Basic Research in Computer Science

BRICS RS-96-32 **P. S. Thiagarajan**: **Regular Trace Event Structures**

Regular Trace Event Structures

P. S. Thiagarajan

BRICS Report Series **RS-96-32**

ISSN 0909-0878 **September 1996**

See back inner page for a list of recent publications in the BRICS Report Series. Copies may be obtained by contacting:

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK - 8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

BRICS publications are in general accessible through World Wide Web and anonymous FTP:

`http://www.brics.dk/`
`ftp://ftp.brics.dk/pub/BRICS`

Regular Trace Event Structures

P.S. Thiagarajan*
BRICS†
Department of Computer Science
University of Aarhus
Ny Munkegade
DK-8000 Aarhus C, Denmark

September, 1996

Abstract

We propose trace event structures as a starting point for constructing *effective* branching time temporal logics in a non-interleaved setting. As a first step towards achieving this goal, we define the notion of a regular trace event structure. We then provide some simple characterizations of this notion of regularity both in terms of recognizable trace languages and in terms of finite 1-safe Petri nets.

0 Introduction

This paper may be viewed as a first step towards the construction of *effective* branching time temporal logics in a non-interleaved setting. We believe the

*On leave from School of Mathematics, SPIC Science Foundation, Madras, India
†**B**asic **R**esearch **I**n **C**omputer **S**cience,
Centre of the Danish National Research Foundation.

study of such logics will yield the formal basis for extending – to a branching time framework – the partial order based verification techniques that have been established in the linear time world [GW, Pel, Val].

For achieving the stated goal one must identify the structures over which the logics are to be interpreted. We propose here objects called trace event structures as suitable candidates. We also initiate their systematic study by pinning down the notion of regularity for these structures.

Trace event structures constitute a common generalization of trees and (Mazurkiewicz) traces. In a linear time setting, moving from sequences to traces has turned out to be a very fruitful way of going from total orders to partial orders. Trees, which may be viewed as objects obtained by gluing together sequences, constitute the basic structures in the branching time world. Hence it seems worthwhile to glue together traces and consider the resulting structures, called trace event structures as a basic class of structures for settings in which the underlying temporal frames have the flavour of both branching time *and* non-interleaved behaviours.

A good deal of the solutions to the decidability and model checking problems for branching time logics hinges on the notion of a regular labelled tree. For instance, SnS, the monadic second order theory of n-branching trees, is decidable because the decision problem for this logic can be reduced (as shown in the famous paper by Rabin [Rab]) to the emptiness problem for tree automata running over labelled infinite trees. The emptiness problem for these tree automata is decidable because the language of labelled infinite trees accepted by a tree automaton is non-empty only if it accepts a regular labelled tree.

Thus, to test the effectiveness and adequacy of automata and logics to be interpreted over trace event structures, one must understand what are regular trace event structures. Here we provide an obvious definition and some simple characterizations of this notion of regularity.

We start with a presentation of trace structures. We do so because they are known in the literature [NW, PK] and the means for going back and forth between trace structures and trace event structures is also well-understood. Indeed, in [PK] a number of branching time temporal logics over trace structures are considered. However these logics turn out to be undecidable. In

our view, the key to obtaining useful and yet decidable branching time logics over trace structures is to suitably limit the quality of the objects over which quantification is to be allowed. We feel that the study of trace event structures will help in identifying the required restrictions.

In section 2 we define regular trace structures and provide an "event-based" characterization of regularity. Trace event structures are introduced in section 3. The notion of regularity and its characterization is transported from trace structures to trace event structures in section 4. Labelled trace event structures are introduced in section 5 and regular labelled trace event structures are characterized in this section.

The result concerning labelled trace event structures turns out to be – in an event-based language – a conservative extension of the standard result concerning regular labelled trees (see for instance [Tho]). In section 6 we show that regular trace event structures and their labelled versions can be identified with unfoldings of *finite* 1-safe Petri nets. In the concluding section we discuss future work.

1 Trace Structures

A (Mazurkiewicz) trace alphabet is a pair (DR, I) where DR is a finite non-empty alphabet set and $I \subseteq DR \times DR$ is an irreflexive and symmetric relation called the independence relation. We will often refer to DR as the set of directions.

Example 1.1 *As a running example we shall use the trace alphabet* (DR_0, I_0) *where* $DR_0 = \{l, m, r\}$ *and* $I_0 = \{(l, r), (r, l)\}$. □

As usual, DR^* is the set of finite words generated by DR and ϵ is the null word. The independence relation I induces the natural equivalence relation $\sim_I$. It is the least equivalence relation contained in $DR^* \times DR^*$ which satisfies:

- If $\sigma, \sigma' \in DR^*$ and $(a, b) \in I$ then $\sigma ab\sigma' \ \sim_I \ \sigma ba\sigma'$.

The $\sim_I$-equivalence classes are called (finite Mazurkiewicz) traces. $[\sigma]_{\sim_I}$ will denote the $\sim_I$-equivalence class containing σ. We let $TR(DR, I)$ be the set of traces over (DR, I). In other words, $TR(DR, I) = DR^*/\sim_I$. Where (DR, I) is clear from the context, we will write $[\sigma]$ instead of $[\sigma]_{\sim_I}$ and we will write TR instead of $TR(DR, I)$.

Example 1.1 (cont.) *Let TR_0 be the set of traces over (DR_0, I_0). Then $\{lrm, rlm\}$ is a member of TR_0. Note also that $[lmr] = \{lmr\}$.* □

Traces can be ordered in an obvious way. This ordering relation $\sqsubseteq_{(DR,I)} \subseteq TR \times TR$ is given by

- $[\sigma] \sqsubseteq_{(DR,I)} [\sigma']$ iff there exists $\sigma'' \in DR^*$ such that $\sigma\sigma'' \in [\sigma']$.

It is easy to observe that $\sqsubseteq_{(DR,I)}$ is a partial order. From now on, we shall almost always write $\sqsubseteq$ instead of $\sqsubseteq_{(DR,I)}$ whenever (DR, I) is clear from the context. Abusing notation, we shall also use $\sqsubseteq$ to denote the restriction of $\sqsubseteq$ to a given subset of TR.

Example 1.1 (cont.) *In TR_0, we have $[r] \sqsubseteq [llrm]$. We also have $[lmr] \not\sqsubseteq [rml]$ and $[rml] \not\sqsubseteq [lmr]$.* □

We can now define one of the primary objects of interest in this paper.

Definition 1.2 *Let (DR, I) be a trace alphabet. A trace structure over (DR, I) is a subset $B \subseteq TR(DR, I)$ of traces which satisfies the following conditions.*

(TS1) If $[\sigma] \in B$ and $[\sigma'] \sqsubseteq [\sigma]$ then $[\sigma'] \in B$.

(TS2) If $[\sigma a], [\sigma b] \in B$ with $\sigma \in DR^*$ and $(a, b) \in I$ then $[\sigma ab] \in B$.

□

Trace structures have a well-understood relationship with prime event structures ([RT, NW]). This relationship, which finds a clean and general presentation in [NW], will play a central role in the present work. Trace structures have been called trace systems in a logical setting [PK].

We shall adopt the standpoint that trace structures represent distributed behaviours in a branching time framework just as traces represent distributed behaviours in a linear time framework (see for instance [Thi]). Let $B \subseteq TR(DR, I)$ be a trace structure. Then B is supposed to stand for the poset $(B, \sqsubseteq)$. The crucial new feature – in contrast to the classical setting – is that some elements of B might have a common future due to the causal independence of directions as permitted by I. Indeed, the classical setting is restored whenever $I = \emptyset$.

Example 1.1 (cont.) $\{[\epsilon], [l], [r], [lm], [lr], [lrm]\}$ *is a trace structure over* (DR_0, I_0). *The Hasse diagram of the behaviour captured by this structure is shown in fig. 1.1.* □

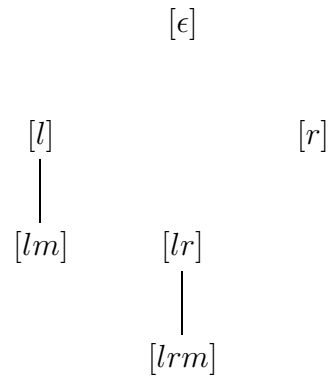

Figure 1.1

As this example suggests, we have a very generous notion of a branching time behaviour at this stage. In the classical setting (i.e. when $I = \emptyset$), one would demand that the tree represented by a trace structure should have "proper" frontiers; for each node either all its successors must be present or none must be present. This demand is usually made for obtaining clean automata theoretic constructions. At present we do not have a good notion

of automata running over trace (event) structures. Hence we shall ignore the issue of proper frontiers and work with the generous class of behaviours admitted by def. 1.2.

It will be convenient to establish the link between trace languages and I-consistent word languages. A trace language is just a subset of TR. The word language $L \subseteq DR^*$ is said to be I-consistent in case $[\sigma] \subseteq L$ for every $\sigma \in L$. In other words, either all members of a trace are in L or none of them are in L. It is easy to see that subsets of TR and I-consistent subsets of DR^* represent each other. Through the remaining sections, we shall often refer to this connection via the map $ts : 2^{TR} \to 2^{DR^*}$ given by

$$ts(\hat{L}) = \bigcup \{[\sigma] \mid [\sigma] \in \hat{L}\}.$$

Clearly, for every $\hat{L} \subseteq TR$, $ts(\hat{L})$ is an I-consistent subset of DR^*. We shall often apply ts to a trace structure. After all, a trace structure can be viewed as a trace language which satisfies the two closure properties (TS1) and (TS2).

2 Regular Trace Structures

Through the rest of the paper we fix a trace alphabet (DR, I) and often refer to it implicitly. We let $a.b.d$ range over DR and let σ, σ', and σ'' with or without subscripts range over DR^*. D is the dependence relation given by $D = (DR \times DR) - I$. The notations and terminology developed so far w.r.t. (DR, I) will be assumed throughout. For convenience, we will often write σ instead of $[\sigma]$ in talking about traces. From the context it should be clear whether we are referring to the word σ or the trace $[\sigma]$.

Definition 2.1

(i) Let $B \subseteq TR$ be a trace structure and $\sigma \in B$. Then $B_\sigma = \{\sigma' \mid \sigma\sigma' \in B\}$.

(ii) The equivalence relation $R_B \subseteq B \times B$ is given by:

$$\sigma \; R_B \; \sigma' \text{ iff } B_\sigma = B_{\sigma'}.$$

(iii) The trace structure B is regular *iff R_B is of finite index.* □

Our main goal is to characterize the regularity of objects called labelled trace event structures to be introduced in section 5. They will be labelled versions of the event structure representations of trace structures. With this as motivation, the rest of this section will be devoted to establishing an event-based characterization of regular trace structures. We note that the regularity of a trace structure just guarantees that it has an ultimately periodic shape. However, for the labelled objects dealt with later, our definition will amount to a conservative extension of the notion of a regular labelled tree.

It should be clear that the trace structure $(B, \sqsubseteq)$ is regular iff B is a recognizable subset of TR. It will be convenient to first bring this out in a more formal fashion.

We say that $\hat{L} \subseteq TR$ is recognizable iff $ts(\hat{L})$ is a recognizable (equivalently, regular) subset of DR^*. For $L \subseteq DR^*$ we denote by $\equiv_L$ the right congruence contained in $DR^* \times DR^*$ which is induced by L via

$$\sigma \equiv_L \sigma' \text{ iff } \forall \sigma''.[\sigma\sigma'' \in L \text{ iff } \sigma'\sigma'' \in L].$$

From the well-known fact that L is a recognizable subset of DR^* iff $\equiv_L$ is of finite index, the next observation is immediate.

Proposition 2.2 *The following statements are equivalent:*

(i) $(B, \sqsubseteq)$ is a regular trace structure.

(ii) $B \subseteq TR$ is recognizable. □

For the event-based characterization we are after, it is necessary to define so-called prime elements of TR. Suppose $\sigma \neq \epsilon$. Then $last(\sigma)$ is the letter that appears last in σ.

We say that σ is *prime* iff $\sigma \neq \epsilon$ and there exists d such that $last(\sigma') = d$ for every $\sigma' \in [\sigma]$.

Example 2.3 *In TR_0, $[llrm]$ is prime but $[lmlr]$ is not.* □

For each σ, we define $pr(\sigma) = \{\sigma' \mid \sigma' \text{ is prime and } \sigma' \sqsubseteq \sigma\}$. Of course, $pr(\sigma) = \emptyset$ only if $\sigma = \epsilon$. Finally, for $\hat{L} \subseteq TR$ we set $pr(\hat{L}) = \bigcup_{\sigma \in \hat{L}} pr(\sigma)$.

It turns out prime traces constitute the building blocks of the poset of traces $(TR, \sqsubseteq)$. To bring this out, let the compatibility relation $\uparrow \subseteq TR \times TR$ be defined as: $\sigma \uparrow \sigma'$ iff there exists σ'' such that $\sigma \sqsubseteq \sigma''$ and $\sigma' \sqsubseteq \sigma''$. Further, if $X \subseteq TR$ then $\sqcup X$ will denote the l.u.b. of X (under $\sqsubseteq$) in TR if it exists. The next set of results have been assembled from [NW].

Proposition 2.4

(i) Suppose $X \subseteq TR$ such that $\sigma \uparrow \sigma'$ for every $\sigma, \sigma' \in X$. Then $\sqcup X$ exists.

(ii) $\sigma = \sqcup pr(\sigma)$ for every σ.

(iii) Let B be a trace structure and $X \subseteq B$ such that $\sqcup X$ exists in TR. Then $\sqcup X \in B$.

(iv) Let B be a trace structure and $\sigma \in TR$. Then $\sigma \in B$ iff $pr(\sigma) \subseteq B$.

The rest of the section will be devoted to establishing the following characterization of regular trace structures.

Theorem 2.5 *Let B be a trace structure. Then the following statements are equivalent.*

(i) B is regular.

(ii) $pr(B)$ is recognizable. □

We shall show that B is recognizable iff $pr(B)$ is recognizable. Theorem 2.5 will then follow at once from proposition 2.2.

Lemma 2.6 *Suppose the trace structure B is recognizable. Then $pr(B)$ is also recognizable.*

Proof: Let $L = ts(B)$. Then L is recognizable and hence there exists a deterministic finite state automaton $\mathcal{A}$ operating over DR such that $\mathcal{L}(\mathcal{A})$, the language recognized by $\mathcal{A}$, is L. Now consider the deterministic automaton $\mathcal{A}^{top} = (Q, \rightarrow, q_{in}, F)$ also operating over DR defined by

- $Q = 2^{DR}$
- $\rightarrow \subseteq Q \times DR \times Q$ is given by: $X \xrightarrow{d} Y$ iff $Y = (X - D(d)) \cup \{d\}$ where $D(d) = \{d' \mid d' \; D \; d\}$.
- $q_{in} = \emptyset$.
- $F = \{\{d\} \mid d \in DR\}$.

It is easy to see that $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}^{top}) = L_{pr}$ where $L_{pr} = ts(pr(B))$. $\square$

For showing the converse of lemma 2.6 we shall make use of Zielonka's theorem [Zie] and the gossip automaton [MS]. For presenting Zielonka's theorem we need to introduce asynchronous automata operating over distributed alphabets. A distributed alphabet is a family $\{\Sigma_p\}_{p \in \mathcal{P}}$ where $\mathcal{P}$ is a finite set of processes (sequential agents) and each Σ_p is a finite set of actions; the set of actions the agent p participates in. We associate a distribution function $loc_{\tilde{\Sigma}} : \Sigma \rightarrow 2^{\mathcal{P}}$ with $\tilde{\Sigma}$ where $\Sigma = \bigcup_{p \in \mathcal{P}} \Sigma_p$ is the global alphabet and $loc_{\tilde{\Sigma}}(x) = \{p \mid x \in \Sigma_p\}$ for each x in Σ. This in turn induces canonically the trace alphabet $(\Sigma, I_{\tilde{\Sigma}})$ where $I_{\tilde{\Sigma}} \subseteq \Sigma \times \Sigma$ is obtained via: $x \; I_{\tilde{\Sigma}} \; y$ iff $loc_{\tilde{\Sigma}}(x) \cap loc_{\tilde{\Sigma}}(y) = \emptyset$.

On the other hand, a trace alphabet can be implemented as a distributed alphabet in many different ways. Here we shall exclusively work with maximal D-cliques. For our specific trace alphabet (DR, I) we call $p \subseteq DR$ a maximal D-clique in case p is a maximal subset of DR with the property $p \times p \subseteq D$. We let $\mathcal{P} = \{p_1, p_2, \ldots, p_K\}$ be the set of maximal D-cliques of (DR, I). We let p, q range over $\mathcal{P}$ and P, Q range over non-empty subsets of $\mathcal{P}$. For $Q = \{p_{i_1}, p_{i_2}, \ldots, p_{i_l}\}$ with $i_1 < i_2 \ldots < i_l$ we will often instead write $Q = \{i_1, i_2, \ldots, i_l\}$. This will be especially convenient when dealing with the gossip automaton.

$\mathcal{P}$, viewed as the names of a set of processes gives rise to the distributed alphabet $\widetilde{DR} = \{DR_p\}_{p\in\mathcal{P}}$ where $DR_p = p$ for each p. This distributed alphabet implements (DR, I) in the sense that the canonical trace alphabet induced by $\widetilde{DR}$ is exactly (DR, I).

Example 2.3 (cont.) *The maximal D-cliques of* (DR_0, I_0) *are* $\{l, m\}$ *and* $\{m, r\}$. *Hence the distributed alphabet obtained via maximal D-cliques is* $\widetilde{DR}_0 = \{\{l, m\}, \{m, r\}\}$. □

In what follows, we shall often have to deal with $\mathcal{P}$-indexed families of the form $\{X_p\}_{p\in\mathcal{P}}$ and DR-indexed families of the form $\{Y_d\}_{d\in DR}$. In both cases, we shall often write $\{X_p\}$ and $\{Y_d\}$ respectively.

An asynchronous automaton over $\widetilde{DR} = \{DR_p\}$ is a structure $\mathcal{A} = (\{S_p\}, \{\rightarrow_d\ \}, S_{in}, F)$ where the various parts of $\mathcal{A}$ are defined as follows. In doing so, we shall also develop some terminology and notations.

- Each S_p is a finite non-empty set of states called p-states. They are the local states of the agent p.

  $S = \bigcup_{p\in\mathcal{P}} S_p$ is the set of local states. A Q-state is a map $s : Q \rightarrow S$ such that $s(q) \in S_q$ for each q in Q. We let S_Q denote the set of Q-states and call $S_{\mathcal{P}}$, the set of global states. A d-state is just a Q-state where $Q = loc_{\widetilde{DR}}(d)$. Recall that $loc_{\widetilde{DR}}(d) = \{p \mid d \in p\}$.

  We let S_d denote the set of d-states. If $P \subseteq Q$ and s is a Q-state then $(s)_P$ is the restriction of s to P.

- $\rightarrow_d \subseteq\ S_d \times S_d$ for each d.

- $S_{in} \subseteq S_{\mathcal{P}}$ is the set of global initial states.

- $F \subseteq S_{\mathcal{P}}$ is the set of global finite states.

From now on we shall only consider asynchronous automata operating over the fixed distributed alphabet $\widetilde{DR} = \{DR_p\}$. Hence we will almost always suppress mention of $\widetilde{DR}$ and write loc instead of $loc_{\widetilde{DR}}$.

Let $\mathcal{A} = (\{S_p\}, \{\rightarrow_d\}, S_{in}, F)$ be an asynchronous automaton. Then $\{\rightarrow_d\}$ induces the global transition relation $\rightarrow_{\mathcal{A}} \subseteq S_{\mathcal{P}} \times DR \times S_{\mathcal{P}}$ given by:

Let $s, s' \in S_{\mathcal{P}}$ and $d \in DR$. Then $s \xrightarrow{d}_{\mathcal{A}} s'$ iff the following conditions are satisfied:

(i) $((s)_d, (s')_d) \in \rightarrow_d$.

(ii) $\forall p \notin loc(d).\ s(p) = s'(p)$.

Let $prf(\sigma)$ be the set of prefixes of σ. Then a run of $\mathcal{A}$ over σ is a map $\rho : prf(\sigma) \rightarrow S_{\mathcal{P}}$ such that $\rho(\epsilon) \in S_{in}$ and for every $\sigma' d \in prf(\sigma)$, $\rho(\sigma') \xrightarrow{d}_{\mathcal{A}} \rho(\sigma' d)$. The run ρ is accepting iff $\rho(\sigma) \in F$. The language recognized by $\mathcal{A}$ is denoted as $\mathcal{L}(\mathcal{A})$ and is defined to be the least subset DR^* satisfying:

$\sigma \in \mathcal{L}(\mathcal{A})$ iff there exists an accepting run of $\mathcal{A}$ over σ.

We say that $\mathcal{A}$ is deterministic in case $\rightarrow_{\mathcal{A}}$ is a deterministic transition relation. In other words, $s \xrightarrow{d}_{\mathcal{A}} s'$ and $s \xrightarrow{d}_{\mathcal{A}} s''$ imply $s' = s''$. Moreover, $|S_{in}| = 1$. We shall say that $\mathcal{A}$ is complete in case $\mathcal{A}$ has a run over every σ in DR^*. Zielonka's theorem can be phrased as follows.

Theorem 2.7 *Let $\hat{L} \subseteq TR$ and $ts(\hat{L}) = L$. Then $\hat{L}$ is recognizable iff there exists a deterministic complete asynchronous automaton $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = L$.*

□

For presenting the gossip automaton we need the notion of a local view of a trace. The p-view of σ is denoted as $\downarrow^p(\sigma)$ and is defined as: $\downarrow^p(\sigma) = \sqcup\{\sigma' \mid \sigma' \in pr(\sigma) \text{ and } last(\sigma') \in p\}$. Noting that $\sqcup\emptyset = \{\epsilon\}$, it follows easily that $\downarrow^p(\sigma)$ is well-defined for every σ.

The next set of observations follows easily from the definitions and [NW].

Proposition 2.8

(i) For every σ, $\sigma = \sqcup_{p \in \mathcal{P}} \downarrow^p(\sigma)$.

(ii) Suppose B is a trace structure and $\sigma \in TR$. Then $\sigma \in B$ iff $\downarrow^p(\sigma) \in B$ for every p. □

We can now define a function which will pick out the agent in Q which has the latest information – among the agents in Q – at a trace about some agent (which might or might not be in Q).

Accordingly, $latest_Q : TR \times \mathcal{P} \to Q$ is defined as:

$latest_Q(\sigma, p) = \hat{q}$ provided $\hat{q}$ is the agent in Q with least index which has the property that $\downarrow^j (\downarrow^q (\sigma)) \subseteq \downarrow^j (\downarrow^{\hat{q}} (\sigma))$ for every $q \in Q$. Recall that $\mathcal{P} = \{p_1, p_2, \ldots, p_K\}$. In dealing with the gossip automaton, we will often write i instead of p_i (with $i \in \{1, 2, \ldots, K\}$). The gossip automaton computes the $latest_Q$ using only a bounded amount of information. For our purposes, the key result proved in [MS] can be phrased as follows:

Theorem 2.9 *There exists an effectively constructible deterministic complete asynchronous automaton*

$$\mathcal{A}_\Gamma = (\{\Gamma_p\}, \{\Rightarrow_d\}, \Gamma_{in}, \Gamma_{\mathcal{P}})$$

such that for each $Q = \{i_1, i_2, \ldots, i_n\} \subseteq \mathcal{P}$ there exists an effectively computable function $gossip_Q = \Gamma_{i_1} \times \Gamma_{i_2} \ldots \times \Gamma_{i_n} \times \mathcal{P} \to Q$ such that, for every σ and every p,

$$latest_Q(\sigma, p) = gossip_Q\ (\nu(i_1), \nu(i_2), \ldots, \nu(i_n), p)$$

where $\rho_\Gamma(\sigma) = \nu$ and ρ_Γ is the unique run of ρ_Γ over σ. □

Thus, by examining the Q-states of $\mathcal{A}_\Gamma$ at $\rho_\Gamma(\sigma)$, we can, with a bit of work, determine which agent among Q has the latest information about p at σ.

Using the gossip automaton we can associate with each asynchronous automaton, a second asynchronous automaton $\mathcal{A}^{pr}$ with the following property.

Fix σ and suppose that $\mathcal{A}$ reaches the global state $s^{(p)}$ after running over $\downarrow^p(\sigma)$ (i.e. after running over some member of $ts(\downarrow^p(\sigma))$. Further suppose that $\mathcal{A}^{pr}$ reaches the global state $\hat{s}$ after running over σ and $\mathcal{A}_\Gamma$ (the gossip automaton) reaches the global state ν after running over σ. Then for each p, it will be the case that $\hat{s}(p) = (s^{(p)}, \nu(p))$.

Using this association between $\mathcal{A}$ and $\mathcal{A}^{pr}$ we can easily obtain the result we are after. Let $\mathcal{A} = (\{S_p\}, \{\rightarrow_d\}, S_{in}, F)$. Recall that $\mathcal{A}_\Gamma = (\{\Gamma_p\}, \{\Rightarrow_d\}, \Gamma_{in}, \Gamma_{\mathcal{P}})$. We now define the asynchronous automaton $\mathcal{A}^{pr} = (\{\hat{S}_p\}, \{R_d\}, \hat{S}_{in}, \hat{F})$ as follows:

- For each q, $\hat{S}_q = S_{\mathcal{P}} \times \Gamma_q$.
- Let $\hat{s}, \hat{t} \in \hat{S}_d$ with $\hat{s}(p) = (s^{(p)}, \nu_p)$ and $\hat{t}(p) = (t^{(p)}, \delta_p)$ for each p in $loc(d)$. Then $(\hat{s}, \hat{t}) \in R_d$ iff the following conditions are satisfied:
  - (i) $((s^{(p)})_d, (t^{(p)})_d) \in \rightarrow_d$. Recall that $(s)_d$ is s restricted to $loc(d)$ in case s is a Q-state with $loc(d) \subseteq Q$.
  - (ii) $(\nu_d, \delta_d) \in \Rightarrow_d$ where ν_d and δ_d are the two d-states of $\mathcal{A}_\Gamma$ satisfying $\nu_d(p) = \nu_p$ and $\delta_d(p) = \delta_p$ for every $p \in loc(d)$.
  - (iii) Suppose $loc(d) = Q = \{i_1, i_2, \ldots, i_n\}$, $q \in Q$ and $p \notin Q$. Then $t^{(q)}(p) = s^{(\hat{q})}(p)$ where $\hat{q} = gossip_Q(\nu_{i_1}, \nu_{i_2}, \ldots, \nu_{i_n}, p)$. Recall that $\hat{s}(x) = (s^{(x)}, \nu_x)$ foe every $x \in loc(d)$.
  - (iv) For every $p, q \in loc(d)$, $t^{(p)} = t^{(q)}$.
- Let $S_{in} = \{s_{in}\}$ and $\Gamma_{in} = \{\nu_{in}\}$. Then $\hat{S}_{in} = \{\hat{s}_{in}\}$ where for each p, $\hat{s}_{in}(p) = (s_{in}, \nu_{in}(p))$.
- Let $\hat{s} \in \hat{S}_{\mathcal{P}}$ with $\hat{s}(p) = (s^{(p)}, \nu_p)$ for each p. Then $\hat{s} \in \hat{F}$ iff $s^{(p)} \in F$ for every p.

Lemma 2.10 *Let B be a trace structure. Suppose $pr(B)$ is recognizable. Then B is also recognizable.*

Proof: Let $ts(pr(B)) = L$. Then by theorem 2.7, there exists a deterministic complete asynchronous automaton $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = L$. Now consider the

automaton $\mathcal{A}^{pr}$ associated with $\mathcal{A}$ and constructed as specified above. Then we claim that $\mathcal{L}(\mathcal{A}^{pr}) = L'$ where $L' = ts(B)$.

To see this, for each σ let ρ_σ ($\hat{\rho}_\sigma$) be the unique run of $\mathcal{A}$ ($\mathcal{A}^{pr}$) over σ. Then induction on the length of σ accompanied by an examination of the definitions will yield the following:

Fact: Let $\hat{\rho}_\sigma(\rho) = \hat{s}$ with $\hat{s}(p) = (s^{(p)}, \nu_p)$ for each p. Let $\sigma_q \in ts(\downarrow^q(\sigma))$ and $\rho_{\sigma_q}(\sigma_q) = s$. Then $s^{(q)} = s$.

Hence from the definition of $\mathcal{A}^{pr}$ it follows that $\sigma \in \mathcal{L}(\mathcal{A}^{pr})$ iff $\downarrow^p(\sigma) \subseteq \mathcal{L}(\mathcal{A}) = L$ for every p. But then according to proposition 2.8, $\sigma \in B$ iff $\downarrow^p(\sigma) \in B$ for every p. Consequently, $\sigma \in L'$ iff $\downarrow^p(\sigma) \subseteq L$ for every p. Thus $\sigma \in L'$ iff $\sigma \in \mathcal{L}(\mathcal{A}^{pr})$ as required. □

Theorem 2.5 now follows at once from lemmas 2.6, 2.10 and prop. 2.2. Lemma 2.10 admits a direct proof as shown in [DM]. However, for the net theoretic characterization of regularity that we obtain later, it is necessary to have our construction underlying the proof of lemma 2.10.

3 Trace Event Structures

We now wish to view trace structures as prime event structures. In this representation the causality, conflict and concurrency relation that glue together a trace structure will become explicit. The main motivation for considering this representation is that we expect the automata theoretic treatment of trace structures to be carried out in terms of their prime event structure representations.

We start with a notation concerning posets. Let $(X, \leq)$ be a poset and $Y \subseteq X$. Then $\downarrow Y = \{x \mid \exists y \in Y, x \leq y\}$ and $\uparrow Y = \{x \mid \exists y \in Y,\ y \leq x\}$. Whenever Y is a singleton with $Y = \{y\}$ we will write $\downarrow y$ ($\uparrow y$) instead of $\downarrow \{Y\}$ ($\uparrow \{Y\}$).

A prime event structure is a triple $ES = (E, \leq, \#)$ where $(E, \leq)$ is a poset and $\# \subseteq E \times E$ is an irreflexive and symmetric relation such that the following conditions are met:

- $\downarrow e$ is a finite set for every $e \in E$.
- For every $e_1, e_2, e_3 \in E$, if $e_1 \# e_2$ and $e_2 \leq e_3$ then $e_1 \# e_3$.

E is the set of events and $\leq$ is the causality relation. $\#$ is the conflict relation.

As usual, the states of a prime event structure will be called configurations. We say that $c \subseteq E$ is a configuration iff $c = \downarrow c$ and $(c \times c) \cap \# = \emptyset$. It is easy to see that $\emptyset$ is always a configuration and more interestingly, $\downarrow e$ is a configuration for every event e. We let C^{∞}_{ES} be the set of (finite and infinite) configurations and C_{ES} denote the set of finite configurations of ES.

It will be useful to introduce two derived relations associated with a prime event structure. Let $ES = (E, \leq, \#)$ be a prime event structure. Then $\lessdot \subseteq E \times E$ is defined as: $e \lessdot e'$ iff $e < e'$ (i.e. $e \leq e'$ and $e \neq e'$) and for every e'', if $e \leq e'' \leq e'$ then $e = e''$ or $e'' = e'$. In other words, $\lessdot = < - <^2$.

Next we define the minimal conflict relation $\#_\mu \subseteq E \times E$ via:

$$e \#_\mu e' \text{ iff } (\downarrow e \times \downarrow e') \cap \# = \{(e, e')\}.$$

A DR-labelled prime event structure is a quadruple $ES = (E, \leq, \#, \lambda)$ where $(E, \leq, \#)$ is a prime event structure and $\lambda : E \to DR$ is a labelling function. We can now present the proposed event structure representation of trace structures.

Definition 3.1 *A trace event structure over (DR, I) is a DR-labelled prime event structure $ES = (E, \leq, \#, \lambda)$ which satisfies the following requirements (with e, e' ranging over E):*

(TES1) $e \#_\mu e'$ *implies* $\lambda(e) \neq \lambda(e')$

(TES2) *If* $e \lessdot e'$ *or* $e \#_\mu e'$ *then* $(\lambda(e), \lambda(e')) \in D$

(TES3) *If* $(\lambda(e), \lambda(e')) \in D$ *then* $e \leq e'$ *or* $e' \leq e$ *or* $e \# e'$.

□

Thus a trace event structure is a DR-labelled prime event structure in which the DR-orientation of the events (as specified by the labelling function) respects the independence relation I. This is captured by the conditions (TES2) and (TES3). The first condition (TES1) merely reflects the fact that if $[\sigma] = [\sigma']$ then $[\sigma d] = [\sigma' d]$. These remarks might be easier to appreciate once we explain how trace structures and trace event structures represent each other. But first we shall consider some examples.

In diagramatic descriptions of labelled prime event structures the poset of events ordered by the causality relation will be shown by its Hasse diagram. The elements of the minimal conflict relation will be shown as squiggly edges. The conflict relation is then the relation uniquely induced by the causality and the minimal conflict relations. The events will be drawn as boxes.

Example 3.1 *Recall (DR_0, I_0) with $DR_0 = \{l, m, r\}$ and $I_0 = \{(l, r), (r, l)\}$. In fig. 3.1 (a) and 3.1 (b) and 3.1 (c) we show three examples of DR_0-labelled prime event structures. None of them constitutes a trace event structure over (DR_0, I_0).*

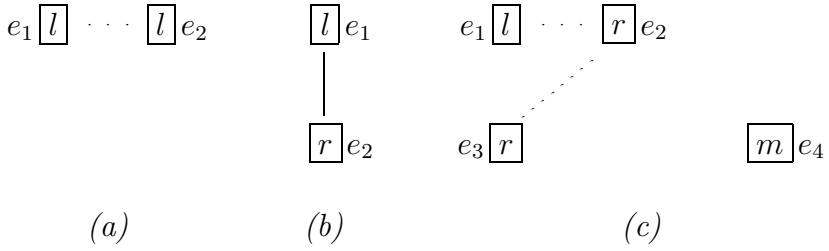

Figure 3.1

In fig. 3.1 (a) we have $e_1 \#_\mu e_2$ with $\lambda(e_1) = \lambda(e_2)$. In fig. 3.1 (b) we have $e_1 \lessdot e_2$ with $(\lambda(e_1) = \lambda(e_2)) \notin D$. In fig. 3.1 (c) we have two violations. Firstly $e_1 \#_\mu e_2$. But $(\lambda(e_1) = \lambda(e_2)) \notin D$. Secondly we have $(\lambda(e_3), \lambda(e_4)) \in D$ but $e_3 \not\leq e_4$, $e_4 \not\leq e_3$ and $(e_3, e_4) \notin \#$.

Example 3.2 *In fig. 3.2 we show an infinite trace event structure over (DR_0, I_0).*

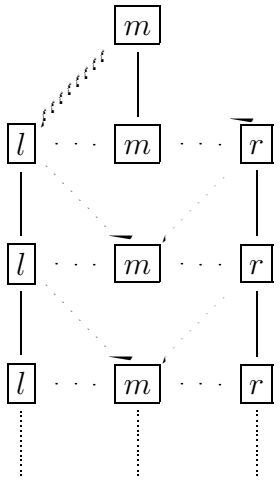

Figure 3.2

Example 3.3 *Let (DR_1, I_1) be the trace alphabet with $DR_1 = \{l, r\}$ and $I_1 = \emptyset$. Then every trace over this trace alphabet is a singleton. In figure 3.3 we show a trace event structure over (DR_1, I_1). It may be viewed as an event structure representation of the full binary tree DR_1^*.*

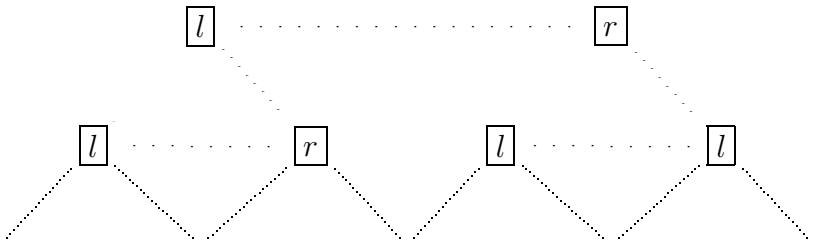

Figure 3.3

Let $ES_i = (E_i, \leq_i, \#_i, \lambda_i)$, $i = 1, 2$ be a pair of DR-labelled prime event structures. We say that ES_1 and ES_2 are isomorphic – and denote this by $ES_1 \equiv ES_2$ – iff there exists a bijection $f : E_1 \to E_2$ such that $e_1 \leq_1 e_1'$ iff $f(e_1) \leq_2 f(e_1')$ and $e_1 \#_1 e_1'$ iff $f(e_1) \#_2 f(e_1')$ for every $e_1, e_1' \in E_1$. Furthermore we require $\lambda_2(f(e_1)) = \lambda_1(e_1)$ for every $e_1 \in E_1$.

Let $\mathcal{TRS}(DR, I)$ be the class of trace structures over (DR, I) (written from now on as just $\mathcal{TRS}$). Let $\mathcal{TES}(DR, I)$ be the class of trace event structures over (DR, I) (written from now on as $\mathcal{TES}$). Using the maps $tes : \mathcal{TRS} \to \mathcal{TES}$ and $est : \mathcal{TES} \to \mathcal{TRS}$ we will bring out the fact that $\mathcal{TRS}$ and $\mathcal{TES}$ are different but equivalent descriptions of the same class of objects.

Let $B \in \mathcal{TRS}$. Then $tes(B) = (E, \leq, \#, \lambda)$ where:

- $E = pr(B)$.
- $\leq$ is $\sqsubseteq$ restricted to $pr(B) \times pr(B)$.
- $\# = \{(\sigma, \sigma') \mid \sigma, \sigma' \in pr(B) \text{ and } \sigma \not\uparrow \sigma'\}$.
  (Recall that $\sigma \uparrow \sigma'$ iff there exists $\sigma'' \in TR$ such that $\sigma \sqsubseteq \sigma''$ and $\sigma' \sqsubseteq \sigma''$.)
- $\lambda(\sigma) = last(\sigma)$ for every $\sigma \in pr(B)$.

To define the map est we must consider linearizations of the configurations of a trace event structure and read off traces from these linearizations using the labelling function. Let $ES = (E, \leq, \#, \lambda)$ be a trace event structure and let $c \in C_{ES}$. Then $\rho \in E^*$ is called a linearization of c iff it is a linearization of the poset $(c, \leq_c)$ where $\leq_c$ is $\leq$ restricted to $c \times c$. More precisely, ρ is required to satisfy:

- No event $e \in E - c$ appears in ρ.
- Every event $e \in c$ appears exactly once in ρ.
- If $e, e' \in c$ with $e < e'$ then e appears before e' in ρ.

We let $lin(c)$ be the set of linearizations of the configuration c. By abuse of notation, we use λ to also denote the unique homomorphic extension of $\lambda : E \to DR$ to $\lambda : E^* \to DR^*$. In other words $\lambda(\epsilon) = \epsilon$ and $\lambda(\rho e) = \lambda(\rho)\lambda(e)$ for $\rho \in E^*$. Finally, we define $\lambda(c) = \{\lambda(\rho) \mid \rho \in lin(c)\}$.

Let $ES = (E, \leq, \#, \lambda) \in \mathcal{TES}$. Then $est(ES) = \{[\sigma] \mid \sigma \in \lambda(c)$ for some $c \in C_{ES}$. From the results of [NW] it is straightforward to establish the following:

Proposition 3.2

(i) tes is well defined. In other words, $tes(B) \in \mathcal{TES}$ for every $B \in \mathcal{TRS}$.

(ii) est is well defined. In other words, $est(ES) \in \mathcal{TRS}$ for every $ES \in \mathcal{TES}$.

(iii) $est(tes(B)) = B$ for every $B \in \mathcal{TRS}$.

(iv) $tes(est(ES)) \equiv ES$ for every $ES \in \mathcal{TES}$. □

It is in the sense of (iii) and (iv) trace event structures and trace structures represent each other. A strong version of this statement in a categorical setting can be found in [NW]. To conclude this section, we show in fig. 3.4, the trace event structure corresponding to the trace behaviour of fig. 1.1.

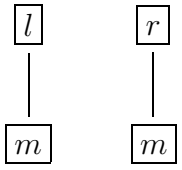

Figure 3.4

4 Regular Trace Event Structures

Our goal here is to transport the notion of regularity from trace structures to trace event structures. As before, the material in this section also will be developed w.r.t. the fixed trace alphabet (DR, I).

Let $ES = (E, \leq, \#, \lambda)$ be a trace event structure and $c \in CS$. We define $\#(c) = \{e' \mid \exists e \in c.\ e \# e'\}$. We then denote the substructure rooted at c as $ES\backslash c$ and define it to be the quadruple $ES\backslash c = (E', \leq', \#', \lambda')$ where

- $E' = E - (c \cup \#(c))$.
- $\leq'$ is $\leq$ restricted to $E' \times E'$.

- $\#'$ is $\#$ restricted to $E' \times E'$.
- λ' is λ restricted to E'.

Proposition 4.1 *Let $ES = (E, \leq, \#, \lambda)$ be a trace event structure and $c \in C_{ES}$. Then $ES\backslash c$ is also a trace event structure.*

Proof: Let $ES\backslash c = (E', \leq', \#', \lambda')$. From the definitions it is clear that $ES\backslash c$ is a DR-labelled prime event structure. We must verify that λ' respects I in the required sense.

Suppose $x, y \in E'$ with $x \lessdot' y$. We must show that $(\lambda'(x), \lambda'(y)) \in D$. It suffices to show that $x \lessdot y$ (in ES) because then $(\lambda(x), \lambda(y)) \in D$ due to the fact that ES is a trace event structure and $\lambda'(x) = \lambda(x)$ and $\lambda'(y) = \lambda(y)$.

So assume for contradiction that there exists $z \in E$ such that $x < z < y$ in ES. If $z \in E'$ then we have $x <' z <' y$ as well which would contradict $x \lessdot' y$. But $z \notin E'$ implies $z \in c$ or $z \in \#(c)$. If $z \in c$ then $c = {\downarrow} c$ leads to $x \in c$ which contradicts $x \in E'$. If $z \in \#(c)$ then for some $z' \in c$ it is the case that $z' \# z$. But then $z < y$ and hence $z' \# y$ so that $y \in \#(c)$ which contradicts $y \in E'$. Hence it must be the case that $x \lessdot y$.

Next suppose that $x, y \in E'$ with $x \#'_\mu y$. Again it suffices to show that $x \#_\mu y$ (in ES) because then we can easily conclude $(\lambda'(x), \lambda'(y)) \in D$. So assume for contradiction that $(x, y) \notin \#_\mu$. Note that $x \#'_\mu y$ implies $x \#' y$ and hence $x \# y$.

If $(x, y) \notin \#_\mu$, then there exists a pair (x', y') in $({\downarrow} x \times {\downarrow} y) \cap \#$ such that $(x', y') \neq (x, y)$. Assume without loss of generality that $x' \neq x$. Then $x' < x$ and $y' \leq y$. Hence $x' \# y$. Suppose $x' \in E'$. Then $x' <' x$ and $x' \#' y$ as well, leading to the contradiction that $(x, y) \notin \#'_\mu$.

So assume that $x' \notin E'$. Hence $x' \in c$ or $x' \in \#(c)$. If $x' \in c$ then $x' \# y$ leads to $y \in \#(c)$ which contradicts $y \in E'$. If $x' \in \#(c)$ then $x' \# z$ for some $z \in c$. But then $x' < x$ leads to $z \# x$ which in turn implies that $x \in \#(c)$. But this contradicts $x \in E'$. Hence $x \#_\mu y$ and consequently $(\lambda'(x), \lambda'(y)) \in D$ as required.

Now suppose that $x, y \in E'$ such that $(\lambda'(x), \lambda'(y)) \in D$. Then $(\lambda(x), \lambda(y)) \in$

D. Since ES is a trace event structure we must have $x \leq y$ or $y \leq x$ or $x \# y$. Hence $x \leq' y$ or $y \leq' x$ or $x \#' y$ as required. □

We can now define regularity of trace event structures.

Definition 4.2 *Let ES be a trace event structure.*

(i) $R_{ES} \subseteq C_{ES} \times C_{ES}$ *is given by:*

$$c \; R_{ES} \; c' \text{ iff } ES\backslash c \equiv ES\backslash c'.$$

(ii) ES is regular iff the equivalence relation R_{ES} is of finite index. □

It should be clear that the trace event structure ES is regular iff $est(ES)$ is a regular trace structure. It will be worthwhile to establish this connection precisely.

Recall that if B is a trace structure with $\sigma \in B$ then $B_\sigma = \{\sigma' \mid \sigma\sigma' \in B\}$.

Lemma 4.3 *Let ES be a trace event structure with $c \in CS$ and $\sigma \in \lambda(c)$. Then $est(ES\backslash c) = B_\sigma$.*

Proof: Follows easily from the definitions and the constructions in [NW]. □

Proposition 4.4 *The trace event structure ES is regular iff $est(ES)$ is a regular trace structure.*

Proof: Follows easily from lemma 4.3 and prop. 4.1. □

Let $ES = (E, \leq, \#, \lambda)$ be a trace event structure and $e \in E$. Then $\downarrow e$ can be identified with the trace $\lambda(\downarrow e)$. We now wish to show that ES is regular iff for every d, the collection of d-labelled events, viewed as a collection of traces, is recognizable. Let $ES = (E, \leq, \#, \lambda)$ be a trace event structure.

We let $E_d = \{e \mid e \in E \text{ and } \lambda(e) = d\}$. We next define, for each d, the trace language L_d^{ES} via:

$$L_d^{ES} = \{[\sigma] \mid \sigma \in \lambda(\downarrow e) \text{ for some } e \in E_d\}.$$

The matching notion for trace structures will be denoted $pr_d(B)$. More precisely, if B is a trace structure then $pr_d(B) = \{\sigma \mid \sigma \in pr(B) \text{ and } last(\sigma) = d\}$. The next observation again follows from [NW] easily.

Proposition 4.5 *Let $ES = (E, \leq, \#, \lambda)$ be a trace event structure and $est(ES) = B$. Let $f : C_{ES} \to B$ be given by $f(c) = \lambda(c)$ for every $c \in C_{ES}$. Then:*

(i) f is an isomorphism between the posets $(C_{ES}, \subseteq)$ and $(B, \sqsubseteq)$.

(ii) $f(\{\downarrow e \mid e \in E\}) = pr(B)$.

(iii) $f(\{\downarrow e \mid e \in E_d\}) = pr_d(B)$.

(iv) $L_d^{ES} = pr_d(B)$ for every d. □

We can now state the main result of this section.

Theorem 4.6 *The trace event structure ES is regular iff L_d^{ES} is recognizable for every d.*

Proof:
$\underline{\Rightarrow}$ Suppose $ES = (E, \leq, \#, \lambda)$ is regular and $est(ES) = B$. Then by prop. 4.4, $(B, \sqsubseteq)$ is regular and hence by theorem 2.5, $pr(B)$ is recognizable. Let $\mathcal{A}$ be a deterministic finite state automaton recognizing $L = ts(pr(B))$. Now recall the automaton $\mathcal{A}^{top} = (Q, \to, q_{in}, F)$ constructed in the proof of lemma 2.6. Consider the automaton $\mathcal{A}_d^{top}$ obtained from $\mathcal{A}^{top}$ by setting $\mathcal{A}_d^{top} = (Q, \to, q_{in}, F_d)$ where $F_d = \{\{d\}\}$. Let L_d be the language recognized by $\mathcal{A}_d^{top}$. It is easy to verify that $L_d = ts(pr_d(TR))$. Hence $ts(pr_d(B)) = L \cap L_d$. Consequently $pr_d(B)$ is recognizable for each d. But now from prop. 4.5 (part (iv)) we also have that L_d^{ES} is recognizable for each d.

$\Leftarrow$ Suppose L_d^{ES} is recognizable for each d. Then by prop. 4.5 (part (iv)), $pr_d(B)$ is recognizable for each d where $B = est(ES)$. But this implies that $pr(B) = \bigcup_{d \in DR} pr_d(B)$ is recognizable and hence, by theorem 2.5, B is regular. Prop. 4.4 now tells us that ES is also regular. □

5 Labelled Trace Event Structures

Through the rest of the paper fix Σ, a finite non-empty set of labels.

Definition 5.1 *A Σ-labelled trace event structure is a pair $LES = (ES, \varphi)$ where $ES = (E, \leq, \#, \lambda)$ is a trace event structure (over (DR, I)) and $\varphi : E \to \Sigma$ is a labelling function.* □

Note that LES has an "internal" labelling function λ. This function orients the events along the directions in DR while respecting the independence relation I in the manner specified in section 3. On the other hand, φ is an "external" and, in the present setting, unrestricted labelling function which labels the events by members of Σ.

One could ask why not start with Σ-labelled trace structures? The answer is that a variety of negative results are at present available concerning logics interpreted over trace structures and related models (see for instance [LPRT, PK]). These results suggest that trace structures accompanied by unrestricted labelling functions will not be a tractable collection of objects. Indeed an observation due to Walukiewicz [Wal] suggests that even trace structures accompanied by unrestricted labelling functions will not constitute a tractable collection of objects. (We will say a little more about this in the concluding section.) Nevertheless we feel that it will be fruitful to study of Σ-labelled trace event structures and identify the required restrictions in some suitable logical and/or automata theoretic framework. These notions can then be transported, if necessary, to trace structures at a later stage.

Since Σ is fixed we shall often say "labelled" to mean "Σ-labelled". Let $LES_i = (ES_i, \varphi_i)$, $i = 1, 2$ be a pair of labelled trace event structures with $ES_i = (E_i, \leq_i, \#_i, \lambda_i)$. Then LES_1 and LES_2 are said to be isomorphic

(also denoted by abuse of notation as $LES_1 \equiv LES_2$) iff there exists a trace event structure isomorphism $f : E_1 \to E_2$ such that $\lambda_1(e_1) = \lambda_2(f(e_1))$ for every $e_1 \in E_1$. Thus the isomorphism is also required to respect the external labelling.

Definition 5.2 *Let $LES = (ES, \varphi)$ be a labelled trace event structure with $ES = (E, \leq, \#, \lambda)$.*

(i) Suppose $c \in C_{ES}$. Then $LES\backslash c = (ES', \varphi')$ where $ES' = (E', \leq', \#', \lambda') = ES\backslash c$ and φ' is φ restricted to E'.

(ii) The equivalence relation $R_{LES} \subseteq C_{ES} \times C_{ES}$ is given by:

$$c \; R_{LES} \; c' \text{ iff } LES\backslash c \equiv LES\backslash c'.$$

(iii) LES is regular iff R_{LES} is of finite index. □

The main result of this section is a conservative extension of the classical characterization of regular Σ-labelled trees [Tho]. To formulate this result let $LES = (ES, \varphi)$ be a labelled trace event structure with $ES = (E, \leq, \#, \lambda)$. Let $x \in \Sigma$. Then $L_x^{LES} \subseteq TR$ is defined as:

$$[\sigma] \in L_x^{LES} \text{ iff } \sigma \in \lambda(\downarrow e) \text{ for some } e \in \varphi^{-1}(x).$$

Theorem 5.3 *The labelled trace event structure $LES = (ES, \varphi)$ is regular iff L_x^{LES} is a recognizable trace language for every $x \in \Sigma$.*

Proof: Let $ES = (E, \leq, \#, \lambda)$. Consider the trace alphabet $(\widehat{DR}, \hat{I})$ where $\widehat{DR} = DR \times \Sigma$ and $\hat{I} \subseteq \widehat{DR} \times \widehat{DR}$ is given by

$$((a, x), (b, y)) \in \hat{I} \text{ iff } (a, b) \in I.$$

Let $\widehat{ES} = (E, \leq, \#, \hat{\lambda})$ where $\hat{\lambda} : E \to \widehat{DR}$ is given by:

$$\hat{\lambda}(e) = (d, x) \text{ iff } \lambda(e) = d \text{ and } \varphi(e) = x.$$

Clearly $\widehat{ES}$ is a trace event structure over $(\widehat{DR}, \hat{I})$. Moreover the labelled trace event structure LES is regular iff the trace event structure $\widehat{ES}$ is regular. By theorem 4.6, $\widehat{ES}$ is regular iff $L_{\hat{d}}^{\widehat{ES}}$ is a recognizable trace language for every $\hat{d} \in \widehat{DR}$. Now define, for each $x \in \Sigma$, the trace language $L_x^{\widehat{ES}}$ via:

$$L_x^{\widehat{ES}} = \bigcup \{L_{\hat{d}}^{\widehat{ES}} \mid \hat{d} \in DR \times \{x\}\}.$$

Clearly $L_x^{\widehat{ES}}$ is recognizable for each x if $L_{\hat{d}}^{\widehat{ES}}$ is recognizable for each $\hat{d} \in \widehat{DR}$. Conversely, suppose $L_x^{\widehat{ES}}$ is recognizable for each $x \in \Sigma$. Then it is easy to verify (using the automaton $\mathcal{A}^{top}$ constructed in the proof of lemma 2.6) that $L_{\hat{d}}^{\widehat{ES}}$ is recognizable for each $\hat{d} \in \widehat{DR}$.

But now the required result follows from the easy observation that $L_x^{ES} = L_x^{\widehat{ES}}$ for each $x \in \Sigma$. □

6 A Net Theoretic Characterization

The basic result shown in [NPW] says that every 1-safe Petri net unfolds into a prime event structure. Later results appearing in a uniform setting (see [NW] for exact references) show in fact that 1-safe Petri nets and prime event structures represent each other in a strong sense. Based on this connection we show here that regular trace event structures and (labelled) *finite* 1-safe Petri nets are strongly related to each other.

We shall define here a 1-safe Petri net to be a quadruple $\mathcal{N} = (B, E, F, M_{in})$ where (B, E, F) is a net and $M_{in} \subseteq B$ is the initial marking. (B, E, F) is a net in the sense B is a set of conditions, E is the set of events with $B \cap E = \emptyset$. Furthermore $F \subseteq (B \times E) \cup (E \times F)$ is the flow relation. We say that $\mathcal{N}$ is finite if both B and E are finite sets. From now by "Petri net" we shall mean "1-safe Petri net". As usual, for $x \in B \cup E$, we set ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (y, x) \in F\}$. The dynamics of $\mathcal{N}$ are captured by the associated transition system $TS_{\mathcal{N}} = (RM_{\mathcal{N}}, \rightarrow_{\mathcal{N}}, M_{in})$ where $RM_{\mathcal{N}} \subseteq 2^B$ and $\rightarrow_{\mathcal{N}} \subseteq RM_{\mathcal{N}} \times E \times RM_{\mathcal{N}}$ are the least sets satisfying:

- $M_{in} \in RM_{\mathcal{N}}$.
- Suppose $M \in RM_{\mathcal{N}}$ and $e \in E$ such that ${}^\bullet e \subseteq M$ and $(e^\bullet - {}^\bullet e) \cap M = \emptyset$. Then $M' \in RM_{\mathcal{N}}$ and $M \xrightarrow{e}_{\mathcal{N}} M'$ where $M' = (M - {}^\bullet e) \cup e^\bullet$.

$RM_{\mathcal{N}}$ is the set of reachable markings of $\mathcal{N}$. $FS_{\mathcal{N}}$, the set of firing sequences of the Petri net $\mathcal{N} = (B, E, F, M_{in})$ is the least subset of E^* defined as follows. In doing so, it will be convenient to also build up the relation $\xrightarrow[*\ \mathcal{N}]{}$

- $\epsilon \in FS_{\mathcal{N}}$ and $M_{in} \xrightarrow[*\ \mathcal{N}]{} M_{in}$
- Suppose $\tau \in FS_{\mathcal{N}}$, $M_{in} \xrightarrow[*\ \mathcal{N}]{\tau} M$ and $M \xrightarrow[\mathcal{N}]{e} M'$. Then $\tau e \in FS_{\mathcal{N}}$ and $M_{in} \xrightarrow[*]{\tau e} M'$.

In what follows, we will define notion of an *es*-unfolding (event structure unfolding) of only finite 1-safe Petri nets. This silliness is mainly due to the fact we have chosen to phrase the basic notions of trace theory in terms of finite trace alphabets in order to appeal to the automaton theoretic constructions for recognizable trace languages. The material to follow that is concerned with unfoldings can however be easily modified to apply to the whole class of Petri nets.

The finite Petri net $\mathcal{N} = (B, E, F, M_{in})$ gives rise to the trace alphabet $(E, I_{\mathcal{N}})$ where $I_{\mathcal{N}} = \{(e_1, e_2) \mid ({}^\bullet e_1 \cup e_1^\bullet) \cap ({}^\bullet e_2 \cup e_2^\bullet) = \emptyset\}$. For $\tau \in E^*$ let $[\tau]_{\mathcal{N}}$ denote the $\sim_{I_{\mathcal{N}}}$-equivalence class containing τ.

Now for every $\tau \in FS_{\mathcal{N}}$ it is easy to see that $[\tau]_{\mathcal{N}} \subseteq FS_{\mathcal{N}}$. Clearly $FS_{\mathcal{N}}$ is prefix-closed. Furthermore, $\tau e, \tau e' \in FS_{\mathcal{N}}$ and $e\ I_{\mathcal{N}}\ e'$ implies $\tau e e' \in FS_{\mathcal{N}}$. Thus $B_{\mathcal{N}} = \{[\tau]_{\mathcal{N}} \mid \tau \in FS_{\mathcal{N}}\}$ is a trace structure with $ts(B_{\mathcal{N}}) = FS_{\mathcal{N}}$. We let $ES_{\mathcal{N}}$ denote $tes(B_{\mathcal{N}})$ and call $ES_{\mathcal{N}}$ the *es*-unfolding of $\mathcal{N}$. Note that according to prop. 3.2, $ES_{\mathcal{N}}$ is a trace event structure over $(E, I_{\mathcal{N}})$.

Let X be a finite non-empty set of labels. Then an X-labelled Petri net is a pair $\mathcal{LN} = (\mathcal{N}, lb)$ where $\mathcal{N} = (B, E, F, M_{in})$ is a Petri net and $lb : E \to X$ is a labelling function.

Let $\mathcal{LN} = (\mathcal{N}, lb)$ be an X-labelled Petri net and $ES_{\mathcal{N}} = (\hat{E}, \hat{\leq}, \hat{\#}, \hat{\lambda})$. Then we let $ES_{\mathcal{LN}}$ denote the X-labelled prime event structure $ES_{\mathcal{LN}} = (\hat{E}, \hat{\leq}, \hat{\#}, lb\ o\ \hat{\lambda})$ and call it the *es*-unfolding of $\mathcal{LN}$.

We can now state our net theoretic characterisation of regular trace event structures.

Theorem 6.1 *The trace event structure ES over (DR, I) is regular iff there exists a finite DR-labelled Petri net $\mathcal{LN} = (\mathcal{N}, lb)$ such that ES is isomorphic to $ES_{\mathcal{LN}}$.* □

One half of the theorem is quite easy to prove.

Lemma 6.2 *Suppose $\mathcal{LN} = (\mathcal{N}, lb)$ is a finite DR-labelled Petri net such that $ES_{\mathcal{LN}}$ is a DR-labelled trace event structure. Then $ES_{\mathcal{LN}}$ is regular.*

Proof: Let $ES_{\mathcal{N}} = (\hat{E}, \hat{\leq}, \hat{\#}, \hat{\lambda})$ so that $ES_{\mathcal{LN}} = (\hat{E}, \hat{\leq}, \hat{\#}, lb\ o\ \hat{\lambda})$. Let $ES = (\hat{E}, \hat{\leq}, \hat{\#})$ and $\mathcal{N} = (B, E, F, c_{in})$. Recall that $ES_{\mathcal{N}}$ is a trace event structure over $(E, I_{\mathcal{N}})$. Define now the map $\mu : C_{ES} \to RM_{\mathcal{N}}$ via $\mu(c) = M$ provided there exists $\tau \in \hat{\lambda}(c)$ such that $M_{in} \xrightarrow[*\ \mathcal{N}]{\tau} M$.

It is easy to check that μ is well defined. Next define the equivalence relation $R_{\mathcal{N}} \subseteq C_{ES} \times C_{ES}$ as:

$$c\ R_{\mathcal{N}}\ c' \text{ iff } \mu(c) = \mu(c').$$

The number of equivalence classes of this equivalence relation is at most $2^{|B|}$.

Claim Suppose $c\ R_{\mathcal{N}}\ c'$. Then $c\ R_{ES_{\mathcal{N}}}\ c'$ and in fact $c\ R_{ES_{\mathcal{LN}}}\ c'$.

If the claim holds then the index of $R_{ES_{\mathcal{LN}}}$ is at most $2^{|B|}$ and hence $ES_{\mathcal{LN}}$ is regular. To see that the claim holds let $\tau \in \hat{\lambda}(c)$ and $\tau' \in \hat{\lambda}(c')$. Then from $\mu(c) = \mu(c')$ it follows that $\forall \tau'' \in E^*$, $\tau\tau' \in FS_{\mathcal{N}}$ iff $\tau\tau'' \in FS_{\mathcal{N}}$. This observation at once leads to the claim. □

To prove the second half of theorem 3.1, we will prune away some local transitions of the automaton $\mathcal{A}^{pr}$ that was constructed to prove lemma 2.10. To be precise, let $\mathcal{A} = (\{S_p\}, \{\rightarrow_d\}, S_{in}, F)$ be a deterministic and complete asynchronous automaton operating over $\{DR_p\}$. Recall that p ranges over $\mathcal{P}$, the set of maximal D-cliques of (DR, I) with $DR_p = p$ for each p.

Let $\mathcal{A}^{pr} = (\{\hat{S}_p\}, \{R_d\}, \hat{S}_{in}, \hat{F})$ be the deterministic complete asynchronous automaton constructed from $\mathcal{A}$ as in the proof of lemma 2.10. We now define the asynchronous automaton $\mathcal{B}^{pr} = (\{\tilde{S}_p\}, \{\rightsquigarrow_d\}, \tilde{S}_{in}, \tilde{F})$ as follows:

- $\tilde{S}_p = \hat{S}_p$ for each p.
- For each d, $\rightsquigarrow_d$ is the least subset of R_d which satisfies: Suppose $(\hat{s}, \hat{t}) \in R_d$ with $\hat{s}(p) = (s^{(p)}, \nu_p)$ and $\hat{t}(p) = (t^{(p)}, \delta_p)$ for each $p \in loc(d)$. Then $(\hat{s}, \hat{t}) \in \rightsquigarrow_d$ iff $s^{(p)} \in F$ and $t^{(p)} \in F$ for each $p \in loc(d)$. Recall that $s^{(p)} \in S_{\mathcal{P}}$, the set of global states of $\mathcal{A}$ and F is the set of global final states of $\mathcal{A}$.
- $\tilde{S}_{in} = \hat{S}_{in}$.
- $\tilde{F}$ is the least subset of $\tilde{S}_{\mathcal{P}}$ (the set of global states of $\mathcal{B}^{pr}$) which satisfies: Suppose $\sigma \in DR^*$ such that $\tilde{\rho}_\sigma$ is a run of $\mathcal{B}^{pr}$ with $\tilde{\rho}_\sigma(\sigma) = \tilde{s}$. Then $\tilde{s} \in \tilde{F}$.

It is easy to observe that $\mathcal{B}^{pr}$ is also a deterministic asynchronous automaton. However it might not be complete. In fact, its characteristic feature is that it accepts a word iff it has a run over the word.

Lemma 6.3 *Let B be a regular trace structure over (DR, I) with $ts(B) = L$. Then there exists an asynchronous automaton $\mathcal{B}$ operating over $\{DR_p\}$ such that $\mathcal{L}(\mathcal{B}) = L$. Moreover, for every $\sigma \in DR^* - L$, $\mathcal{B}$ has* no *run over σ.*

Proof: Since B is regular, L is recognizable. Let $\mathcal{A}$ be a deterministic and complete asynchronous automaton with $\mathcal{A} = (\{S_p\}, \{\rightarrow_d\}, S_{in}, F)$ such that $\mathcal{L}(\mathcal{A}) = L$. Let $\mathcal{A}^{pr} = (\{\hat{S}_p\}, \{R_d\}, \hat{S}_{in}, \hat{F})$ be the automaton constructed from $\mathcal{A}$ as in the proof of lemma 2.10. Finally, let $\mathcal{B}^{pr}$ be the deterministic asynchronous automaton constructed from $\mathcal{A}^{pr}$ as detailed above. We wish to

argue that $\mathcal{B}^{pr}$ has the properties required by the lemma. We shall decompose the argument into three steps.

Claim 1 $\mathcal{L}(\mathcal{A}^{pr}) = \mathcal{L}(\mathcal{A})$.

Claim 2 $\mathcal{L}(\mathcal{B}^{pr}) = \mathcal{L}(\mathcal{A}^{pr})$.

Claim 3 Let $\sigma \in DR^*$. Then $\sigma \in \mathcal{L}(\mathcal{B}^{pr})$ iff $\mathcal{B}^{pr}$ has a run over σ.

To see that claim 1 must hold, we note that, by the construction of $\mathcal{A}^{pr}$, $\sigma \in \mathcal{L}(\mathcal{A}^{pr})$ iff $\downarrow^p(\sigma) \subseteq \mathcal{L}(\mathcal{A})$ for every p. But from the fact that B is a trace structure it follows from prop. 2.8, that $\sigma \in \mathcal{L}(\mathcal{A})$ iff $\downarrow^p(\sigma) \subseteq \mathcal{L}(\mathcal{A})$ for every p. Hence indeed $\mathcal{L}(\mathcal{A}^{pr}) = \mathcal{L}(\mathcal{A})$.

To settle claim 2, we consider $\sigma \in DR^*$ and proceed by induction on $|\sigma|$ to show that $\tilde{\rho} : Prf(\sigma) \to \tilde{S}_{\mathcal{P}}$ is a run of $\mathcal{B}^{pr}$ over σ iff $\tilde{\rho}_\sigma$ is the accepting run of $\mathcal{A}^{pr}$ over σ.

Suppose $|\sigma| = 0$ so that $\sigma = \epsilon$. Since L is prefix-closed, $\epsilon \in L$ and hence the basis step follows immediately from the definitions.

So suppose $\sigma = \sigma' d$. First assume that $\tilde{\rho} : Prf(\sigma) \to \tilde{S}$ is a run of $\mathcal{B}^{pr}$ over σ. Let $\tilde{\rho}_{\sigma'}$ be the restriction of $\tilde{\rho}$ to $Prf(\sigma')$. By the induction hypothesis, $\tilde{\rho}_{\sigma'}$ is the accepting run of $\mathcal{A}^{pr}$ over σ'. Let $\tilde{\rho}(\sigma') = \hat{s}$ and $\tilde{\rho}(\sigma' d) = \hat{t}$. Then $((\hat{s})d, (\hat{t})d) \in \rightsquigarrow_d \subseteq R_d$ and consequently $\tilde{\rho}$ is also the run of $\mathcal{A}^{pr}$ over σ.

Suppose $\tilde{\rho}$ is not an accepting run of $\mathcal{A}^{pr}$ over σ. Then $\hat{t} \notin \hat{F}$. Hence, by the definition of $\hat{F}$, it must be the case that $t^{(p)} \notin F$ for some p. Let $\hat{t}(p) = (t^{(p)}, \delta_p)$ for each p.

If $p \notin loc(d)$ then $\hat{s}(p) = \hat{t}(p)$ and hence $\hat{s} \notin \hat{F}$. But this contradicts the fact that $\tilde{\rho}_{\sigma'}$ is the accepting run of $\mathcal{A}^{pr}$ over σ'. So suppose that $p \in loc(d)$. We now have $t^{(p)} \notin F$ and $p \in loc(d)$ which contradicts that $((\hat{s})_d, (\hat{t})_d) \in \rightsquigarrow_d$. Hence it must be the case that $\hat{t} \in \hat{F}$.

On the other hand, if $\tilde{\rho} : prf(\sigma) \to \tilde{S}$ is the accepting run of $\mathcal{A}^{pr}$ over σ. Then from the fact that L is prefix-closed, it follows that $\sigma' \in L$ and hence $\hat{s} \in \hat{F}$ where $\tilde{\rho}(\sigma) = \hat{t}$ and $\tilde{\rho}(\sigma') = \hat{s}$. Consequently $((\hat{s})_d, (\hat{t})_d) \in \rightsquigarrow_d$. But then by the induction hypothesis, $\tilde{\rho}$ restricted to $prf(\sigma')$ is a run of $\mathcal{B}^{pr}$ over σ'. Consequently $\tilde{\rho}$ is a run of $\mathcal{B}^{pr}$ over $\sigma = \sigma' d$.

Claim 3 follows at once from the definition of $\tilde{F}$. Hence $\mathcal{B}^{pr}$ has the properties required by the lemma. $\Box$

With these preparations out of the way, we can now deal with the second half of the proof of theorem 6.1.

Lemma 6.4 *Suppose ES is a regular trace structure over (DR, I). Then there exists a finite DR-labelled Petri net $\mathcal{LN} = (\mathcal{N}, lb)$ such that ES is isomorphic to $ES_{\mathcal{LN}}$.*

Proof: Let $est(ES) = B$ and $ts(B) = L$. B is a regular trace structure by prop. 4.4 and hence $B \subseteq TR$ is recognizable by prop. 2.2. Consequently, by theorem 2.7, there exists a deterministic complete asynchronous automaton $\mathcal{A} = (\{S_p\},$
$\{\rightarrow_d\}, S_{in}, F)$ such that $\mathcal{L}(\mathcal{A}) = L$. Let $\mathcal{A}^{pr} = (\{\hat{S}_p\}, \{R_d\}, \hat{S}_{in}, \hat{F})$ and $\mathcal{B}^{pr} = (\{\tilde{S}_p\}, \{\rightsquigarrow_d\}, \tilde{S}_{in}, \tilde{F})$ be the deterministic asynchronous automaton constructed from $\mathcal{A}$ as in the proof of the previous lemma. In what follows, we will assume without loss of generality that if $p \neq q$ then $\tilde{S}_p \cap \tilde{S}_q = \emptyset$ and that if $a \neq b$ then $\rightsquigarrow_a \cap \rightsquigarrow_b = \emptyset$.

We define the DR-labelled finite Petri net $\mathcal{LN} = (\mathcal{N}, lb)$ with $\mathcal{N} = (B, E, F, M_{in})$ as follows:

- $B = \tilde{S} = \bigcup_p \tilde{S}_p$.
- $E = \bigcup_d \rightsquigarrow_d$.
- $F = \{(\tilde{s}(p), (\tilde{s}, \tilde{t})) \mid (\tilde{s}, \tilde{t}) \in \rightsquigarrow_d \text{ and } p \in loc(d) \text{ for some } d\}$
  $\cup \{((\tilde{s}, \tilde{t}), \tilde{t}(p)) \mid (\tilde{s}, \tilde{t}) \in \rightsquigarrow_d \text{ and } p \in loc(d) \text{ for some } d\}$.
- $M_{in} = \{\tilde{s}_{in}(p) \mid p \in \mathcal{P}\}$ where $\tilde{S} = \{\tilde{s}_{in}\}$.
- For each $e \in E$, $lb(e) = d$ provided $e \in \rightsquigarrow_d$.

By abuse of notation let the unique extension of lb to E^* (with co-domain DR^*) also be denoted lb.

Claim 1 Let $\tau \in E^*$. Then $\tau \in FS_{\mathcal{N}}$ iff $lb(\tau) \in L$.

Claim 2 Let $\tau, \tau' \in FS_{\mathcal{N}}$. Then $\tau \sqsubseteq_{I_{\mathcal{N}}} \tau'$ iff $lb(\tau) \sqsubseteq lb(\tau')$.

It is tedious but straightforward to prove these two claims. The required result then follows easily. □

Thus theorem 6.1 is now established. A simple consequence is that one can now obtain a similar characterization of regular Σ-labelled trace event structures in terms of finite Σ-labelled Petri nets.

Corollary 6.5 *Let ES be a Σ-labelled trace event structure over (DR, I). Then ES is regular iff there exists a finite Σ-labelled Petri net $\mathcal{LN} = (\mathcal{N}, lb)$ such that ES is isomorphic to $ES_{\mathcal{LN}}$.*

Proof: Consider the trace alphabet $(\widehat{DR}, \hat{I})$ where $\widehat{DR} = DR \times \Sigma$ and $\hat{I} = \{((a,x),(b,y)) \mid a\ I\ b\}$. Then following the lines of the proof of theorem 6.1, we can easily extract the corollary from theorem 6.4. □

7 Conclusion

In this paper we have introduced the notion of a regular trace event structure. We view trace event structures as a common generalization – in an event-based framework – of trees and Mazurkiewicz traces. As pointed out earlier, the notion of a trace event structure and hence that of a trace structure admitted here is rather generous. Even trace event structures whose underlying prime event structures have an empty conflict relation will be allowed. Surely this is not what one intends when talking about branching time behaviours. In the case of (finite) trees the required branching structure is guaranteed by typing the nodes with arities or by demanding that each node should have all its successors or none of its successors present in the tree. Something similar will have to be done for trace event structure. The possibilities that are available are many. It is not clear at present what is the best way to proceed.

We do not know at present what is the proper monadic second order logic for trace event structures. As mentioned earlier, blind generalization over configurations (i.e. over traces in the setting of trace structures) will at once result in an undecidable logic. Surprisingly enough, it turns out, as pointed out by Walukiewicz [Wal] that a logic over trace event structures which permits quantification over events will also be undecidable. To see this, consider the trace alphabet (DR_0, I_0) with $DR_0 = \{l, m, r\}$ and $I_0 = \{(l, r), (r, l)\}$. Then the events corresponding to the set of prime traces $\{l^{n_1}, r^{n_2}m \mid n_1 \geq 0, n_2 \geq 0\}$ can be used to code up the two-dimensional grid. Thus we must find a suitable restriction on trace event structures in order to arrive at branching time legics that are decidable and hopefully interesting.

Recently, Huhn and Niebert [HN] have proposed what may be called confusion-free trace event structures as a suitable class of objects. They also formulate a related class of automata and show that the emptiness problem for these automata is decidable. We feel that confusion-freeness is a much too drastic restriction and one should look for a larger class of trace event structures. It is however not clear at present what this larger class ought to be.

Our notion of regular trace event structures gives rise to an interesting conjecture which appears to be difficult to prove. First note that for a prime event structure $ES = (E, \leq, \#)$ we can define the equivalence relation $R_{ES} \subseteq C_{ES} \times C_{ES}$ via:

$$c \; R_{ES} \; c' \text{ iff } ES\backslash c \equiv ES\backslash c'.$$

(The notion of isomorphism that $\equiv$ captures is the natural one.)

Next we say that $e \in E$ is enabled at $c \in C_{ES}$ iff $e \notin c$ and $c \cup \{e\} \in C_{ES}$. Let $en(c)$ be the set of events enabled at the configuration c.

Finally, we say that ES is regular iff R_{ES} is of finite index *and* there exists an integer k such that $|en(c)| \leq k$ for every $c \in C_{ES}$.

Conjecture The prime event structure ES is regular iff it is isomorphic to the *es*-unfolding of a *finite* 1-safe Petri net.

References

[DM] V. Diekert and A. Muscholl: Deterministic Asynchronous Automata for Infinite Traces, *Acta Informatica*, 31, (1994) 379-397.

[GW] P. Godefroid and P. Wolper: A Partial Approach to Model Checking, *Information and Computation*, 110 (1994) 305-326.

[HN] M. Huhn and P. Niebert: Towards Automata for branching time and partial order, *Proc. of CONCUR'96, LNCS 1119* (1996), 611-627.

[LPRT] K. Lodaya, R. Parikh, R. Ramanujam and P.S. Thiagarajan: A Logical Study of Distributed Transition Systems, *Information and Computation*, 119 (1995) 91-118.

[MS] M. Mukund and M.S. Sohoni: Keeping track of the latest gossip: Bounded time-stamps suffice, *Proc. FST & TSC'93, LNCS 761* (1993) 388-399.

[NPW] M. Nielsen, G. Plotkin and G. Winskel: Petri nets, Event structures and Domains I, *Theor. Comput. Sci.*, 13 (1980) 86-108.

[NW] M. Nielsen and G. Winskel: Trace Structures and other Models for Concurrency, In: V. Diekert and G. Rozenberg (eds.), *The Book of Traces*, World Scientific, Singapore (1995) 271-305.

[Pel] D. Peled: All from one, one for all: On model checking using representatives. *Proceedings of CAV'93, LNCS 697* (1993) 409-424.

[PK] W. Penczek and R. Kuiper: Traces and Logic, In: V. Diekert and G. Rozenberg (eds.), *The Book of Traces*, World Scientific, Singapore (1995) 307-379.

[Rab] M.O. Rabin: Decidability of second order theories and automata on infinite trees, *Trans. Amer. Math. Soc.*, 141: 1-35, 1969.

[RT] B. Rozoy and P.S. Thiagarajan: Event Structures and Trace Monoids, *Theoretical Computer Science*, 91 (1991) 285-313.

[Thi] P.S. Thiagarajan: A Trace Based Extension of Linear Time Temporal Logic, *Proc. 9th IEEE LICS* (1994) 438-447.

[Tho] W. Thomas: Automata on infinite objects, In: J. van Leeuwen (ed.) *Handbook of Theoretical Computer Science, Volume B*, North-Holland, Amsterdam (1990) 130-191.

[Val] A. Valmari: Stubborn Sets for Reduced State Space Generation, *LNCS 483* (1990) 491-515

[Wal] I. Walukiewicz. Private communication.

[Zie] W. Zielonka: Notes on finite asynchronous automata, *R.A.I.R.O. – Inf. Théor. et Appl.*, 21 (1987) 99-135.

Recent Publications in the BRICS Report Series

RS-96-32 **P. S. Thiagarajan.** ***Regular Trace Event Structures.*** **September 1996. 34 pp.**

RS-96-31 **Ian Stark.** ***Names, Equations, Relations: Practical Ways to Reason about `new'.*** **September 1996. ii+22 pp.**

RS-96-30 **Arne Andersson, Peter Bro Miltersen, and Mikkel Thorup.** ***Fusion Trees can be Implemented with AC^0 Instructions only.*** **September 1996. 8 pp.**

RS-96-29 **Lars Arge.** ***The I/O-Complexity of Ordered Binary-Decision Diagram Manipulation.*** **August 1996. 35 pp. An extended abstract version appears in Staples, Eades, Kato, and Moffat, editors,** ***Algorithms and Computation: 6th International Symposium,*** **ISAAC '95 Proceedings, LNCS 1004, 1995, pages 82–91.**

RS-96-28 **Lars Arge.** ***The Buffer Tree: A New Technique for Optimal I/O Algorithms.*** **August 1996. 34 pp. This report is a revised and extended version of the BRICS Report RS-94-16. An extended abstract appears in Akl, Dehne, Sack, and Santoro, editors,** ***Algorithms and Data Structures: 4th Workshop,*** **WADS '95 Proceedings, LNCS 955, 1995, pages 334–345.**

RS-96-27 **Devdatt Dubhashi, Volker Priebe, and Desh Ranjan.** ***Negative Dependence Through the FKG Inequality.*** **July 1996. 15 pp.**

RS-96-26 **Nils Klarlund and Theis Rauhe.** ***BDD Algortihms and Cache Misses.*** **July 1996. 15 pp.**

RS-96-25 **Devdatt Dubhashi and Desh Ranjan.** ***Balls and Bins: A Study in Negative Dependence.*** **July 1996. 27 pp.**

RS-96-24 **Henrik Ejersbo Jensen, Kim G. Larsen, and Arne Skou.** ***Modelling and Analysis of a Collision Avoidance Protocol using SPIN and UPPAAL.*** **July 1996. 20 pp. Presented at** ***DIMACS Workshop SPIN96 – 2nd International SPIN Verification Workshop on Algorithms, Applications, Tool Use, Theory*** **(Rutgers University, New Jersey, USA, August 5, 1996).**