regulation circuits** are convenient for this purpose (see [Reisig86] as an example of embedded system description using Petri nets). For the railroad crossing a strong constraint, not taken into account in our model, is that two trains cannot be on the same section, provided that the traffic lights are respected.

Giving **priority** is also a way to avoid malfunctions. A priority resolves a conflict in a deterministic manner. This solution is suggested for risky state avoidance [LevSto87]. When a conflict is objective [Reisig87], a priority can be expressed by net transformations. Unfortunately, for physical systems conflicts are often not objective [André88]. So, enhanced nets are needed.

Sometimes it is not necessary to introduce feedback. Some actions are launched and their effects are surely available some time later. These situations can be captured by a net model if we introduce **timing constraints on the firings of transitions**. Some transitions can be assigned an upper and lower bounds for the delay in firing. In most of the physical systems an upper bound is known for these delays. Typically, the controller is built to act "promptly" with respect to the response time of the system to be controlled. This "open loop" way of controlling a system can be dangerous, and certainly not allowed for high risk system such as railway transportation. Nevertheless, many systems are running in this way, a classical example is the communication between a microprocessor and its memory. The handshake (communication with feedback control) is not used, the processor requests information from the memory and the system is designed in such a way that the information will be available on the bus, in time.

The **Synchronized Elementary Net Systems** (SENS for short) have been introduced to capture both priority and generalized timing constraints [André88]. Unlike usual Petri nets "with time", SENS can model constraints involving multiple time references (**multiform time**). Approaches considering multiform time, are common in synchronous programming languages (e.g see ESTEREL [BeCo85]) and, all in all, are natural for distributed systems. The "Railroad Crossing System" (Σ2) is a simple, albeit significant, illustration of systems with timing constraints. We will explain in the following sections how to deal with such constraints in SENS
For analysis and illustrative purpose we consider the following constraints:
E_{Ref} = {m, h, s} where m stands for "meter", h for the "clock signal" of the controller, s for "second". "m", "h", "s" are the different "times" for our example. Event m occurs in the locomotive while the wheels are rolling ("tick" of a revolution counter).

The distance between pb and the crossing is 500m, so is the distance between the crossing and pp. The length of the crossing is 100m. A railroad section (safe distance between trains) is 1800m. The maximal length for a train is 200m. These physical constraints are expressed by **inertias** (I) and **latencies** (L). With respect to the reference event "m": I(Bc,pic) = I(Pc,pp↑) = 500; I(Wc,poc) = 100; I(Tr,pb↓) = 1800; L(pb↓) = ω (infinity); L(poc) = 200; L(pic) = L(pp↑) = 0.

For the controller, inputs change asynchronously w.r.t "h". In [André88], we explained how to model such a system using SENS. For the railroad system we assume that I(C1,e5) = I(C3,e6) = 7; I(C5,e5) = I(C6,e6) = 0; L(e5) = L(e6) = 20 w.r.t "h".

As for the gate, due to the mechanical inertia, we take: I(C2,lg) = I(C4,rg) = 0; I(Up,lg) = I(Dn,rg) = 5; L(lg) = L(rg) = 10 w.r.t "s".

Roughly speaking, an **inertia** represents the minimal number of occurrences of some "time reference event" before a pre-condition can be considered as holding. When the inertias are fulfilled for each pre-condition of an event, this event becomes "ready to occur". E.g: when the locomotive enters the crossing (occurrence of "pic"), 100 occurrences of m (meter) are required before event "poc" may happen.(I(Wc, poc) = 100 m) A **latency** indicates the maximal number of occurrences of some "time reference event" allowed before the effective occurrence of an event "ready to occur". E.g: when event "poc" is "ready to occur", it must occur within 200 m (L(poc) = 200 m) because of the maximal length of the train (200 m). Inertia and latency are general concepts, not restricted to our example (see [André88]).

II.3. CONSEQUENCES OF TIMING CONSTRAINTS

The dynamics of a concurrent system can be expressed by "(non-sequential) processes". Processes consider only causal dependencies expressed by the net. As stated in Para.II.2, the Railroad Crossing System is not contact-free and as such, processes cannot be directly derived. Some extension to the definition of processes will be convenient, at least as temporary representations of the evolutions. We will introduce "pseudo-processes" for this purpose.

This extension is not sufficient to take temporal dependencies into account. Annotations can be used to express those dependencies. If a unique time reference is assumed, then annotations could be intervals of non-negative integers augmented with a special symbol (ω) necessary to represent unbounded delays. However, the set $N_\omega = N \cup \{\omega\}$, with an extended addition, is not sufficient to capture multiform time. So, we will consider a richer algebraic structure: a dioid of sets of bags.

III. FORMAL DEFINITIONS

In this section we introduce the technical material (SENS, bags, graphs). Synchronized Elementary Net Systems (SENS for short) have been presented in [André88]. We recall definitions. Then, we introduce bags for the purpose of representing multiform temporal constraints. The computation of delays needs sets of bags and associated algebra. The formalism and notations are fairly complex, proofs are rather long and involve some straightforward parts. A technical report is devoted to this mathematical material [André89], proofs are not essential for this paper and they are omitted.We strive to give motivations and informal presentations, postponing complete definitions to annexes.

III.1. SYNCHRONIZED ELEMENTARY NET SYSTEMS

III.1.1. The Model

A synchronized elementary net system (SENS) is a triple
? = $<N, C_{in}, CS>$ where
$<N,C_{in}>$ is an elementary net system called the **(D1)**
underlying elementary net system of Σ;

CS is the **constraint structure** of Σ.

Let E be the set of events, B be the set of conditions and F be the flow relation of N.

The **constraint structure** is a 5-tuple
CS = < E_{Ref}, Ref, Inertia, Latency, Prior > where
E_{Ref} is a set (the set of Reference Events);
Ref : $E \rightarrow E_{Ref}$ is a partial function associating events with Reference events;
Inertia : $B \times E \times E_{Ref} \rightarrow N$ is the inertia (partial) function; **(D2)**
Latency : $E \times E_{Ref} \rightarrow N_{\omega}$ is the latency (partial) function ;
Prior $\subseteq E \times E$ is the priority relation .

Some restrictions are imposed [André88]:
- E_{Ref} and E are disjoint sets;
- the partial functions Inertia and Latency are such that:

$\forall b \in B \;\; \forall e \in E \;\; \forall h \in E_{Ref}$:
Inertia(b,e,h) is defined iff $h = Ref(e) \wedge (b,e) \in F$;
Latency(e,h) is defined iff $h = Ref(e)$;
- other restrictions are relative to priority and not necessary for this paper.

III.1.2. Change-of-state in SENS

For a given ENS, the concession for an event depends only on the case. For SENS, on the contrary, we have to take account of the occurrences of a synchronizing event.

Let c be a case, e an event whose reference event is H (i.e, Ref(e)=H).
e is said to be "**ready to occur**" at c *iff*
e has concession at c in the underlying ENS, and
each precondition b of e has held for at least Inertia(b,e,H) occurrences of H.
An event e which becomes "ready to occur" at c and such that **(D3)**
$\forall e' \in E$: e' Prior e $\Rightarrow \neg$ (e' ready to occur at c)
must
either occur ,**or** lose its concession } within Latency(e,H) occurrences of H
The occurrence of e changes the case of the underlying net in the usual way.

In this paper, we consider only systems without choice, so Prior is void and an event e which becomes ready to occur at c **must** occur within Latency(e,H) occurrences of H.

III.1.3. Pseudo-process

The evolutions of a SENS Σ = (N, C_{in}, CS), where N = (B,E;F), can be represented by forward conflict occurrence nets ([NiPlWi81],[André88]). (Ordinary) occurrence nets [BesFer86] are sufficient in the absence of choice. Let Σ'=(N,C_{in}) be the underlying ENS of Σ. An occurrence net K=(S,T;FF) can be considered as an "unfolding" of Σ'. A mapping p from ($S \cup T$) to ($B \cup E$) associates elements of K to events and conditions. For convenience, the pair (K,p) is represented by K, an element x of K is labelled by its image p(x). If Σ' is contact-free and if p meets special requirements (see [BesFer86]) then, (K,p) is called a **process** of Σ'. Unfortunately, the underlying ENS of $\Sigma 2$ is not contact-free, so we need something less restrictive than a process. We define **pseudo-process** by a constructive algorithm:

Let Σ' = (N, C_{in}) be an ENS, where N=(E,B;F).
Let K=(S,T;FF) be an occurrence net.
Let p be a mapping from ($S \cup T$) to ($B \cup E$).
(K,p) is a **pseudo-process** of Σ *iff*
K and p are given by a finite number of steps of the following construction:

step 0 (initialization):
$^{(0)}S = \{s_1,s_2,...,s_n\}$ where $n = |C_{in}|$; $^{(0)}p: {}^{(0)}S \rightarrow C_{in}$ is a bijection;
$^{(0)}T = \varnothing$; $^{(0)}FF = \varnothing$;

step k+1:
Let $^{(k)}K = (^{(k)}S, {}^{(k)}T, {}^{(k)}FF)$ where $^{(k)}S = \{s_1,s_2,....,s_m\}$;
$^{(k)}T = \{t_1,t_2,...,t_k\}$.
$^{(k+1)}K$ and $^{(k+1)}p$ are obtained as follows:
Let $S_M = \{s \in {}^{(k)}S \mid s^\circ = \varnothing\}$, $c = {}^{(k)}p(S_M)$;
Choose $e \in E$ which has concession in c;
Let $S_k = \{s_{m+1},s_{m+2},...,s_{m+n}\}$ where $n = |e^\circ|$; **(D4)**
$^{(k+1)}T = {}^{(k)}T \cup \{t_{k+1}\}$; $^{(k+1)}S = {}^{(k)}S \cup S_k$;
$^{(k+1)}p(t_i) = {}^{(k)}p(t_i)$ for $1 = i = k$; $^{(k+1)}p(t_{k+1}) = e$;
$^{(k+1)}p(s_i) = {}^{(k)}p(s_i)$ for $1 = i = m$;
$^{(k+1)}p$ restricted to S_k is a bijection from S_k to e°;
$^{(k+1)}FF = {}^{(k)}FF \cup \{(s,t_{k+1}) \mid s \in S_M \cap {}^{(k)}p^{-1}({}^\circ e)\} \cup \{(t_{k+1},s) \mid s \in S_k\}$.

Clearly, this construction is **context-preserving** (i.e $\forall t \in T$: $p({}^\circ t) = {}^\circ p(t) \wedge p(t^\circ) = p(t)^\circ$). The image of the min K (i.e $p(^{(0)}S) = C_{in}$) is a case of Σ. But we cannot assert that for any slice sl of K, p restricted to sl is injective. In K2, Fig.2, for example, there exists a slice where two S-elements are labelled by C1. When Σ' is contact-free, the pseudo-process (K,p) is a (ordinary) process of Σ'.

III.1.4. S-contraction of a net

Pseudo-processes are too detailed. Only event occurrences are relevant to delays. So, we use a special projection which hides conditions. This operation is called the S-contraction.

Let K= (S, T, F) be a net. G = (T, Γ) is the S-contraction of K *iff* : **(D5)**

$\forall x,y\in T$: [$(x,y)\in\Gamma \Leftrightarrow \exists s\in S: (x,s)\in F \wedge (s,y)\in F$] [(1)].

III.2. BAGS

Inertia and Latency are partial functions. Bags allow easier handling of temporal parameters.

III.2.1. Extension of N

Let S be a set; a **multiset** over S is a function $f{:}S\rightarrow N$ where N is the set of the non-negative integers. f is a finite multiset iff $\{s\in S \mid f(s)?0\}$ is finite. In this paper, S is always a finite set, so any multiset over S is finite.
We need a slightly broader definition that we shall call **bag**. N is enriched with two new elements α and ω such that $\forall n \in N : \alpha < n < \omega$. Let N' be the new set.

ω can be interpreted as +8, i.e an element greater than any integer. ω stands for unbounded temporal constraints, e.g the latency of "pb↓". α could be interpreted as -8, but because of its absorbing property with respect to addition it stands for undefined components.
N' is provided with extended addition and multiplication. They are the smallest operations which behave like addition and multiplication on N and satisfy the following laws:

$N' = N \cup \{\alpha, \omega\}$;

$\forall n\in N'$: $n+\alpha = \alpha+n = \alpha$;

$\forall n\in N_\omega$: $n+\omega = \omega+n = \omega$; **(D6)**

$\forall n\in N'$: $n*0 = 0*n = 0$

$\forall n\in N'\backslash\{0\}$: $\alpha*n = n*\alpha = \alpha$;

$\forall n\in N_\omega\backslash\{0\}$: $n*\omega = \omega*n = \omega$;

α is absorbent for + (any operand undefined implies result undefined); ω is absorbent for + on N_ω (prolonging an unbounded delay yields an unbounded delay).

III.2.2. Bags over a set

A **bag over S** is a function $b{:}S\rightarrow N'$. Call $_\beta S$ the space of bags over S and S its basis. Useful operations and relations on bags are induced pointwise by operations and relations on N'.

Let $f,g\in {}_\beta S$. Define

f O,< g $\Leftrightarrow (f?g) \wedge (\forall s\in S: f(s) = g(s))$

$\forall s\in S$: $(f \oplus g)(s) = f(s) + g(s)$. **(D7)**

$\forall n\in N', \forall s\in S$: $(n * f)(s) = n * f(s)$ (scalar multiplication)

Let $\underline{0}$ be the two-sided identity element for ⊕: $\underline{0}(s)=0$ for each $s\in S$.

Any finite bag f can be decomposed into a finite sum of terms f(s)* ^,s for any $s\in S$, where ^,s stands for the "singleton": ^,s(t) = (*if* t=s *then* 1 *else* 0). Where no confusion is possible we write s for ^,s and a bag is denoted as a sum.

---

[1] $\exists s\in S$: $s\in x^\circ\cap{}^\circ y \Leftrightarrow \exists s\in S: (x,s)\in F \wedge (s,y)\in F$

The changes of bases can be expressed by "multirelations" ([Winskel85]).

A **multirelation** from S_1 to S_2 is a function $\theta: S_1 \times S_2 \rightarrow N$.
Let f be a bag over S_1, θ be a multirelation from S_1 to S_2, **(D8)**
the **application** of θ to f is the bag θf over S2 such that:
$(\theta f)(s') = \Sigma_{sPS_1} \theta(s,s') * f(s)$ for all $s' \in S_2$.

III.2.3. Inertia and Latency in terms of bags

Inertias and latencies can be thought of as bags over E_{Ref}.

Let $I: B \times E \rightarrow {}_{\beta}E_{Ref}$ and $L: E \rightarrow {}_{\beta}E_{Ref}$ such that
$\forall b \in B, \forall e \in E$: **(D9)**
$I(b,e) =$ *if* $(b,e) \in F$ *then* Inertia(b,e,Ref(e))$*$Ref(e) *else* α ;
$\forall e \in E$: L(e) = Latency(e,Ref(e))$*$Ref(e) .

According to this definition, α is the value associated with non-existing arc. Thanks to α, the function I can be substituted for the partial function Inertia. Of course, those bags are special: they are "singletons".

III.2.4. Sets of bags

Operations over bags are not sufficient for the computation of delays. Sets of bags are necessary. Even for a finite set S, ${}_{\beta}S$ is infinite. So, we restrict our attention to finite sets of bags. Let $\wp_f({}_{\beta}S)$ be the set of all **finite** subsets of ${}_{\beta}S$.

The application of a multirelation to bags is extended, in a natural way, to sets of bags:

For any $F \in \wp_f({}_{\beta}S)$: $\theta F =_{def} \{\theta f \mid f \in F\}$.

For our purpose it is not necessary to distinguish all the sets of bags. We introduce an equivalence relation on $\wp_f({}_{\beta}S) \backslash \varnothing$ and we work on equivalence classes:

Let $F \in \wp_f({}_{\beta}S) \backslash \varnothing$. $(F,O,<)$ is a POset. $Max_{O,<}(F) =_{def} \{ f \in F \mid \neg \exists g \in F: f\ O,<\ g \}$ is defined for all F.

Define ˜ as follows:

$\forall F,G \in \wp_f({}_{\beta}S) \backslash \varnothing$: F˜G *iff* $Max_{O,<}(F) = Max_{O,<}(G)$ **(D10)**
Let ${}_{\beta\beta}S =_{def} (\wp_f({}_{\beta}S) \backslash \varnothing)/\tilde{}$ be the partition of $\wp_f({}_{\beta}S) \backslash \varnothing$ induced by ˜
The class of $F \in \wp_f({}_{\beta}S) \backslash \varnothing$ is denoted by $[F] \in {}_{\beta\beta}S$.

Remark: Inertias and latencies can be redefined as classes of sets of bags over E_{Ref}.

$I: B \times E \rightarrow {}_{\beta\beta}E_{Ref}$ and $L: E \rightarrow {}_{\beta\beta}E_{Ref}$ **(D11)**

We assume this definition in what follows.

In order to improve legibility, wherever no confusion is possible, we will omit square brackets and denote singleton without braces (i.e $\underline{\alpha}$ stands for $[\{\underline{\alpha}\}]$, $\underline{\omega}$ for $[\{\underline{\omega}\}]$ where $\forall s \in S$: $\underline{\alpha}(s)=\alpha$, $\underline{\omega}(s)=\omega$, and {f,g} for [{f,g}]). We are mainly interested in non-α components, we call them "**effectively defined components**", for convenience. A bag without α-component is said to be a "**effectively defined bag**".

III.2.5. Algebra over classes of sets of bags

Taking ββS as carrier, it is possible to obtain a useful algebra, namely a **dioid**. A dioid is an algebraic structure with two internal operations, weaker than a ring (see ANNEX I), but with interesting properties.

Let □□and□□ be operations on ββS :

$\forall$[F],[G]$\in$ββS: [F] □ [G] $=_{def}$ [F $\cup$ G]

$\forall$[F],[G]$\in$ββS: [F] □ □[G] $=_{def}$ [$\cup_{fPF,\ gPG}$ {f$\oplus$g}] **(D12)**

Let Ξ=[{0}] be the identity element for □ □and□

□ℍ= [{α}] be the identity for □.

Proposition P1

(ββS,□,□ ,ℍ,Ξ) is a (commutative) dioid. **(P1)**

Sets of bags will be shown useful to express delays in pseudo-processes. Introducing sets of bags instead of integers, is an essential contribution of this paper.

IV. DELAYS IN PROCESSES

IV.1. INFORMAL PRESENTATION

The ENS Σ2 is a model for the Railroad crossing system where the train can run forever. Let N2 be the net of Σ2. In N2 there is no shared place, so N2 is a T-graph [BesFer86]. Σ2 is a system without choice. This particular case is sufficient for introducing our notion of delays and moreover the analysis will be easier.

We have assigned some temporal constraints to this system (para.II.3).

The constraint structure may reduce the concurrency, i.e pairs of concurrent events of the pseudo-process may be changed into ordered pairs. We suppose that a pseudo-process of Σ2 is given. We explain how to take account of the temporal constraints.

IV.2. AIM

Let π=(K,p) be a pseudo-process of Σ2 (e.g Fig.2) where K = (S, T, F). For x and y belonging to T, we would like to have a relation giving date(y) when date(x) is known.

In a previous paper [André86] we defined a date (relative to some reference event) as an interval of N_ω. min date(x) was the "earliest occurrence date" and max date(x) was the "latest occurrence date" for x. In this paper, we need a more general definition, which captures multiform time (or reference).

> date : T → $\beta\beta E_{Ref} \times \beta\beta E_{Ref}$.
> The first component is referred to as the **Earliest** (occurrence) date, **(D13)**
> the second as the **Latest** (occurrence) date.

For given x∈T, let date(x) = (m,M). If H=Ref(p(x)) then the N-interval [m(H), M(H)] corresponds to the former definition of date(x). But, for H'∈ E_{Ref}, m(H') = 0 and M(H') = ω means that occurrences of p(x) and H' are non-correlated.

We search for a relation of the kind:

> date(y) = date(x) □ delay(x,y)
> where delay : T × T → $\beta\beta E_{Ref} \times \beta\beta E_{Ref}$ **(D14)**
> The first component of delay is called the **S**-component (**S** for Shortest),
> the second is called the **L**-component (**L** for Longest).

Delay(x,y) binds the earliest and latest occurrence dates of x and y as follows:

> Earliest.date(y) = Earliest.date(x) □ S.delay(x,y) ∈ $\beta\beta E_{Ref}$
> Latest.date(y) = Latest.date(x) □ L.delay(x,y) ∈ $\beta\beta E_{Ref}$ **(D15)**

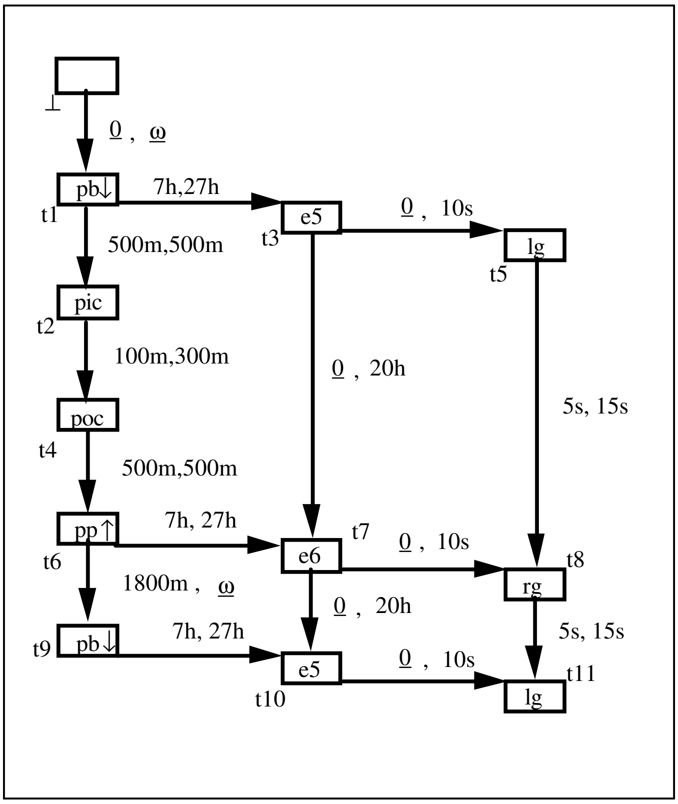

FIGURE 3: Event graph of (K2,p).

Delay(x,y) has effectively defined values (non-α values) if e=p(x) is a **cause** and e'=p(y) an **effect**, thus x precedes y (i.e there exists a path from x to y) in K. If x precedes y, in the absence of known "temporal" constraints, delay(x,y) = ($\underline{0}$,$\underline{\omega}$). Constraints allow natural number components. Delay(x,y) = ($\underline{\alpha}$,$\underline{\alpha}$) stands for undefined delay. Recall that α has been introduced in order to avoid dealing with partial functions.

We require for delay function to have the **additive property**:

$$x,y,z \in T: \text{delay}(x,y) = \text{delay}(x,z) \,\square\, \text{delay}(z,y) \qquad \textbf{(D16)}$$

As (pseudo-)processes are suitable for expressing causal dependencies, we take them as our starting point. For a given SENS Σ, let π=(K,p) be a pseudo-process of Σ. Given the occurrence net K, we build up a graph G'_π = (T,Γ') which is the S-contraction of K on T. As K is acyclic, so is G'_π.

Let G_π=(T',Γ) be the graph obtained from G'_π by creating a root (1) called "$\perp$" and where T'=T$\cup${$\perp$}. We call G_π **the event graph associated to the process π**. The subscript π is omitted where no confusion is possible. Fig.3 is the event graph associated with (K2,p) (Fig.2). The inscriptions next to arcs are related to temporal constraints, they will be explained in Para.IV.4.2.

IV.3. RECURSIVE DEFINITION OF DELAYS

We start with the firing rule of a SENS. Let e be an event of a SENS Σ, let π=(K,p) be a pseudo-process of Σ, and let y be a transition of K such that e=p(y). The date of occurrence of y is deduced from the dates of occurrence of each immediate predecessors of y ($\{z_k \in \Gamma^{-1}_y\}$) by:

$$\text{date}(y) = \max \{ \text{date}(z_k) \,\square\, (i_k, i_k \square l) \mid z_k \in \Gamma^{-1}_y \} \; {}^{(2)} \qquad \textbf{(D17)}$$
where $i_k = I(p(s_k),p(y))$, $l = L(p(y))$ and $\{s_k\} = z_k{}^\circ \cap {}^\circ y$.

We extend subtraction to N' as follows:

$\forall n \in N'$: α-n = n-α = α; $\forall n \in N' \backslash \{\alpha\}$: ω-n = ω;

$\forall n,n' \in N$: n - n' = *if* n = n' *then* n - n' *else* α.

i.e, the subtraction is defined if the result of the subtraction is positive or null, it is undefined (equal to α) otherwise. This subtraction induces a subtraction over βS and over $\beta\beta$S.

Let x,y $\in$T', from D17: date(y) $\square$ ~date(x) = max {date(z_k) $\square$ (i_k, $i_k \square$l) |$z_k \in \Gamma^{-1}_y$ }$\square$~date(x) (3) $\Rightarrow$ delay(x,y) = max{date(z_k)$\square$~date(x)$\square$(i_k, $i_k\square$l) | $z_k \in \Gamma^{-1}_y$ } = max{delay(x,z_k) $\square$ (i_k, $i_k\square$l) | $z_k \in \Gamma^{-1}_y$ }. This relation holds when each delay(x,z_k) has effectively defined values, otherwise, delay(x,y) should be ($\underline{\alpha}$,$\underline{\alpha}$).

If delay(x,z_k) is effectively defined for each $z_k \in \Gamma^{-1,y}$
then delay(x,y) = max { delay(x,z_k) $\square$ (i_k, $i_k\square$l) | $z_k \in \Gamma^{-1,y}$ } **(D18)**

---

1 See ANNEX II, Definition D26 for root and rooted graph.

2 max { (a_i,b_i) | i$\in$I } = (max {a_i | i$\in$I }, max {b_i | i$\in$I })

3 (~date(x) is the opposite of date(x) w.r.t $\square$): ~date(x) = (-Earliest.date(x), -Latest.date(x)).

Equation D18 is similar to the one that characterizes the "longest" paths in a weighted graph. Note that "longest path" stands for "path whose weight is the largest". In fact, Eq.D18 corresponds to two instances of the longest path problem:

$$\text{S.delay}(x,y) = \max \{ \text{S.delay}(x,z_k) \,\square\, i_k \mid z_k \in \Gamma^{-1,y} \}$$
$$\text{L.delay}(x,y) = \max \{ \text{L.delay}(x,z_k) \,\square\, i_k \,\square\square \mid z_k \in \Gamma^{-1,y} \} \qquad \textbf{(D19)}$$

Solutions, to the problem of the longest path in a rooted graph between the root and each node, are well-known (PERT, for example). Our problem is harder because we are interested in the longest path between any pair of vertices and moreover lengths are complex objects (classes of sets of bags) instead of numbers. The **algebra of dioids** is a convenient framework for our purpose. Note that α-delays require a special attention.

IV.4. PATH ALGEBRA

IV.4.1. Generalized incidence matrix and quasi-inverse

Many problems in graphs can be solved by an algebraic approach. The reader is urged to refer to dedicated books ([GonMin85], for example). We only recall general results, further details are available in referenced books for fundamentals, and in [André89] for special applications to delays.

Given a weighted graph (G,W) [1], where weights are elements of a dioid, for any vertices i and j of G, let C_{ij} be the set of paths from i to j. Let $L_{ij} = \Sigma_{\mu P C_{ij}}$ W(μ), for all i,j. L_{ij} can be obtained as the entry ($a^{*,ij}$) of a matrix A* called the **quasi-inverse** of the **generalized incidence matrix** A of the graph G [GonMin85]. A=($a_{i,j}$) is such that $a_{i,j}$ = *if* (i,j)∈Γ *then* W(i,j) *else* 0-element of the underlying dioid. Let $A^{(k)} =_{def} U + A + A^2 + ... + A^k$ where U is the unit matrix, then $A^* = \lim_{k \to 8} A^{(k)}$. For a circuit-free graph with N vertices, there exists a natural number k<N such that $A^* = A^{(k)} = A^{(N-1)} = A^{(N)} = ...$. This property holds for any dioid [GonMin85]. Therefore, using generalized matrix products, we can compute A^*.

In our case, elements of matrices are classes of sets of bags (elements of $\beta\beta E_{Ref}$) and the dioid is ($\beta\beta E_{Ref}$,□,□ ,□□). Let $M_N(\beta\beta E_{Ref})$ be the set of square matrices of order N and with elements in $\beta\beta E_{Ref}$. □ and □ induce operations on $M_N(\beta\beta E_{Ref})$. For any A=(a_{ij}) and B=(b_{ij}) belonging to $M_N(\beta\beta E_{Ref})$:

$C = A \,\square\, B = (c_{ij}) \Leftrightarrow_{def} c_{ij} = a_{ij} \,\square\, b_{ij} \quad \forall i,j: 1..N$

$C = A \,\square\, B = (c_{ij}) \Leftrightarrow_{def} c_{ij} =$ **Erreur!**$\forall$i,j: 1..N

The zero-matrix Z is such that ∀i,j: z_{ij} = □and the unit-matrix U is such that ∀i,j i?j: u_{ij} = □and ∀i: $u_{i,i}$=□). Then, ($M_N(\beta\beta E_{Ref})$, □,□ , Z, U) is a dioid. ($\beta\beta E_{Ref}$,□,□ ,□□) being of the kind (Max, +), $a^{*,ij}$ is the **maximal weight of paths** from i to j.

Algorithm:

---

[1] G=(V,Γ) is a graph and W:Γ→S a weighting function, where (S,+,*,zero,unit) is some dioid.

Let (G,W) be a rooted weighted graph, where $G = (V, \Gamma)$ and $\perp$ be the root of G. h is the height function of the acyclic graph.

Let $A = (a_{ij})$ be the generalized incidence matrix for G.

Initialization: ${}^{(1)}V = V \setminus \{\perp\}$ and ${}^{(1)}a_{1,1} = \underline{0}$;

$(k+1)^{th}$ step: let $v_{k+1} \in \{ v \in {}^{(k)}V \mid h(v) \text{ is minimal}\}$;

${}^{(k+1)}V = {}^{(k)}V \setminus \{v_{k+1}\}$, ${}^{(k+1)}a_{k+1,k+1} = \underline{0}$;

for i:=1 *to* k *do* **(A1)**

${}^{(k+1)}a_{i,k+1} =$ **Erreur!***if* h(i)<h(j) ;

${}^{(k+1)}a_{i,k+1} = \underline{\alpha}$ *otherwise*.

IV.4.2. Application to delays

Principle:

To analyse a SENS, first we build a "pseudo-process", which represents possible evolutions without temporal constraints. Then, those constraints are integrated in the pseudo-process as weighting functions.

The calculations for the Shortest and the Longest delays are made on different weighted graphs: the arcs of the event graph are weighted by sets of bags expressing:

- inertias for the Shortest delays,
- inertias **and** latencies for the Longest delays.

"Longest paths" in the weighted graphs are taken as first approximations to delays. We propose an algorithm (A1) which gives for each pair of nodes, say (x,y), a set of bags. A bag stands for the weight of a path from x to y. But the actual "length" is the maximal "length" over all possible paths from x to y. Since bags are only partially ordered, we have got a set of bags, instead of a single bag. For efficiency, those sets are reduced while they are built: when a bag is "less than" another one, there is no need to keep it. That is the reason why we work on equivalence classes of sets of bags.

In para IV.5, we will explain how to deduce delays from "length" of paths.

Let G_π be the event graph associated to a pseudo-process π=(K,p). From G_π we derive two weighted graphs: (G,Wi) and (G,Wl) where the weighting functions are defined as follows:

- $\forall x,y \in T'$, x?y: let e=p(y) Wi(x,y)=I(p(s),e); Wl(x,y)=Wi(x,y) □ L(e) where $\{s\} = x^\circ \cap {}^\circ y$ in K (Note that, if s does not exist then Wi(x,y) = $\underline{\alpha}$);
- $\forall x \in T'$: Wi(x,x) = Wl(x,x) = $\underline{0}$, by convention.

Denote by LI(x,y) (LL(x,y), resp.) the "longest" path from x to y in (G,Wi) (in (G,Wl), resp.). Let A_i (A_l, resp.) be the generalized incidence matrix of (G,Wi) ((G,Wl), resp.).

Referring to the event graph (K2,p) (Fig.3) we have:

A_i	⊥	t1	t2	t3	t4	t5	t6	t7	t9	t8
⊥	0	0	α	α	α	α	α	α	α	α
t1	α	0	500m	7h	α	α	α	α	α	α
t2	α	α	0	α	100m	α	α	α	α	α
t3	α	α	α	0	α	0	α	0	α	α
t4	α	α	α	α	0	α	500m	α	α	α
t5	α	α	α	α	α	0	α	α	α	5s
t6	α	α	α	α	α	α	0	7h	1800m	α
t7	α	α	α	α	α	α	α	0	α	0
t9	α	α	α	α	α	α	α	α	0	α
t8	α	α	α	α	α	α	α	α	α	0

A_l	⊥	t1	t2	t3	t4	t5	t6	t7	t9	t8
⊥	0	ω	α	α	α	α	α	α	α	α
t1	α	0	500m	27h	α	α	α	α	α	α
t2	α	α	0	α	300m	α	α	α	α	α
t3	α	α	α	0	α	10s	α	20h	α	α
t4	α	α	α	α	0	α	500m	α	α	α
t5	α	α	α	α	α	0	α	α	α	15s
t6	α	α	α	α	α	α	0	27h	ω	α
t7	α	α	α	α	α	α	α	0	α	10s
t9	α	α	α	α	α	α	α	α	0	α
t8	α	α	α	α	α	α	α	α	α	0

Applying algorithm A1 that computes the quasi-inverse of A_i and A_l, we obtain:

$A^{*,i}$	⊥	t1	t2	t3	t4	t5	t6	t7	t9	t8
⊥	**0**	**0**	**500m**	**7h**	**600m**	**7h**	**1100m**	**1100m +7h**	**2900m**	**{(7h+5s), (1100m+7h)}**
t1	α	**0**	**500m**	**7h**	**600m**	**7h**	**1100m**	**1100m +7h**	**2900m**	**{(7h+5s), (1100m+7h)}**
t2	α	α	**0**	α	**100m**	α	**600m**	600m +7h	**2400m**	600m +7h
t3	α	α	α	**0**	α	**0**	α	0	α	5s
t4	α	α	α	α	**0**	α	**500m**	500m +7h	**2300m**	500m +7h
t5	α	α	α	α	α	**0**	α	α	α	5s
t6	α	α	α	α	α	α	**0**	7h	**1800m**	7h
t7	α	α	α	α	α	α	α	**0**	α	0
t9	α	α	α	α	α	α	α	α	**0**	α
t8	α	α	α	α	α	α	α	α	α	**0**

$A^{*,l}$	⊥	t1	t2	t3	t4	t5	t6	t7	t9	t8
⊥	**0**	ω	ω	ω	ω	ω	ω	ω	ω	ω
t1	α	**0**	**500m**	**27h**	**800m**	**27h +10s**	**1300m**	**{47h, (1300m +27h)}**	ω	**{(27h+25s), (1300m+27h+10s) , 47h+10s }**
t2	α	α	**0**	α	**300m**	α	**800m** +27h	800m	ω	800m+ 27h+10s
t3	α	α	α	**0**	α	**10s**	α	20h	α	{25s, 20h+10s}
t4	α	α	α	α	**0**	α	**500m**	500m +27h	ω	500m+27h +10s
t5	α	α	α	α	α	**0**	α	α	α	15s
t6	α	α	α	α	α	α	**0**	27h	ω	27h+10s
t7	α	α	α	α	α	α	α	**0**	α	10s
t9	α	α	α	α	α	α	α	α	**0**	α
t8	α	α	α	α	α	α	α	α	α	**0**

Above-mentioned results are only longest paths computed on particular weighted graphs. Now, **we have to interpret these results in terms of delays**.

IV.5. FUNDAMENTAL PREDECESSORS

In para.IV.3, we explained that delay(x,y) could be computed as a "longest path" if delay(x,z) had effectively defined value for each z immediate predecessor of y. Due to the recursive definition D18, this property must be true for any immediate predecessor of z, as well. So, we introduce a binary relation on T' called the "fundamental predecessor" relation.

> $\forall x,y \in T'$: x is a **fundamental predecessor** of y in G
> (written x **fund_pred**$_G$ y) *if and only if* **(D20)**
> x belongs to any path from the root ($\perp$) to y in G.
> Let $FUND_G(y) =_{def} \{ x\in T' \mid x$ **fund_pred**$_G$ $y \}$

$FUND_G(y)$ is the set of the fundamental predecessors of y in G, it can be easily computed (see Algorithm A2, in ANNEX II).
A2 applied to $G_{\pi 2}$ yields (in this order):

$FUND_G(\perp)=\{\perp\}$;
$FUND_G(t1)=\{\perp,t1\}$;
$FUND_G(t2)=\{\perp,t1,t2\}$; $FUND_G(t3)=\{\perp,t1,t3\}$;
$FUND_G(t4)=\{\perp,t1,t2,t4\}$; $FUND_G(t5)=\{\perp,t1,t3,t5\}$;
$FUND_G(t6)=\{\perp,t1,t2,t4,t6\}$;
$FUND_G(t7)=\{\perp,t1,t7\}$; $FUND_G(t9)=\{\perp,t1,t2,t4,t6,t9\}$;
$FUND_G(t8)=\{\perp,t1,t8\}$; $FUND_G(t10)=\{\perp,t1,t10\}$;
$FUND_G(t11)=\{\perp,t1,t11\}$.

The relation **fund_pred**$_G$ has a nice property with respect to delay (see property P2 in the next para.). Entries of matrices $A^{*,i}$ and $A^{*,l}$ associated with vertices in the **fund_pred**$_G$ relation have been written in bold characters.

IV.6. FIRST APPROACH TO DELAYS

If $x \in FUND_G(y)$, the subgraph whose vertices are members of $[x,y]_G$ [1] is a rooted weighted graph, the root of which is x. The quasi-inverse $A^{*,i}$ of A_i ($A^{*,l}$ of A_l, resp.) gives LI(x,z) (LL(x,z), resp.) for each $z\in[x,y]_G$ and delay(x,z) is effectively defined. Therefore, for each $z\in[x,y]_G$, $x \in FUND_G(y)$: delay(x,z) = (LI(x,z), LL(x,z)), i.e S.delay(x,z) = LI(x,z) and L.delay(x,z) = LL(x,z). In particular, for any vertex y which has only one immediate predecessor x, we have $x \in FUND_G(y)$, thus delay(x,y) is given by (LI(x,y), LL(x,y)) = (Wi(x,y),Wl(x,y)).

Consider now, vertices with several immediate predecessors. Let $z,z'\in\Gamma^{-1,y}$ and $x\in FUND_G(y)$. LI(x,y) = $Max_{z\in\Gamma^{-1,y}}$ {LI(x,z) □ LI(z,y)}. If there is no path from z to z', then LI(z,z') = $\underline{\alpha}$ and LI(z,y) takes no account of z'. For $x\in FUND_G(y)$ and $z\in[x,y]_G$, LI(x,y) = S.delay(x,y) = $Max_{z\in\Gamma^{-1,y}}$ {S.delay(x,z) □

---

[1] For $G=(V,\Gamma)$ and $V_1,V_2 \subseteq V$: $[V_1,V_2]_G =_{def} \{ z\in V \mid \exists x\in V_1\ \exists y\in V_2\ \ (x,z),(z,y)\in\Gamma^* \}$

LI(z,y)} and LI(x,z) = S.delay(x,z). So, S.delay(x,y) = S.delay(x,z) □ LI(z,y). But, in order to respect the additivity of delays (D16), we have S.delay(x,y) = S.delay(x,z) □□S.delay(z,y). Hence, LI(z,y) = S.delay(z,y). In a similar way, we establish LL(z,y) = L.delay(z,y). This discussion can be summed up by the following proposition:

proposition:

$\forall x,y \in T'$: LI(x,y) = S.delay(x,y) $\wedge$ LL(x,y) = L.delay(x,y)
if $x \in$ FUND$_G$(y) *then* delay(x,y) = (LI(x,y), LL(x,y)) **(P2)**

Note that using additivity, it is possible to deduce delay for vertices not in the fundamental predecessor relation.

IV.7. USES OF DELAYS

The constraint structure of the SENS may reduce concurrency. This is not surprising if one remembers that "temporal" constraints stand for hidden causal dependencies (see [André88]).

Consider $x,y,z \in T'$ such that L.delay(z,x) < S.delay(z,y). According to D15:

•Earliest.date(y) = Earliest.date(z) □□S.delay(z,y)

•Latest.date(x) = Latest.date(z) □□L.delay(z,x)

Let e = p(z) be the event associated with z in the pseudo-process; let H=Ref(e) be the Reference event for e; assume that e occurs between the k^{th} and the $(k+1)^{th}$ occurrence of H, we have [André86] Earliest.date(z) = Latest.date(z) (= k*H).

Hence, Latest.date(x) < Earliest.date(y) and x occurs definitely before y, we will say "x temporally precedes y in G".

This type of precedence is induced by temporal constraints and can be formalized as follows.

The event graph associated with a process can be thought of as a Hass diagram for a POset of transitions. Let **prec**$_G$ = Γ^+ be the order (**prec** is short for precede). Unordered pairs of transitions are said to be concurrent. Let **co**$_G$ be the co-relation of the POset.

Let G be the event graph associated to a pseudo-process (K,p) of a SENS.
$x,y \in T'$, x **temp_prec**$_G$ y (read x temporally precedes y in G) *iff* **(D21)**
x **co**$_G$ y and $\exists z \in T'$: L.delay(z,x) < S.delay(z,y);
Let **before**$_G$ =$_{def}$ **prec**$_G$ $\cup$ **temp_prec**$_G$.

Graphically, if x temporally precedes y, then we add a dotted arc from x to y in G.

The temporal precedence has been used in a previous paper to prove that some potential conflicts were resolved in a unique way ([André88]). More generally, temporal precedence can **prove that some potential cases are not reachable**. In the railroad crossing system, for example, the subcase {Wc, Up} is dangerous and must be avoided. The safety of the crossing is guaranteed if t5 **before**$_G$ t2 (i.e lg occurs before pic) and t4 **before**$_G$ t8 (i.e poc occurs before rg). The latter condition is met because t4 **prec**$_G$ t8 in the event graph. Since t2 **co**$_G$ t5, the former condition will be fulfilled if t5 **temp_prec**$_G$ t2, i.e L.delay(t1,t5) < S.delay (t1,t2) $\Leftrightarrow$ 27h+10s < 500m.

Without known relationships between the reference events (m, h, s), the two bags are not comparable and we cannot conclude.

Suppose that the controller clock cycle is 1µs and that the maximal train speed, in the crossing section, is 90 km/h (25 m/s). These hypotheses can be translated into multirelations from {m, h, s} to {h}: $\theta_{min} = (4.10^4, 1, 10^6)$ and $\theta_{max} = (\omega, 1, 10^6)$. There is no need to compute delays again. It is sufficient to apply the multirelation θ to delays:

θ_{max} (L.delay(t1,t5)) = $(27*1 + 10*10^6) *h$ = 10 000 027 h;

θ_{min} (S.delay(t1,t2)) = $(500*4.10^4) *h$ = 20 000 000 h.

Therefore, t2 **temp_prec**$_G$ t5, and the gate is closed whenever a train is in the crossing.

Up to now, fundamental predecessors play an important role in the evaluation of delays. However, few vertices are in the fundamental predecessor relation. We need a weaker relation.

IV.8. DOMINANT PREDECESSOR

When a transition has several direct predecessors, none of them can be a fundamental predecessor. Consider t7: $\Gamma^{-1,t7}$ ={t3, t6}, t1 **fund_pred**$_G$ t7, p(t7)=e6, p(t1)=pb↓ and °e6 = {C3, C6}. With respect to the date of occurrence of pb↓,

- the latest "precondition C6 is met for e6" is L.delay(t1,t3) □ $\underline{0}$ = 27h;
- the earliest "precondition C3 is met for e6" is S.delay(t1,t6) □ 7h = 1100m + 7h.

Using θ, we have: 27h = L.delay(t1,t3) □ $\underline{0}$ << S.delay(t1,t6) □ 7h = 44 000 007h. Thus, when C3 has been holding for a duration equal to I(C3,e6), the precondition C6 has been already holding for a duration greater than I(C6,e6). It works out as if C3 was the sole precondition for e6, i.e t6 was the only immediate predecessor of t7. t6 is called the **dominant predecessor** of t7 in G.

IV.8.1. Definition

$\forall y \in T'$, $z \in \Gamma^{-1,y}$ is a dominant predecessor of y in G
(written z **dom_pred**$_G$ y) *iff* **(D22)**
$\exists x \in T'$: $\forall z' \in \Gamma^{-1,y}$ z'?z $\Rightarrow$ L.delay(x,z')□ Wi(z',y) < S.delay(x,z)□ Wi(z,y).

A dominant predecessor can play the same role as a fundamental predecessor:

IV.8.2. Proposition

if z **dom_pred**$_G$ y *then* delay(z,y) = (Wi(z,y),Wl(z,y)). **(P3)**

Proof: (see ANNEX II). □

IV.8.3. Application

Using temporal precedence we can prove, also, that **the system is contact-free**. The case {Tr, C1, C5} is reachable in the underlying ENS of Σ2. At this case, there is a contact situation for pb↓. To prove that this case is not reachable we establish that t3 **before**$_G$ t9 (i.e the holding of C1 ceases before another train arrives at the crossing). In fact, a stronger relation holds: t8 **before**$_G$ t9. We prove that, using dominant predecessors:

L.delay(t1,t5) □ 5s = (27h +10s) □ 5s = 27h + 15s $\rightarrow_{\theta_{max}}$ 15 000 027h; S.delay(t1,t7) □ $\underline{0}$ = 1100m + 7h $\rightarrow_{\theta_{min}}$ 44 000 007h, which implies t7 **dom_pred**$_G$ t8. Now, t6 **dom_pred**$_G$ t7, so delay(t6,t8) = (7h □ $\underline{0}$, 27h □□10s) = (7h, 27h + 10s), delay(t6,t9) = (1800m, $\underline{\omega}$). θ_{max} (L.delay(t6,t8)) = 10 000 027h < θ_{min} (S.delay(t6,t9)) = 72 000 000h ⇒ t8 **temp_prec**$_G$ t9. □

IV.9. ESSENTIAL PREDECESSOR

For a vertex y, we would have to memorize its fundamental predecessor set (FUND$_G$(y)) and its dominant predecessor, if defined. In fact, it is possible to associate only one vertex to y, in order to obtain delay from any predecessor of y to y. This vertex is called the essential predecessor of y.

Definition

The **essential predecessor** of y in G is
if ∃z∈T': z **dom_pred**$_G$(y) *then* z **(D23)**
else x such that x∈FUND$_G$(y)\{y} ∧ h(x) is maximal.

Refer to [André90], for details.

For Σ2, under the relationships summarized by (θ_{min},θ_{max}) (para.IV.7), we have for any vertex y whose essential predecessor is x:

y	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11
x	⊥	t1	t1	t2	t3	t4	t6	t7	t6	t9	t10
E.delay(x,y)	$\underline{0}$	500m	7h	100m	$\underline{0}$	500m	7h	$\underline{0}$	1800m	7h	$\underline{0}$
L.delay(x,y)	$\underline{\omega}$	500m	27h	300m	10s	500m	27h	10s	$\underline{\omega}$	27h	10s

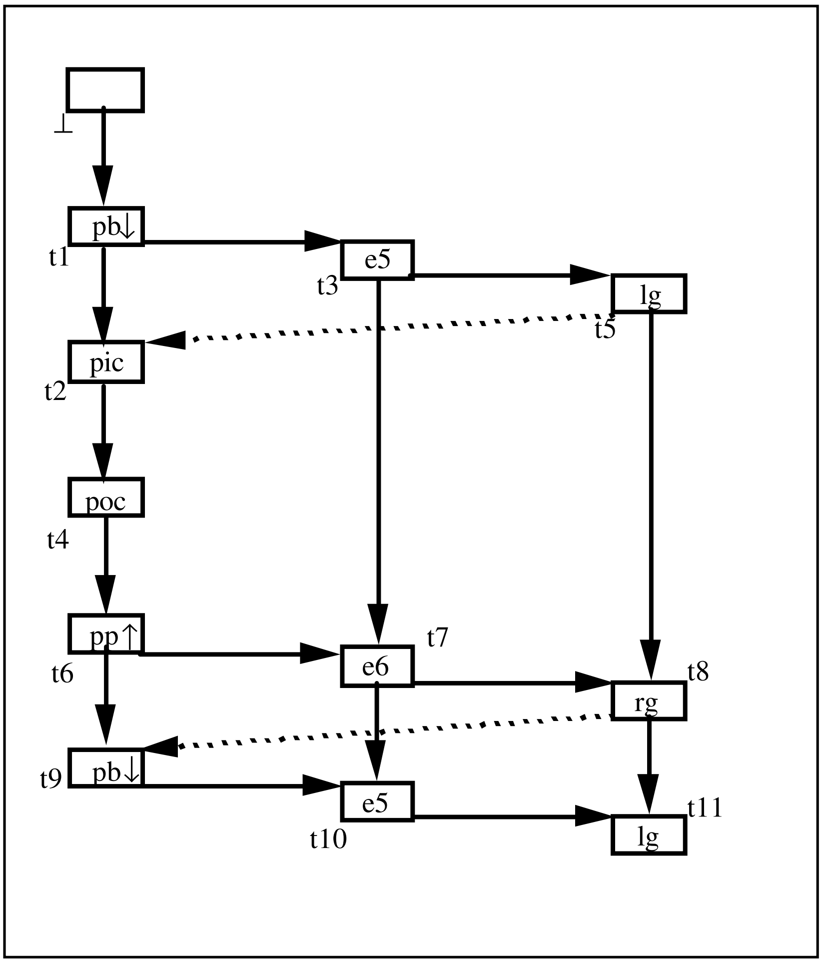

FIGURE 4: Event graph with induced precedences.

Fig.4 is the event graph with induced precedences. In fact, the relation **before**$_G$ is a total order over T'. The events occur in a strict sequential order: (pb↓.e5.lg.pic.poc.pp↑.e6.rg)* which is the expected behaviour for the Railroad Crossing System.

V. CONCLUSION

In this paper, we have presented an example of concurrent system with "multiform time" constraints. The **Synchronized Elementary Net Systems** (SENS) [André88] have been taken as a model. The behaviour of a SENS has been represented by a **pseudo-process** which is an unfolding of the underlying Elementary Net System [Thiag86]. Starting with pseudo-process which stands for possible evolutions without temporal constraints, we integrate the constraints through weighting functions. We have proposed **algorithms** that compute delays between event occurrences. A similar approach can been found in [Best88]. In his paper, E.Best considers "Basic Net Systems", which are more general than ENS, but which remain safely marked nets. Weighting functions are introduced in order to calculate the running time of a process. The Weighted Basic Net model resembles various "Timed Petri Nets" models, but it does not cover "Time Petri Nets" from which SENS are derived. Moreover, for us, timing inscriptions modify the behaviour of the system. Therefore, the two methods have, in common, the same background (processes and calculations of weighted paths in acyclic graphs), but they differ on their objectives and their algorithmics.

Moreover, in order to capture multiform time, a particular algebraic structure has been introduced, namely, **a dioid whose elements are sets of bags**.

Constraints, through the delays, induce **"temporal" precedences**. So, the partial ordering of event occurrences is strengthened and the forward case class is reduced. This alteration of the behaviour of the system has been used to prove that some undesirable cases are effectively discarded by the timing constraints.

Analyses of SENS, based on pseudo-processes are **dynamic methods**, they need some kind of "simulation" of the net. For **Time Petri Nets** ([Merlin75]), from which SENS are derived, the enumerative approach proposed in [MeBe83] is a dynamic method, too. The main drawback of dynamic methods is that, generally, they lead to tremendous computations when applied to real-world systems. In fact, the computation of delays presented in this paper is an effective analysis method for SENS insofar as processes can be used to analyze ENS. Using bags or sets of bags is not really complex; the SENS algebra can be implemented as an abstract data type. The computational weakness of our method is that it is based upon processes. Therefore, its application could be restricted to theoretical issues. The main contribution of this paper is **conceptual**: we attempt to enrich General Net Theory with material for Petri nets with time.

Static methods, such as S-invariants, can establish some properties in a faster and more elegant way. For subclasses of (deterministic) **Timed Petri Nets** ([Ramch74]) several results can be directly inferred from the net structure ([Sifak80], [ChrCar84], [HilLev87],..). As Timed Petri Nets are generally used for performance evaluations, they apply the "earliest firing rule", which is only a special case for Time Petri Nets (and SENS). Nevertheless, we believe that the **algebra of bags is a suitable ground for static analysis methods of SENS**. We are confirmed in our opinion by a recent paper ([CMQV89]) where the authors have shown that Timed Petri Nets without shared places could be considered as "linear systems" when some particular algebras were used. Theses algebras all fall into the axiomatic definition of dioids. Investigations on this topic are in progress.

VI. ACKNOWLEDGEMENTS

I would like to thank the anonymous referees for their valuable comments. They pointed out the necessity for intuitive presentation of mathematical concepts. These remarks have greatly helped me to improve this paper. I hope that both designer and mathematically inclined readers may find interest in this new approach of time in nets.

VII. REFERENCES

[ACPN86] **Advanced Course on Petri nets:***Application and Relationships to Other Models of Concurrency.* LNCS 255, Springer-Verlag 87.

[André86] **C.ANDRE**: *A semantics of timed Petri nets in terms of low level Petri nets. 7th European Workshop* on Petri Nets, OXFORD (UK), 86, 115-130.

[André88] **C.ANDRE**: *Synchronized Elementary Net Systems.* 9th European Workshop on Petri nets, VENICE (I), June 88, published in Advances in Petri Net 1989, LNCS n°424 , Springer-Verlag 1990.

[André89] **C.ANDRE**: *An Algebra for SENS.* Technical Report, I3S-LASSY, NICE, December 1989.

[André90] **C.ANDRE**: *Delays in Synchronized Elementary Net Systems (Extended version).* Technical Report, I3S-LASSY, TR90-1, NICE, January 1990.

[BeCo85] **G.BERRY, L.COSSERAT**:*The Synchronous Programming Language ESTEREL and its Mathematical Semantics.* Seminar on Concurrency, Springer-Verlag, LNCS 197, 1985.

[BesFer86] **E.BEST, C.FERNANDEZ**: *Notation and Terminology of Petri Net Theory.* Arbeitspapiere der GMD, 195, Jan 1986.

[Best88] **E.BEST**: *Weighted Basic Petri Nets.* Concurrency'88, LNCS n° , Springer-Verlag 1988, pp 257- 276.

[ChrCar84] **P CHRETIENNE, J CARLIER**: *Modelling scheduling problems with Timed Petri Nets* Advances in Petri Nets 1984, LCNS n°188, Springer-Verlag, 1984, pp62-82.

[CMQV89] **G COHEN, P MOLLER, J.P QUADRAT, M VIOT**: *Evaluation of Discrete Event Systems.* Proceedings of the IEEE, vol 77, n°1, Jan 89, pp39-58.

[GonMin85] **M.GONDRAN, M.MINOUX:** *Graphes et algorithmes*. Collection de la Direction des Etudes et Recherches d'Electricité de France, n°37, Editions EYROLLES, 1985.

[HilLev87] **H.P HILLION, A.H LEVIS**: *Timed Event-Graph and Performance evaluation of systems.* 8th Europ. Workshop on Appl. and Theory of Petri Nets, Zaragoza, SPAIN, June 1987.

[LevSto87] **N G.LEVESON, J L STOLZY**:*Safety Analysis Using Petri Nets.* IEEE Trans. on Software Engineering, Vol SE-13, no 3, March 87, pp386-397.

[MeBe83] **M.MENASCHE, B.BERTHOMIEU**: *An Enumerative Approach for Analyzing Time Petri Nets.* IFIP 83, PARIS, Sept 83.

[Merlin75] **MERLIN:***On the Relation between Timings and Structure of Recoverable Processes*. IBM Research Rep RC5551 (24094) 7/31/75.

[NiPlWi81] **NIELSEN, PLOTKIN, WINSKEL**: *Petri Nets, Event Structures and Domains*. TCS 134 (1981), 85-108.

[OstWon87] **J S.OSTROFF, W M.WONHAM**:*Modelling, Specifying, and Verifying Real-time Embedded Computer Systems.* Proceedings of the Real-time Symposium, SAN JOSE (California), Dec 1-3, 1987, pp124-132.

[Petri77] **C.A PETRI**: *Non-sequential processes.* Internal Report GMD-ISF-77-5 (1977), Gesellschaft für Mathematik und Datenverarbeitung, BONN, (RFA).

[Ramch74] **C RAMCHANDANI**: *Analysis of asynchronous concurrent systems by Timed Petri Nets.* Technical Report n°120, Lab. for Computer Science, M.I.T, Cambridge, MA, 1974.

[Reisig86] **W.REISIG**: *Embedded system description using Petri nets.* in Embedded System, LNCS n° 284, March 1986, pp18-62..

[Reisig87] **W.REISIG:***A strong part of concurrency.* Advances in Petri Nets 1987, Lecture Note in Computer Science 266, Springer-Verlag, 238-272.

[Rozen86] **G.ROZENBERG**: *Behaviour of Elementary Net Systems*. in [ACPN86] Part I, 60-94.

[Sifak80] **J SIFAKIS**: *Performance evaluation of systems using nets.* Net Theory and Applications, Ed Brauer, LNCS n° 84, Springer-Verlag, 1980, pp307-319.

[Thiag86] **P.S.THIAGARAJAN**:*Elementary Net Systems*. in [ACPN86] Part I, 26-59.

[Winskel85] **G. WINSKEL**:*Petri Nets, Morphisms and Compositionality* , Advances in Petri Nets 1985, LNCS n°222, Springer-Verlag, 1985, pp453-477.

VIII. ANNEXES

VIII.1. ANNEX I: DIOID

(ββS,□,□,ℍ,Ξ) is a (commutative) **dioid** *iff:*

i) □ : ββS × ββS → ββS is a binary operation over ββS, such that (ββS,□,ℍ) is a commutative monoid (i.e: □ is commutative, associative and ℍ is its zero element) [1];

ii) □ : ββS × ββS → ββS is a binary operation over ββS, such that (ββS,□,Ξ) is a (commutative) monoid (i.e: □ is (commutative), associative and Ξ is its zero element);

iii) □ is left and right distributive over □;

[1] For rings, the opposite element is required, so that the algebraic structure for the first operation is a group.

iv) $\mathbb{H}$ is an absorbing null element for □□ (absorbent element).

In order to prove proposition P1, first define operation $\underline{\oplus}$ over $\wp_f(\beta S)\backslash\varnothing$:

$\forall F,G\in \wp_f(\beta S)\backslash\varnothing$: $F\underline{\oplus}G = \cup_{f\in F,\ g\in G}\{f\oplus g\}$, $\upsilon=\{\underline{0}\}$ is the identity element for $\underline{\oplus}$.

Then, establish that $(\wp_f(\beta S)\backslash\varnothing, \underline{\oplus}, \upsilon)$ is a commutative monoid.

Introduce the equivalence relation ˜; prove that ˜ is a congruence relation on $\wp_f(\beta S)\backslash\varnothing$, with respect to $\underline{\oplus}$ and $\cup$.

Finally, show that ($\beta\beta S$,□,$\mathbb{H}$) is a commutative monoid; ($\beta\beta S$,□,Ξ) is a (commutative) monoid; □ is left and right distributive over □ and $\mathbb{H}$ is an absorbing null element for □. $\square$

VIII.2. ANNEX II: GRAPHS

VIII.2.1. Paths in graph

Let $G = (V, \Gamma)$ be a graph.

A **path** μ from vertex a to vertex b is a finite sequence of nodes, written $\mu = (v_0,v_1,...,v_n)$ such that v_0=a, v_n=b, and $\forall$i: i:0..n-1, $(v_i,v_{i+1})\in\Gamma$. **(D24)**

The **length** of $\mu=(v_0,v_1,...,v_n)$ (written $|\mu|$) is n, i.e the number of arcs on μ.

A path $\mu = (v_0,v_1,...,v_n)$ such that $v_0 = v_n$ is called a **circuit**.

We consider only directed graphs and directed paths. Thus, in this paper, path stands for **directed path**.

Let $PATH_G(x,y)$ be the set of the paths from x to y, in G. $PATH_G =_{def} \cup_{x,y\in V} PATH_G(x,y)$.

We introduce a binary relation "**on**":

$\mathbf{on}_G \subseteq V \times PATH_G$ such that $x\ \mathbf{on}_G\ \mu=(v_0,v_1,...,v_n)$ *iff* $\exists i$: $v_i = x$ **(D25)**

VIII.2.2. Rooted graphs

$G = (V, \Gamma)$ is said to be a rooted graph *iff*

G is acyclic, i.e without (directed) circuit; **(D26)**

and $\exists!\perp \in V$: $\forall y \in V\backslash\{\perp\}$: $\perp \Gamma^+ y$, written $y \in \Gamma^{+,\perp}$.

$\perp$ is called the root of G.

Let G be a rooted graph, the root of which is $\perp$, **the height function** from V to N, written $\mathbf{h_G}$, is such that:

$\forall x \in V$: $\mathbf{h_G}(x) =_{def} \max \{ |\mu| \mid \mu \in PATH_G(\perp,x) \}$ **(D27)**

Of course, $\mathbf{h_G}(\perp)=0$.

VIII.2.3. (Set of bags-)Weighted graphs

A weighted graph is a couple (G, W) where G=(V, Γ) is a graph
and $W:\Gamma \rightarrow \beta\beta E$ is a weight function for some given set E. **(D28)**

For any $x,y \in V$ *if* $(x,y) \in \Gamma$ *then* W(x,y) is called the weight of the arc (x,y).

W is extended to paths as follows: $\forall x,y,z \in V$, W((x,...,y)•(y,z)) = W(x,...,y)□W(y,z). Note that ordinary weight functions have N or □ + as codomain, we take equivalence classes of sets of bags, instead.

VIII.2.4. Fundamental predecessors

Definition

$\forall x,y \in T'$: x **fund_pred**$_G$ y $\Leftrightarrow$ $\forall \mu \in PATH_G(\perp,y)$: x **on**$_G$ μ

Let $FUND_G(y) =_{def} \{ x \in T' \mid x$ **fund_pred**$_G$ $y \}$ **(D20')**

Construction of FUND$_G$

$FUND_G(\perp) := \{\perp\}$;

for i := 1 *to* h(T') *do*

for each $x \in T$ *such that* h(x)=i *do* **(A2)**

$FUND_G(x) := \{x\} \cup \{ \bigcap FUND_G(y) \mid y \in \Gamma^{-1}_x \}$

The proof is typically graph theoretical, it is given in an extended version of this paper [André90].

VIII.2.5. Dominant predecessor

Definition

$\forall y \in T'$, $z \in \Gamma^{-1,y}$ is a dominant predecessor of y in G
(written z **dom_pred**$_G$ y) *iff* **(D22')**

$\exists x \in T'$: $\forall z' \in \Gamma^{-1,y}$ z'?z $\Rightarrow$ L.delay(x,z')□Wi(z',y) < S.delay(x,z)□Wi(z,y).

A dominant predecessor can play the same role as a fundamental predecessor:

Proposition

if z **dom_pred**$_G$ y *then* delay(z,y) = (Wi(z,y),Wl(z,y)). **(P3')**

Proof:

Note that $\forall v,v' \in T'$: Wi(v,v') = Wl(v,v') and LI(v,v') = LL(v,v') $\Rightarrow$ S.delay(v,v') = L.delay(v,v').

Let $x \in T'$ such that delay(x,z') is given for any $z' \in \Gamma^{-1}_y$.

By D18, delay(x,y) = Max $_{z'\in\Gamma^{-1},y}$ { delay(x,z') □ (Wi(z',y),Wl(z',y)) }, which can be decomposed into S.delay(x,y) = Max $_{z'\in\Gamma^{-1},y}$ { S.delay(x,z') □□Wi(z',y)} and L.delay(x,y) = Max $_{z'\in\Gamma^{-1},y}$ { L.delay(x,z') □□Wi(z',y) □□L} where L = L(p(y)).

• S.delay(x,y) = Max $_{z'\in\Gamma^{-1},y}$ { S.delay(x,z') □□Wi(z',y)} = Max $_{z'\in\Gamma^{-1},y}$ { L.delay(x,z') □□Wi(z',y)} < S.delay(x,z) □□Wi(z,y) when z **dom_pred**$_G$ y. Now, S.delay(x,y) is maximal ⇒ S.delay(x,y) = S.delay(x,z) □□Wi(z,y) ⇒$_{D16}$ S.delay (z,y) = Wi(z,y).

• L.delay(x,y) = Max $_{z'\in\Gamma^{-1},y}$ { L.delay(x,z') □□Wi(z',y) □□L} = Max $_{z'\in\Gamma^{-1},y}$ {L.delay(x,z') □□Wi(z',y)} □□L < (S.delay(x,z) □□Wi(z,y)) □□L = S.delay(x,y) □□L = L.delay(x,z) □□Wi(z,y) □□L = L.delay(x,z) □□Wl(z,y) = L.delay(x,y) ⇒ L.delay(x,y) = L.delay(x,z) □□Wl(z,y) = S.delay(x,y) □□L ⇒$_{D16}$ L.delay(z,y) = Wl(z,y). $\square$