On the Connection of Partial Order Logics and Partial Order Reduction Methods

by

Pieter Niebert and Wojciech Penczek

95/15
On the Connection of Partial Order Logics and Partial Order Reduction Methods

Peter Niebert* and Wojciech Penczek**

1 Institut für Informatik
Universität Hildesheim
niebert@informatik.uni-hildesheim.de
2 Dept. of Computer Science
Eindhoven University of Technology
wojtek@info.win.tue.nl

Abstract. We examine the connection between “equivalence robust” subsets of propositional temporal logics (LTL and CTL*), for which partial order reduction methods can be applied in model checking, and partial order logics and equivalences.

For the linear case we show how to naturally translate “equivalence robust” LTL properties into Thagagarajan’s linear time temporal logic for traces (TrPTL), substantiating the claim that partial order logics have the right syntax for equivalence robust properties.

For the branching case we define a parametrised dependency relation \((D, V)\) yielding an \((D, V)\)-equivalence notion for trees that generalizes Mazurkiewicz’s trace equivalence. Then, we show that under some condition \((D, V)\)-equivalent trees are stuttering equivalent and therefore cannot be distinguished by any CTL-X formulas. We prove that partial order reductions for CTL-X give \((D, V)\)-equivalent trees.

Our approach can be used as a semantic basis for branching time partial order logics for expressing equivalence robust branching time properties.

1 Introduction

Temporal logic is an established formalism to specify the behaviour of concurrent systems. The most commonly used approach is to interpret temporal logics on the interleaving semantics of a computation in the distributed system. This means that the concurrent system is given a transition system semantics generating all the interleavings. However this approach gives rise to the state explosion problem, because the transition system may grow exponentially with the number of system components.

Several approaches have been undertaken to overcome the state explosion problem, including partial order reductions, which attempt to reduce the number of states of the transition system to be investigated, see e.g. [Val89, Val90, GW93a, GW93b]. A recent systematic approach in this direction is given in [Pe193] for linear time temporal logics and in [GKPP95] for branching time. The idea (explained for linear time) is to use a notion of sequential run equivalence, which is induced by commutativity of the “independent actions”. One equivalence class of interlevings (a Mazurkiewicz trace) can be considered as a partial ordering of events. Peled’s approach to avoid the state explosion problem uses a restriction to temporal properties, which are robust w.r.t. such commutations of independent actions. Then, the transition system can be reduced such that it generates at least one interleaving “representative” of each partially ordered run of the system, without changing the (restricted) logical properties.

* This work was partially supported by the ESPRIT Working Group No. 6067 CALIBAN (Causal Calculi Based on Nets).
** On leave from Institute of Computer Science, Polish Academy of Sciences, Warsaw. Supported in part by De stichting informatica-onderzoek in Nederland (SION) and by KBN grant No. 8 T11C 029 08.
On the other hand, there have been efforts in generalizing temporal logics to partial order semantics, recently by Thiagarajan with a "trace based" temporal logic (TrPTL) [Thi94]. This logic is directly interpreted over partially ordered runs. Model checking for TrPTL is reduced to checking non-emptiness of the intersection of a Büchi Asynchronous Automaton (BAA) representing the program and a BAA representing the negation of the formula. As in case of LTL, partial order reduction methods can be used for alleviating the state explosion problem arising when non-emptiness is checked [PP95]. Since TrPTL formulas cannot distinguish between (equivalent) interleavings of the same partially ordered run, the partial order reduction method deals with the unrestricted language of TrPTL.

In this paper we show that TrPTL contains the efficiently checkable subset of LTL as subset, i.e., every such formula of LTL can naturally be translated to an equivalent TrPTL formula. Moreover there is a clean syntactic classification of these formulas as subset of TrPTL.

This means that the subset of LTL properties defined in [PeI93] - while interpreted using an interleaving semantics - represent partial order temporal properties and can naturally be expressed as such.

For the branching time case currently no matching partial order temporal logic exists. Because interleaving branching time logics can distinguish between the order of independent choices, we relativize independence w.r.t. choices and a distinguished set of "visible actions", yielding a generalization of Mazurkiewicz trace equivalence to trees. We show that this equivalence meets the requirements of the partial order reduction approach, thus giving a semantic basis for a branching time partial order logic with a close connection to the interleaving logics.

2 Foundations

In this section we describe the underlying system assumptions of both works [PeI93, GKPP95] and [Thi94]. We slightly modify the exposition to simplify the bringing together of two formalisms. However there is no loss of generality.

2.1 Interpreted trees

As an abstraction of the behaviour of (deterministic) concurrent systems we consider computation trees over an alphabet of actions, such that words (finite sequences) of actions represent states. Let \((\Sigma, D)\) be a concurrent alphabet, where \(D\) is a reflexive and symmetric dependency relation on \(\Sigma\). \((\Sigma^*, \circ, \epsilon)\) denotes the free monoid over \(\Sigma\). Let \(\sim_D\) denote the least congruence on \(\Sigma^*\) such that \(ab \sim_D ba\) if \(-, aDb\). The equivalence classes are called Mazurkiewicz traces.

We assume a set of propositions \(P\). An interpreted \((\Sigma, D, P)\)-tree is a pair \((T, I)\), where \(T\) is a prefix closed subset of \(\Sigma^*\) and \(I : T \to 2^P\) is an interpretation of the vertices of the tree, such that: for \(v, w \in T\) and \(v \sim_D w\) we have \(I(v) = I(w)\). The intuition behind this is to consider the equivalence classes (Mazurkiewicz traces) as states and a sequence of actions leads to its equivalence class.

For any proposition \(P\) let \(\text{vis}(P) = \{a \in \Sigma \mid \exists \exists w \in \Sigma^* \text{ with } wa \in T \text{ and } P \in I(w) \text{ iff } P \notin I(wa)\}\) denote the set of actions visible for \(P\), i.e., that may change its interpretation. Let \(\text{vis}(P) = \bigcup_{P \in P} \text{vis}(P)\). We require that \(\text{vis}(P) \times \text{vis}(P) \subseteq D\), i.e., each proposition can only be modified by dependent actions.

A full \((\Sigma, D, P)\)-tree additionally satisfies:

- For all \(v \in T\) and \(w \sim_D v\) also \(w \in T\).
- For \(w \in \Sigma^*\), \(a, b \in \Sigma\) with \(-aDb\) if \(wa, wb \in T\), then also \(wab \in T\), i.e., actions can only be disabled by dependent actions.

A path through a tree \(T\) (beginning at \(v\)) is a maximal (possibly infinite) word \(w\) such that \((v\) is a prefix of \(w\) and) \(w' \in T\) for every finite prefix \(w' \) of \(w\).
The behaviour of a concurrent system is to be represented by full trees. However, full trees are redundant and can be reconstructed from reduced trees by completing them. Such reductions are useful for model checking if the investigated properties of the full system are the same for the reduced ones. In the following sections we investigate two related frameworks for LTL and CTL* properties.

2.2 Systems of sequential agents

A different approach to Mazurkiewicz traces takes the look of a parallel system of sequential processes, which communicate via joint actions [Thi94]. The underlying assumption is a fixed number \( K \) of locations, each with its own set of actions. In this setting actions are “dependent” if they belong to the same location. Propositions may belong to several locations, and we define

\[
(\Sigma, D) \equiv \bigcup_{i=1}^{K} \Sigma_i
\]

where \( \Sigma_i \) denotes the set of actions and \( D_i \) denotes the set of propositions affiliated with each location \( i \). The sets \( \Sigma_i \) may overlap and we define \( \Sigma = \bigcup_{i=1}^{K} \Sigma_i \) as the set of all actions. Likewise the sets \( D_i \) may overlap, i.e., a proposition may be affiliated with several locations, and we define \( D = \bigcup_{i=1}^{K} D_i \). Then a dependency relation \( D \) can be defined on \( \Sigma \) such that \((a, b) \in D\) iff there exists a location \( i \) such that \( a, b \in \Sigma_i \).

2.3 Interpreted partially ordered runs

For a partial order \((E, \leq)\) and a set \( M \subseteq E \), let \( \downarrow M = \{ e \in E; \exists e' \in M. e \leq e' \} \), the “downward closure” of \( M \). In particular we write \( \downarrow \{e\} \) for \( \downarrow \{e\} \).

A po run over \( \Sigma \) is a labelled poset \( F = (E, \leq, \lambda) \), where \( E \) is a set of “events”, with \((E, \leq)\) a partial order, and \( \lambda : \lambda : E \rightarrow 1 \lambda \) a labelling, such that

- \( E \) is (at most) countable, \( \downarrow \{e\} \) is always finite;
- \( \lambda : \lambda : E \rightarrow 1 \lambda \) is a partial order, and \( \lambda : \lambda : E \rightarrow 1 \lambda \) a labelling, such that
- \( \lambda : \lambda : E \rightarrow 1 \lambda \) is total, (i.e., the events of one agent are causally ordered),

and the global order \( \leq \) is induced by the local total orders, i.e., \( \leq = \left( \bigcup_{i \in \text{locations}} \downarrow \{ \lambda (e) \} \right)^+ \).

A configuration \( c \) of a po run \( F \) is a finite, downward closed set of events (i.e., \( \downarrow c = c \)). Let \( C_F \) denote the set of all configurations of the po run \( F \). Further we define a “localization” of configurations, the “i-view”:\n
\[
\downarrow_i c := \{ c \cap E_i \} \quad \text{i.e. the least configuration that coincides with } c \text{ w.r.t. } i\text{-events.
}
\]

An interpretation of a po run is a mapping \( I : C_F \rightarrow 2^P \) such that if \( \downarrow_i c = \downarrow_i c' \), then \( I(c) \cap P_i = I(c') \cap P_i \), i.e., the interpretation of propositions affiliated with location \( i \) depends only on \( i\)-events. Moreover this implies that a proposition affiliated with several locations may only be changed by joint actions of these locations.

Two interpreted po runs \((F_1, I_1)\) and \((F_2, I_2)\) over \( \Sigma \) are isomorphic, iff there exists an order isomorphism \( h : E_1 \rightarrow E_2 \) between \( F_1 \) and \( F_2 \) that respects the labelling and induces a bijection \( h' : C_{F_1} \rightarrow C_{F_2} \) such that for each \( e \in C_{F_1} \) we have \( I_1(e) = I_2(h'(e)) \).

3 From sequential to partially ordered runs

In this section we show how to define from a dependency relation \( D \) respected by an interpreted tree \((T, I)\) over an alphabet \( \Sigma \), a distributed alphabet and a partition of the propositions so that we can give the tree a semantics in form of interpreted po runs.

A clique w.r.t. \( D \) is a set \( A \subseteq \Sigma \) of actions, which are pairwise dependent, (i.e., \( A \times A \subseteq D \)). A clique \( A \) is maximal if it cannot be extended, i.e., iff for any \( a \in \Sigma \setminus A \) there exists a \( b \in A \) with

\[\text{considering } (\Sigma, D) \text{ as an undirected graph}\]
Let \( \{1, \ldots, K\} \) be the set of all maximal cliques (without repetition). Then we define \( \text{Loc} := \{1, \ldots, K\} \) and \( \tilde{\Sigma} := (\Sigma_1, \ldots, \Sigma_K) \). As required we obtain \( \Sigma = \bigcup_{i=1}^{K} \Sigma_i \), because every letter belongs to at least one maximal clique. Moreover \( aD b \) iff \( \{a, b\} \) is a clique, and the latter holds, if \( \{a, b\} \) is contained in a maximal clique, i.e., iff there exists an \( i \in \text{Loc} \) with \( a, b \in \Sigma_i \).

Next, we have to associate the propositions with fixed locations. Since the set of actions \( \text{vis}(P) \), which may change the value of some proposition \( P \) is pairwise dependent (i.e., again: a clique) it must be contained in some maximal clique \( E_i \). So we put \( P \) into every set \( P_i \) such that \( \text{vis}(P) \not\subseteq E_i \).

So we can define \( \tilde{\Sigma} \) and \( \tilde{\mathcal{F}} \) to meet the requirements for systems of sequential agents. Next we want to show how to map paths of \( T \) to po runs with a corresponding interpretation.

First let \( a_1 a_2 \ldots \) be a path of \( T \). As a set of events \( E \) we take the set of indices \( n \) (which may be finite - in case of a finite path - or infinite, i.e., the set of all natural numbers). As a labelling we take \( \ldots(a_n) := a_n \) and as partial ordering on \( E \) we take \( n \sim m \) iff there exists a subsequence \( n = n_1 \leq n_2 \ldots \leq n_l = m \) such that \( a_n \sim \ldots a_m \). It is easy to check that \( \mathcal{F} = (E, \preceq, \lambda) \) is a po run over \( \tilde{\Sigma} \).

Next we have to give an interpretation \( I' \) of each configuration of \( F \). Thus, let \( c \subseteq E \) be a configuration and let \( P \in P_i \) be a proposition.

For the special case of \( c = \emptyset \) or \( c \) contains no event which could modify \( P \), let \( P \in I'(c) \) iff \( P \in I(c) \). Otherwise let \( n \) be the biggest index (in the index ordering) of \( c \), such that \( a_n \) can modify \( P \). Then we define \( P \in I'(c) \) iff \( P \in I(a_1 \ldots a_n) \).

The interpretation is easily seen to meet the requirements formulated in the last section. Observe also that \( \sim_D \)-equivalent paths are mapped to isomorphic interpreted po runs.

### 4 Linear framework

In this section we define the interleaving based logic LTL and the trace based logic TrPTL.

#### 4.1 Linear Time Temporal Logic

Linear time temporal logic is a formalism to reason about sequences and hence also paths of a tree. Let \( \mathcal{F} \subseteq \mathcal{LTL} \); \( \mathcal{F} \) if \( \phi \in \mathcal{LTL} \), then also \( \neg \phi, X \phi \in \mathcal{LTL} \); \( \mathcal{F} \) if \( \phi, \psi \in \mathcal{LTL} \), then also \( \phi \lor \psi, \phi U \psi \in \mathcal{LTL} \).

Formulas are given a semantics w.r.t. a \((\Sigma, D, P)\)-tree \((T, I)\) as follows: let \( \rho = a_1 a_2 \ldots \) be a maximal path through \( T \) and let \( n \) be an index on \( \rho \). We will omit \( T \) and \( I \), when it is clear from the context.

- \( \rho_n \models P \) iff \( P \in I(a_1 \ldots a_n) \);
- \( \rho_n \models \neg \phi \) iff \( \rho_n \not\models \phi \);
- \( \rho_n \models \phi \lor \psi \) iff \( \rho_n \models \phi \) or \( \rho_n \models \psi \);
- \( \rho_n \models X \phi \) iff \( \rho_{n+1} \models \phi \);
- \( \rho_n \models \phi U \psi \) iff there exists an \( m \geq n \) such that \( \rho_m \models \psi \) and for all \( n \leq k < m \) we have \( \rho_k \models \phi \);
- \( \rho \models \phi \) iff \( \rho_0 \models \phi \), i.e., we evaluate sequential runs from the initial state.
- \((T, I) \models \phi \) iff \( \rho \models \phi \) for all maximal paths \( \rho \) of \( T \), i.e., the tree satisfies a property \( \phi \) if all its paths do.

An important part of the contribution of [Pei93] is to identify a subset of LTL, which is equivalence robust w.r.t. \( D \), i.e., \( \phi \) is equivalence robust if for each two equivalent sequential runs \( \rho_1 \sim_D \rho_2 \); \( \rho_1 \models \phi \) iff \( \rho_2 \models \phi \).
The idea of model checking using representatives is to verify formulas over a tree \( T' \subseteq T \), which contains for every sequential run \( \rho \) of \( T \) at least one equivalent sequential run \( \rho' \). Then, for equivalence robust properties \( \phi \) it holds that \((T, I) \models \phi \Longleftrightarrow (T', I') \models \phi \) \((I' = I | T')\), so that model checking can be performed on the hopefully significantly smaller transition system generating \( T' \).

A formula \( \phi \) is next free if it does not contain the operator \( X \).

The set of actions \( \text{vis}(\phi) \), which are visible for a formula \( \phi \), is defined as the union of the sets \( \text{vis}(P) \), such that \( P \) occurs in \( \phi \).

**Definition 1.** The set of syntactically equivalence robust LTL formulas is the least set of next free LTL formulas \( R_D \), such that
- If \( \text{vis}(\phi) \times \text{vis}(\phi) \subseteq D \), then \( \phi \in R_D \);
- \( R_D \) is closed under boolean combinations.

**Proposition 2 [Pel93].** Syntactically equivalence robust LTL formulas are indeed semantically equivalence robust.

So this proposition gives a sufficient syntactic condition on formulas, for which model checking using representatives is possible. Later we will give a translation of this syntactic subclass into TrPTL, which will also yield an alternative proof of the result.

### 4.2 Trace Based Temporal Logic

We assume \( \text{Loc}, \hat{S} \) and \( \hat{P} \) as above. The idea of the generalization from LTL to TrPTL is to regard the total order of events as “local time” – in contrast to the “global time”, which the total ordering of events in interleaving semantics induces. Syntactically the generalization is very simple: each temporal operator additionally has an index \( i \in \text{Loc} \).

Let \( \text{TrPTL} \) denote the least set (of formulas) such that
- \( \mathcal{P} \subseteq \text{TrPTL} \);
- if \( \phi \in \text{TrPTL} \) and \( i \in \text{Loc} \), then also \( \neg \phi, X_i \phi \in \text{TrPTL} \);
- if \( \phi, \psi \in \text{TrPTL} \) and \( i \in \text{Loc} \), then also \( \phi \lor \psi, \phi U_i \psi \in \text{TrPTL} \).

First, we define the semantics for po runs, then for trees. While the semantics of LTL is defined in terms of indices of a sequential run, the semantics of TrPTL is defined in terms of configurations of a po run. Let \( F \) be a po run, \( I \) an interpretation and \( c \) a configuration in \( F \). We will omit \( F \) and \( I \), when it is clear from the context.

- \( c \models P \) iff \( c \in I(P) \);
- \( c \models \neg \phi \) iff \( c \not\models \phi \);
- \( c \models X_i \phi \) iff for the least \( c' \) such that \( |^i c \subseteq |^i c' \) (which has exactly one more \( i \)-event than \( c \))
  \( |^i c' \models \phi \) holds;
- \( c \models \phi U_i \psi \) iff there exists a chain of configurations \( |^i c \subseteq \cdots \subseteq |^i c_n \) such that
  - \( |^i c_1 = c_0 \), i.e., all configurations in the chain are \( i \)-views;
  - for \( c_i \subseteq |^i c' \subseteq c_{i+1} \) either \( c_i = |^i c' \) or \( |^i c' = c_{i+1} \), i.e., the chain has no “i-holes”;
  - \( c_n \models \psi \) and for all \( 1 \leq i < n \) we have \( c_i \models \phi \).
- \( (F, I) \models \phi \) iff \( \emptyset \models \phi \), where \( \emptyset \) denotes the empty configuration.

Now we extend the interpretation of TrPTL formulas to trees \((T, I)\) and their paths via the translation of sequential to po runs:

- \( \rho \models \phi \) iff the po run \( F \) with the interpretation \( I \) corresponding to the sequential run \( \rho \) satisfies \((F, I) \models \phi \);
- \((T, I) \models \phi \) iff for every sequential run \( \rho \) of \((T, I)\) we have \( \rho \models \phi \).

Note that we immediately obtain by this definition of the semantics on interleaved runs, every TrPTL-formula is equivalence robust, because equivalent runs are mapped on isomorphic po runs, and isomorphism clearly preserves the truth of formulas.
5 Embedding dependency robust LTL formulas in TrPTL

In this section we give a translation of LTL formulas \( \phi \) satisfying Peled's sufficient criterion for equivalence robustness to TrPTL-formulas satisfied by the same sequential runs \( \rho \). This is also an indirect proof that the criterion is indeed sufficient!

We have to distinguish two cases:

- \( \text{vis}(\phi) \times \text{vis}(\phi) \subseteq D \). Then, \( \text{vis}(\phi) \) must be contained in a maximal clique of \((\Sigma, D)\), and hence by our construction there must exist a location \( i \), such that \( \text{vis}(\phi) \subseteq \Sigma_i \). Moreover this means that all propositions occurring in \( \phi \) belong to \( \mathcal{P}_i \). Then we translate \( \phi \) to TrPTL by attaching this index \( i \) to every occurrence of the \( U \)-operator.

- \( \phi \) is a boolean combination. Then we translate the components of the combination first and combine them in exactly the same way.

**Theorem 3.** Let \( \rho \) be a sequential run, \( \phi \) be a syntactically equivalence-robust LTL formula, \( \phi' \) the translation of \( \phi \) to TrPTL. Then, \( \rho \models \phi \) if \( \rho \models \phi' \)

**Proof.** Follows from the following lemmas.

We only treat the case of \( \text{vis}(\phi) \subseteq \Sigma_i \), the generalization to boolean combinations is trivial. Let \( \rho = a_1 a_2 \ldots \) be a sequential run, and \( F \) be the po run constructed from \( \rho \) with the corresponding interpretation. First we need the following lemma:

**Lemma 4.** For next free LTL formulas \( \psi \) with \( \text{vis}(\psi) \subseteq \Sigma_i \) and for an index \( n \) in a run \( \rho = a_1 a_2 \ldots \) such that \( a_{n+1} \notin \Sigma_i \) we have: \( \rho_n \models \psi \) iff \( \rho_{n+1} \models \psi \).

**Proof.** Follows an easy induction on the structure of \( \psi \).

Then we need a corresponding lemma on the TrPTL side:

**Lemma 5.** Let \( \psi' \) be a TrPTL formula, such that all temporal operators are indexed by \( i \) and all propositions belong to \( \mathcal{P}_i \). Then, for any configuration \( c \) we have \( c \models \psi' \) iff \( \{t \} c \models \psi \)

**Proof.** This property can also be shown by an easy induction on the structure of \( \psi' \).

Now we can bring configurations and the indices of runs together:

**Lemma 6.** Let \( c_n \) denote the configuration containing exactly the events \( 1 \) through \( n \). Then, \( \rho_n \models \psi \) iff \( c_n \models \psi' \), where \( \psi \) is a sub-formula of \( \phi \), and \( \psi' \) is the translation of \( \psi \) to TrPTL.

**Proof.** We proceed by induction on the complexity of \( \psi \). The only interesting case is \( \psi = \psi_1 U \psi_2 \). Then \( \psi' = \psi'_1 U \psi'_2 \). The idea we use is to translate sequences of indices of a sequential run to sequences of \( i \)-view configurations and vice versa.

Let's assume \( \rho_n \models \psi_1 U \psi_2 \). Then there exists an \( m \geq n \) such that \( \rho_m \models \psi_2 \) and for all \( n \leq k < m \) we have \( \rho_k \models \psi_1 \). By induction also \( c_m \models \psi'_2 \) and for all \( n \leq k < m \) we have \( c_k \models \psi'_1 \). From the chain \( c_n \subseteq \ldots \subseteq c_m \) we obtain the chain \( l^1 c_n \subseteq \ldots \subseteq l^1 c_m \), for which also holds \( l^1 c_n \models \psi'_2 \) and for all \( n \leq k < m \) we have \( l^1 c_k \models \psi'_1 \). By throwing out duplicates we obtain the necessary sequence of \( i \)-views and finally know that \( c_n \models \psi'_1 \cup \psi'_2 \).

For the converse direction let \( c_n \models \psi'_1 \cup \psi'_2 \). Then there exists a chain of \( i \)-views \( l^1 c_n \subseteq \ldots \subseteq l^1 d_u \) such that again there are no intermediate \( i \)-views between the members of this chain, \( l^1 d_u \models \psi'_2 \) and for all \( 1 \leq u < v \) we have \( l^1 d_u \models \psi'_1 \). For each \( d_u \) in this sequence let \( k_u \) denote the index in \( \rho \) of the maximal \( i \)-event in \( d_u \). Then, since \( l^1 d_u = l^1 c_k \), we obtain \( \rho_k \models \psi'_2 \) and for all \( 1 \leq u < v \) we get \( \rho_{k_u} \models \psi_1 \). Now, the actions with indices between \( k_u \) and \( k_{u+1} \) are not from \( \Sigma_i \) and hence at these indices the same properties hold as for \( k_u \). In particular, \( \rho_k \models \psi_1 \), for \( k \): \( k_1 \leq k < k_2 \). So, finally \( \rho_n \models \psi_1 U \psi_2 \).
As a consequence of the theorem we obtain a syntactic characterization of equivalence robust LTL properties in terms of TrPTL: A TrPTL-formula $\phi$ is local iff it is a boolean combination of formulas $\psi$ for which there exists an $i$ such that all temporal operators in $\psi$ are indexed with $i$ and all propositions occurring in $\psi$ belong to $P_i$. In short: local formulas do not change the point of view. Thiagarajan calls the set of local formulas the "product subset" of TrPTL, because local formulas can be translated to product Büchi automata accepting the same language [Thi95]. The syntactically equivalence robust LTL formulas correspond exactly to the local, next free TrPTL formulas, in particular:

**Theorem 7.** Every local, next free TrPTL formula can be translated to an equivalent (syntactically equivalence robust) LTL formula by omitting the location indices.

**Proof.** Follows exactly the same lines as the converse direction shown above.

6 Branching framework

In this section we define Computation Tree Logic (CTL).

6.1 Computation Tree Logic

Computation Tree Logic is a formalism to reason about paths of a tree including its branching. Let CTL denote the least set (of formulas) such that

- $P \subseteq CT$;
- if $\phi \in CT$, then also $\neg \phi, EX \phi, EG \phi \in CT$;
- if $\phi, \psi \in CT$, then also $\phi \lor \psi, E\psi U \psi \in CT$.

Formulas are given a semantics w.r.t. a $(\Sigma, D, P)$-tree $(T, I)$. Let $w \in T$. We will omit $T$ and $I$, when it is clear from the context.

- $w \models P$ iff $P \in I(w)$;
- $w \models \neg \phi$ iff $w \not\models \phi$;
- $w \models \phi \lor \psi$ iff $w \models \phi$ or $w \models \psi$;
- $w \models EX \phi$ iff $wa \models \phi$, for some $wa \in T$;
- $w \models EG \phi$ iff there exists a maximal path $a_1 a_2 ...$ starting at $w$ and for all $k \geq 0$ we have $wa_1 ... a_k \models \phi$;
- $w \models E\phi U \psi$ iff there exists a maximal path $a_1 a_2 ...$ starting at $w$ and an $m \geq 0$ such that $wa_1 ... a_m \models \psi$ and for all $0 \leq k < m$ we have $wa_1 ... a_k \models \phi$;
- $(T, I) \models \phi$ iff $I \models \phi$, i.e., the tree satisfies a property $\phi$ if its root does.

We will also use formulas of the form $AG\phi$ as abbreviations for $\neg Etrue U \neg \phi$, and the standard logical connectives $\land, \lor$. CTL* is an extension of CTL where the modalities $E, U$ and $G$ can be nested without restrictions. CTL(*)-X denotes the restriction of CTL(*) without the next step operator $EX$.

In [GKPP95] the partial order approach to model checking is extended to a fragment of CTL-X along similar lines as for equivalence robust LTL-X formulas. This raises the question, whether also this fragment of CTL can be considered as or translated to a partial order logic.

It turns out that things are more complicated for branching time, which is also reflected by more restrictions on the reductions: the equivalence has to take the branching into account. The reason is that CTL-X can speak not only about the order of local (property changing) events along a path through the system (reflected by the total ordering of events of one component in the location based approach), but also about the order of branchings, i.e., events in other components that take decisions.
Consider the following two full trees, here drawn with double circled root, equivalent words leading to the same node, propositions written at nodes, where they are true, and \{b, e\} is the set of visible actions with bDe

For the left hand tree the CTL-X formula \[ E((AG\neg P) \rightarrow (AG\neg S)) U Q \] says there is a path to Q on which the decision against S is taken before the decision against P, although these (invisible) decisions (manifested in the execution of a and d) are independent. Thus, the \(\sim_D\)-equivalent paths ade and dae can be distinguished.

For the right hand tree the formula \[ E(P \rightarrow AG\neg S) U R \] says, that there is a path to R on which the decision against S is taken before P is set, although the decision (by invisible action c) is independent of the setting of P (by the visible action b). Thus, the \(\sim_D\)-equivalent paths cbe and bee can be distinguished.

Both formulas satisfy the conditions of syntactically equivalence robust formulas for LTL (i.e., they are next free and \(vis(\phi) \times vis(\phi) \subseteq D\)), but they are not semantically equivalence robust w.r.t. \(\sim_D\). CTL-X can distinguish certain \(\sim_D\)-equivalent paths, which therefore cannot be eliminated by property preserving reductions. Hence we need a finer equivalence than \(\sim_D\).

Therefore, we define a strengthening of Mazurkiewicz trace equivalence for trees, which takes the necessary dependence of decisions among each other and with visible actions into account. This will be done in the traditional lines rather than the location based presentation.

### 6.2 Conflict and visibility respecting trace systems

As mentioned above, we have to take visibility into the considerations for equivalence, hence we add it as a parameter to the concurrent alphabet:

Let \((\Sigma, D, V)\) be a concurrent alphabet with visibility, where \((\Sigma, D)\) is defined as above and \(V \subseteq \Sigma\) is a distinguished set of visible actions.

Let \(\Sigma^* = \{(w, d); w \in \Sigma^* \text{ and } d \subseteq \{v|v|\}\}\), the set of words over \(\Sigma\) with marked positions. The set \(d\) in a word is supposed to represent the positions in \(w\), where a decision has been taken, i.e., a conflict has been resolved. On \(\Sigma^*\) we define concatenation by \((v, d) \circ (w, d') = (vw, d \cup \{v|v| + m; m \in d')\)\).

On \(\Sigma^* = (\Sigma^*, \circ, (e, \emptyset))\), which obviously is a free monoid, we define \(\sim_{D,V}\) to be the least congruence such that \((ab, \emptyset) \sim (ba, \emptyset)\) for \(\sim a D b\) and \((ab, \{1\}) \sim_{D,V} (ba, \{2\})\) for \(\sim a D b\) and \(b \notin V\).

The intuition behind this definition is that an action resolving a conflict can only be permuted with invisible actions which do not resolve a conflict.

The equivalence classes are called conflict marked traces, \(\Sigma^*/\sim_{D,V}\) is the set of conflict marked traces over \((\Sigma, D, V)\). Concatenation and prefix relation \(\leq\) are defined on traces in the obvious way, traces are denoted by \(\tau\).

Let \(M\) be a nonempty, prefix closed set of conflict marked traces and \(I : M \rightarrow 2^P\) an interpretation. \(M\) is called a conflict and visibility respecting trace system (CVTS), iff

- \(M\) is consistently marked, i.e., for any \([w, d], [w, d'] \in M\) we have \(d = d'\).
- $M$ is proper, i.e., for any $\tau_1, \tau_2 \in M$, such that there exists a $\tau_1, \tau_2 \leq \tau_3 \in \Sigma^*/\sim_{D,\nu}$, then there also exists a $\tau_1, \tau_2 \leq \tau_3 \in M$.
- $M$ is conflict consistent, i.e., for any $(w,d) \in M$ the last position is marked (i.e., $|w| + 1 \in d$) iff there exists a $a = a_1 \cdots a_n b \in \Sigma^*$ with $\neg a_1 D a$ and $b D a$ and $b \neq a$, such that $(wu, d') \in M$ for some $d'$, i.e., $a$ really resolves a conflict.
- $I$ is $D$-respecting, i.e., $I([w, d]) = I([w, d'])$ for $v \sim_D w$.
- $I$ is visibility consistent, i.e., for $I([w, d]) \neq I([w, d'])$ we have $a \in V$.
- A run of a CVTS $(M, I)$ is an infinite (or finite and maximal) directed, and prefix closed subset of $M$.

Note that in the case of a CVTS consisting of a single run (the linear case) no trace contains any marked conflicts, and hence the equivalence coincides with Mazurkiewicz's trace equivalence. Also note that we can define dependence graphs over conflict labelled traces just as for the simple case, when we take the dependence of marked letters (and visible actions) into account. This means, CVTS still represent a partial order semantics, albeit with more order than induced by $D$ alone.

### 6.3 Equivalence on representative trees

In a sense explained below, there is a correspondence between CVTS and full trees. What we want to develop now is a notion of a partial tree representing some CVTS. This means in particular that the nodes of the tree should have a branching future corresponding to that of the corresponding trace in the CVTS up to equivalence. This is not the case for arbitrary trees, hence the concept of representative trees.

Let $(T, I)$ be an interpreted $(\Sigma, D, P)$-tree. For each $w \in T$ we inductively define a marking $d_w$ by $d_e = \emptyset$ and $d_{wa} = d_w \cup \{w| + 1\}$ if there exists a $a$ as in the definition of conflict consistent above with $w \in T$, and $d_{wa} = d_w$ otherwise.

An interpreted tree $(T, I)$ is called $\sim_{D,\nu}$-representative if for each $v, w \in T$ such that $[v, d_v], [w, d_w] \in [u, d]$ for some $u$ and $d$ (i.e., $v$ and $w$ do not contradict w.r.t. the equivalence), there exists an extension $u \in T$ such that $[w, d_w] \leq [u, d_u]$ (and vice versa). Intuitively this means that the paths beginning at some node $v$ of the tree cover all the behaviour in $T$ that is not excluded by $v$ w.r.t. the equivalence. Technically this property is important for mapping (representative) trees to CVTS via equivalence classes.

Note that full $(\Sigma, D, P)$-trees are trivially $\sim_{D,\nu}$-representative.

**Definition 8.** Let $\text{cuts}(T) = \{\tau \in \Sigma^*/\sim_{D,\nu} | \tau \leq [w, d_w] \text{ for some } w \in T\}$ be the mapping of a representative tree to the CVTS. $\text{cuts}$ induces an equivalence relation on representative trees (also denoted by $\sim_{D,\nu}$), which is a counterpart to Mazurkiewicz equivalence on sequences.

### 6.4 Connection to stuttering equivalence

Our first aim is to show that $\sim_{D,\nu}$ (on representative trees) is finer than stuttering equivalence:

**Definition 9.** A stuttering relation for a set of propositions $Q \subseteq P$ between two trees $T_1$ and $T_2$ is a relation $\sim \subseteq T_1 \times T_2$, such that

1. $\epsilon \sim \epsilon$
2. if $w_1 \sim w_2$, then for each $P \in Q$ we have $P \in I(w_1)$ iff $P \in I(w_2)$, i.e., the same propositions hold at the states reached via $w_1$ and $w_2$;
3. if $w_1 \sim w_2$, then for each path $\pi_1$ in $T_1$ beginning at $w_1$ there exists a path $\pi_2$ in $T_2$ beginning at $w_2$, a partition $B_1B_2\ldots$ of $\pi_1$ and a partition $B'_1B'_2\ldots$ of $\pi_2$, such that $B_i \times B'_i \subseteq \sim$ for all $i$;
4. the analogue condition to (3) when exchanging $w_1, T_1$ with $w_2, T_2$ holds.

$T_1$ and $T_2$ are stuttering equivalent w.r.t. $Q$ (denoted by $T_1 \sim_Q T_2$), iff there exists a stuttering equivalence for $Q$ between them.
The use of stuttering in the context of program logics is given by the following theorem:

**Theorem 10 [BCG88].** Let $\phi$ be a $\text{CTL}^*\text{-X}$ formula containing only propositions in $Q$, $T_1 \sim_Q T_2$ with $w_1 \sim_Q w_2$ stuttering equivalent nodes, then $T_1, w_1 \models \phi$ if and only if $T_2, w_2 \models \phi$.

This means, that a reduction of a (full) tree $T_f$ to a tree $T_r$, that guarantees $T_f \sim_Q T_r$, allows to model check $\text{CTL}^*\text{-X}$ properties of $T_f$ on $T_r$, as done in [GKPP95].

The following theorem explains the method of [GKPP95] in terms of $\sim_D, V$: the latter equivalence is finer than stuttering equivalence.

**Theorem 11.** If $T_1 \sim_D V, T_2$, $\text{vis}(Q) \subseteq V$, $\text{vis}(Q) \times \text{vis}(Q) \subseteq D$, then also $T_1 \sim_Q T_2$.

**Proof.** We have to construct a stuttering equivalence between the nodes of $T_1$ and $T_2$. The idea is to reduce words to significant letters, i.e., visible actions (w.r.t. $Q$) and decision marked actions: if two words contain the same significant letters in the same order, they should be stuttering equivalent for $Q$.

For this we inductively define a projection $\text{proj}$ on decorated words: $\text{proj}(\epsilon, \emptyset) = \epsilon$ and $\text{proj}(v, \{d\}) = \text{proj}(v, \{d\} \setminus \{v\} + 1) v_\epsilon \epsilon$, where $\epsilon = a$ if $a \in \text{vis}(Q)$ or $|v| + 1 \in d$, and $\epsilon = \epsilon$ otherwise. It is easy to see that $(v_1, d_1) \sim_D v_2 (v_2, d_2)$ implies $\text{proj}(v_1, d_1) = \text{proj}(v_2, d_2)$, i.e., the induced equivalence is weaker than $\sim_D, V$ (where it is important that the elements of $\text{vis}(Q)$ are pairwise dependent).

Now for $v_1 \in T_1$ and $v_2 \in T_2$ define $w_1 \sim w_2$ iff $\text{proj}(w_1, d_{w_1}) = \text{proj}(w_2, d_{w_2})$. We claim that $\sim$ is a stuttering relation.

For seeing that the interpretations of $w_1$ and $w_2$ coincide as required by (2), let $\tau = [v, d]$ be a maximal trace, such that $\tau \subseteq [v, d_{w_1}], [v, d_{w_2}]$. Then $[v, d_{w_1}] = [v, d][v, d_1]$ and $[v, d_{w_2}] = [v, d][v, d_2]$. It is easy to see that $u_1 \in (\Sigma \setminus \text{vis}(Q))^*$ and $v_1 = \emptyset$. Otherwise there would be a first significant letter $a$ in (e.g.) $u_1$ which must also occur as first significant letter in $v_2$ and which could by $\sim_D, V$ be shifted to the front and over to $v$, such that some $[v_2, d'] \subseteq [v_1, d_{w_1}], [v_1, d_{w_2}]$, contradicting the maximality of $[v, d]$. Now $\sim_D, V$ and extension by invisible actions preserve the validity of propositions at words, hence we obtained a chain from $w_1$ via $v_1$ via $v$ via $w_2$ to $w_2$, all with the same interpretation.

Finally we have to care about paths (conditions 3 and 4). Let $w_1 \sim w_2$ and let $\pi_1$ be a path in $T_1$ beginning at $w_1$. We assume that $\pi_1$ is represented by a (possibly infinite) word over $\Sigma$, such that $w_1$ is a prefix of $\pi_1$ and every finite prefix of $\pi_1$ belongs to $T_1$. Therefore in particular for every finite prefix $w$ we have $[w, d_w] \subseteq \text{cvt}(T_1) = \text{cvt}(T_2)$. This also means that we find a word $w'$ in $T_2$, such that $[w, d_w] \subseteq [w', d_{w'}]$. Moreover $w'$ can be chosen such that $[w, d_w] \subseteq [w', d_{w'}]$ as prefix by an argument similar to the proof of (2) above. Finally since $T_2$ is representative we can find such a $w'$ which actually has $w_2$ as prefix, i.e., which lies on a path beginning at $w_2$.

By König's lemma we can fix a path $\pi_2$ in $T_2$, such that for every finite prefix $w$ of $\pi_1$ there exists a $w'$ in $\pi_2$ such that $[w, d_w] \subseteq [w', d_{w'}]$. Now it is immediate to partition the paths into word fragments of the form $au$, where $a$ is a significant letter and $u$ consists only of insignificant letters (according to visibility and the marking $d_w$, which can be extended from words to paths). Let $B_1^1, B_1^2, \ldots$ be the resulting sets of prefixes of $\pi_1$ and accordingly $B_2^1, B_2^2, \ldots$ for $\pi_2$. By construction it is immediate, that $B_1^1 \times B_2^1 \subseteq \sim$, moreover if $w_1 \in B_1^1$ we can take any element of $B_2^1$ as $w_2$ and consider the partitions in both paths from the starting points. This proves condition (3), condition (4) follows by symmetry.

Similarly to the linear framework we can define semantic equivalence robustness for $\text{CTL}^*$: $\phi$ is **semantically equivalence robust**, if for each two $\sim_D, V$-representative trees $T_1 \sim_D, V T_2$: $T_1 \models \phi$ if and only if $T_2 \models \phi$.

**Definition 12.** The set of syntactically equivalence robust $\text{CTL}^*$ formulas is the least set of next free $\text{CTL}^*$ formulas $R_D, V$, such that
Let \( \text{vis}(\phi) \subseteq V \) and \( \text{vis}(\phi) \times \text{vis}(\phi) \subseteq D \), then \( \phi \in R_{D,V} \);

- \( R_{D,V} \) is closed under boolean combinations.

**Corollary 13.** Syntactically equivalence robust \( \text{CTL}^* \) formulas are indeed semantically equivalence robust.

### 6.5 Partial order reductions for branching time

In this section we give a representation of the partial order reductions of [GKPP95] on trees. This is done to highlight the connection of these reductions to the equivalence \( \sim_{D,V} \).

[GKPP95] deals with the reduction of transition systems (removal of transitions) in contrast to the reduction of trees given here. The connection naturally lies in the unfolding of the (deterministic) transition systems to trees, such that the elimination of transitions corresponds to a reduction of one or several, possibly even an infinite number of subtrees. In [GKPP95] four conditions for the reduction of transition systems are given, which we translate to conditions for reductions on trees, where this is possible.

For a tree \( T \) with \( w \in T \) let \( S(w,T) = \{ w^i | w \in T \} \) denote the set of successors. Let \((T,I)\) be a full \((E,D,P)\)-tree. A reduction of \( T \) to a tree \( T' \subseteq T \) is admissible if

- **C1** For \( wb \in S(w,T) \neq S(w,T) \) there is no \( w_{a_1} \cdots a_nb \in T \), such that \( b' D b \) and \( b' \neq b \) and \( \neg a_i D b \) for all \( a_i \), i.e., on any path in \( T \) beginning at \( w \) the first action dependent on \( b \) to be taken is \( b \) itself.

- **C2** If \( wa \in S(w,T) \neq S(w,T) \), then \( a \notin V \).

- **C3'** If \( w \in T \) and \( wb \in T \), then \( wub \in T' \) such that for every prefix \( wuc \) of \( wu \) \( S(wuc,T) \neq S(wuc',T) \), i.e., \( b \) is reachable by a path with reductions in every step.

C1 and C2 correspond directly to the equally named conditions in [GKPP95]. The condition C3 there is technically too strongly connected to the reduction algorithm on transition systems to be represented on trees, however it implies condition C3' (c.f. Lemma A.8 of [GKPP95], which is what we actually needed). Then, [GKPP95] additionally have a condition C0, which states that in case of a reduction at some node only one successor should remain, but which is not necessary for the correctness of reductions. Therefore we omit it here without loss of generality.

**Proposition 14.** Let \( T \) be a full \((E,D,P)\)-tree, \( T' \) be a reduced tree according to C1, C2, and C3'. Then, \( T' \) is \( \sim_{D,V} \)-representative and \( T \sim_{D,V} T' \).

**Proof.** We assume the canonical marking \( d \) from the full tree. For a word \( wv \) such that \( v \in T' \) and \( wv \in T \) it is possible to show by induction on the length of \( w \) that there exists a \( v' \in T' \) such that \( [wv,d_{wv}] \leq [v',d_{wv}'] \). The case \( w = \epsilon \) is obvious. Otherwise for \( aw \) let \( u \) be according to C2 such that \( v_{au} \in T' \) such that at all actions along \( u \) the tree has been reduced. Then, by C2 all actions of \( u \) are invisible and by C1 no element in \( u \) resolves a conflict. Then, it is easy to see that \( [v_{au},d_{v_{au}}], [v_{au},d_{v_{au}}] \leq [t',d] \) for some \( t' \) and \( d \) and because \( T' \) is representative we find \( v_{au}' \in T \) with \( [v_{au},d_{v_{au}}] \leq [v_{au}',d_{v_{au}'}] \). Using the fact that \( T \) is full we can choose \( w' \) such that \( [w'] \leq [w] \) and conclude by the induction hypothesis.

Now, this observation can then be applied several times for showing that the marking of \( T \) is conflict consistent for \( T' \) as well, \( T' \) is \( \sim_{D,V} \)-representative and equivalent to \( T \).

### 7 Discussion

We have shown that the (syntactically) "equivalence-robust" subset of LTL is essentially identical to the local, next free part of the partial order logic TrPTL. In fact, in the same way as for TrPTL, we could have shown that the "equivalence-robust" subset of LTL can be translated to a subset of TLC [APP95]. In a work [Thi95] evolving in parallel to this one, Thiagarajan has given a correspondence
between the full product subset of TrPTL and a semantically defined subset of a version of LTL with action labelled modalities instead of next, which enables the encoding of local next modalities. For the branching case we have defined a generalization of Mazurkiewicz's trace equivalence to trees. Then, we showed that under some condition equivalent trees are stuttering equivalent and therefore cannot be distinguished by any CTL*-X formulas. We proved that partial order reductions for CTL*-X give equivalent trees.

Our approach can be also used as a semantic basis for branching time partial order logics for expressing equivalence robust branching time properties. This motivates our thesis that partial order reduction methods are only applicable to partial order logics.

References


This article was processed using the \LaTeX macro package with LLNCS style
<table>
<thead>
<tr>
<th>Date</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>93/01</td>
<td>R. van Geldrop</td>
<td>Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.</td>
</tr>
<tr>
<td>93/02</td>
<td>T. Verhoeff</td>
<td>A continuous version of the Prisoner's Dilemma, p. 17</td>
</tr>
<tr>
<td>93/03</td>
<td>T. Verhoeff</td>
<td>Quicksort for linked lists, p. 8.</td>
</tr>
<tr>
<td>93/04</td>
<td>E.H.L. Aarts, J.H.M. Korst, P.J. Zwietering</td>
<td>Deterministic and randomized local search, p. 78.</td>
</tr>
<tr>
<td>93/05</td>
<td>J.C.M. Baeten, C. Verhoeven</td>
<td>A congruence theorem for structured operational semantics with predicates, p. 18.</td>
</tr>
<tr>
<td>93/06</td>
<td>J.P. Veldkamp</td>
<td>On the unavoidability of metastable behaviour, p. 29</td>
</tr>
<tr>
<td>93/07</td>
<td>P.D. Moerland</td>
<td>Exercises in Multiprogramming, p. 97</td>
</tr>
<tr>
<td>93/08</td>
<td>J. Verhoosel</td>
<td>A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.</td>
</tr>
<tr>
<td>93/10</td>
<td>K.M. van Hee</td>
<td>Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.</td>
</tr>
<tr>
<td>93/16</td>
<td>H. Schepers, J. Hoorman</td>
<td>A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27.</td>
</tr>
<tr>
<td>93/17</td>
<td>D. Alstein, P. van der Stok</td>
<td>Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.</td>
</tr>
<tr>
<td>93/18</td>
<td>C. Verhoeff</td>
<td>A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.</td>
</tr>
<tr>
<td>93/19</td>
<td>G.-J. Houben</td>
<td>The Design of an Online Help Facility for ExSpect, p. 21.</td>
</tr>
<tr>
<td>93/21</td>
<td>M. Codish, D. Dann, G. Filé, M. Bruynooghe</td>
<td>Freeness Analysis for Logic Programs - And Correctness, p. 24.</td>
</tr>
<tr>
<td>93/22</td>
<td>E. Poll</td>
<td>A Typechecker for Bijective Pure Type Systems, p. 28.</td>
</tr>
<tr>
<td>93/23</td>
<td>E. de Kogel</td>
<td>Relational Algebra and Equational Proofs, p. 23.</td>
</tr>
<tr>
<td>93/24</td>
<td>E. Poll, Paula Severi</td>
<td>Pure Type Systems with Definitions, p. 38.</td>
</tr>
<tr>
<td>93/26</td>
<td>W.M.P. van der Aalst</td>
<td>Multi-dimensional Petri nets, p. 25.</td>
</tr>
<tr>
<td>93/27</td>
<td>T. Kloks and D. Kratsch</td>
<td>Finding all minimal separators of a graph, p. 11.</td>
</tr>
<tr>
<td>93/28</td>
<td>F. Kamareddine, R. Nederpelt</td>
<td>A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.</td>
</tr>
<tr>
<td>93/29</td>
<td>R. Post and P. De Bra</td>
<td>GOLD, a Graph Oriented Language for Databases, p. 42.</td>
</tr>
<tr>
<td>93/30</td>
<td>J. Deogun, T. Kloks, D. Kratsch</td>
<td>On Vertex Ranking for Permutation and Other Graphs, p. 11.</td>
</tr>
</tbody>
</table>
H. Müller

93/31 W. Köver
Derivation of delay intensive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.

93/32 H. ten Eikelder and H. van Geldrop

93/33 L. Loyens and J. Moonen
ILLAS, a sequential language for parallel matrix computations, p. 20.

93/34 J.C.M. Baeten and J.A. Bergstra
Real Time Process Algebra with Infinitesimals, p. 39.

93/35 W. Ferrer and P. Severi
Abstract Reduction and Topology, p. 28.

93/36 J.C.M. Baeten and J.A. Bergstra
Non Interleavee Process Algebra, p. 17.

93/37 J. Brunekreef, J-P. Katoen, R. Koymans, S. Mauw
Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73.

93/38 C. Verhoef
A general conservative extension theorem in process algebra, p. 17.

93/39 W.P.M. Nuijten, E.H.L. Aarts, D.A.A. van Eip Taalman Kip, K.M. van Hee
IUAS, a sequential language for parallel matrix computations, p. 20.

93/40 P.D.V. van der Stok, M.M.M.P. Claessen, D. Alstein

93/41 A. Bijlsma
Temporal operators viewed as predicate transformers, p. 11.

93/42 P.M.P. Rambags
Automatic Verification of Regular Protocols in P/T Nets, p. 23.

93/43 B.W. Watson
A taxonomy of finite automata construction algorithms, p. 87.

93/44 B.W. Watson
A taxonomy of finite automata minimization algorithms, p. 23.

93/45 E.J. Luit, J.M.M. Martin
A precise clock synchronization protocol, p.

93/46 T. Kloks, D. Kratsch, J. Spinrad

93/47 W. v.d. Aalst, P. De Bra, G.J. Houben, Y. Komatsky

93/48 R. Gorth
Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.

93/49 R. The object-oriented paradigm, p. 28.

93/50 F. Kamaroddine, R.P. Nederpelt
Canonical typing and Π-conversion, p. 51.

94/03 L.B. Hartman, K.M.O. van Hee

94/04 J.C.M. Baeten, J.A. Bergstra
Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.

94/05 P. Zhou, J. Hooyman

94/06 T. Basten, T. Kanz, J. Black, M. Coffin, D. Taylor
Time and the Order of Abstract Events in Distributed Computations, p. 29.

94/07 K.R. Apt, R. Bol

94/08 O.S. van Rooyen
A Hierarchical Diagrammatic Representation of Class Structure, p. 22.
94/09 J.C.M. Baeten  
J.A. Bergstra  
Process Algebra with Partial Choice, p. 16.

94/10 T. Verhoef  
The testing Paradigm Applied to Network Structure, p. 31.

94/11 J. Pelzka  
C. Huizing  
C. Petersohn  
A Comparison of Ward & Mellor’s Transformation  

94/12 T. Kloks  
D. Kratsch  
H. Müller  

94/13 R. Soljic  
A New Method for Integrity Constraint checking in Deductive Databases, p. 34.

94/14 W. Peremans  
Ups and Downs of Type Theory, p. 9.

94/15 R.J.M. Vaessens  
E.H.L. Aarts  
J.K. Lenstra  
Job Shop Scheduling by Local Search, p. 21.

94/16 R.C. Backhouse  
H. Doornbos  
Mathematical Induction Made Calculational, p. 36.

94/17 S. Mauw  
M.A. Reniers  
An Algebraic Semantics of Basic Message  
Sequence Charts, p. 9.

94/18 F. Kamareddine  
R. Nederpelt  
Refining Reduction in the Lambda Calculus, p. 15.

94/19 B.W. Watson  
The performance of single-keyword and multiple-keyword pattern matching  
algorithms, p. 46.

94/20 R. Bloo  
F. Kamareddine  
R. Nederpelt  
Beyond $\beta$-Reduction in Church’s $\lambda \rightarrow$, p. 22.

94/21 B.W. Watson  
An introduction to the Fire engine: A C++ toolkit for Finite automata and Regular  
Expressions.

94/22 B.W. Watson  
The design and implementation of the FIRE engine:  
A C++ toolkit for Finite automata and regular Expressions.

94/23 S. Mauw and M.A. Reniers  
An algebraic semantics of Message Sequence Charts, p. 43.

94/24 D. Dams  
O. Grumberg  
R. Genh  
Abstract Interpretation of Reactive Systems:  
Abstractions Preserving $\text{VCTL}^*$, $\text{SCTL}^*$ and $\text{CTL}^*$, p. 28.

94/25 T. Kloks  
$K_{13}$-free and $W_4$-free graphs, p. 10.

94/26 R.R. Hoogerwoord  
On the foundations of functional programming: a programmer’s point of view, p. 54.

94/27 S. Mauw and H. Mulder  

94/28 C.W.A.M. van Overveld  
M. Verhoefen  

94/29 J. Hooman  
Correctness of Real Time Systems by Construction, p. 22.

94/30 J.C.M. Baeten  
J.A. Bergstra  
Gh. Stefanescu  
Process Algebra with Feedback, p. 22.

94/31 B.W. Watson  
R.E. Watson  
A Boyer-Moore type algorithm for regular expression  
pattern matching, p. 22.

94/32 J.J. Vereijken  

94/33 T. Laan  
A formalization of the Ramified Type Theory, p.40.

94/34 R. Bloo  
F. Kamareddine  
R. Nederpelt  
The Barendregt Cube with Definitions and Generalised  
Reduction, p. 37.

94/35 J.C.M. Baeten  
S. Mauw  
Delayed choice: an operator for joining Message  
Sequence Charts, p. 15.

94/36 F. Kamareddine  
R. Nederpelt  
Canonical typing and $\Pi$-conversion in the Barendregt  
Cube, p. 19.

94/37 T. Baaten  
R. Bol  
M. Voorhoeve  
Simulating and Analyzing Railway Interlockings in  
ExSpect, p. 30.

94/38 A. BiJlsma  
Point-free substitution, p. 10.
On the equivalence covering number of splitgraphs, p. 4.

Distributed Consensus and Hard Real-Time Systems, p. 34.

Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, p. 6.

Concatenation of Graphs, p. 7.

Category Theory as Coherently Constructive Lattice Theory: An Illustration, p. 35.

Verifying Sequentially Consistent Memory, p. 160

Tutorial voor de ExSect-bibliotheek voor "Administratieve Logistiek", p. 43.

The A-cube with classes of terms modulo conversion, p. 16.

On \( \Pi \)-conversion in Type Theory, p. 12.

Fixed-Point Calculus, p. 11.

Process Algebra with Propositional Signals, p. 25.

A short and flexible proof of Strong Normalization for the Calculus of Constructions, p. 27.

Listing simplicial vertices and recognizing diamond-free graphs, p. 4.

Traces and Logic, p. 81

A Partial Order Approach to Branching Time Logic Model Checking, p. 20.

The Construction of a small Communication Library, p. 16.

Formalizing Process Algebraic Verifications in the Calculus of Constructions, p. 49.

Concrete process algebra, p. 134.

An Isotopic Invariant for Planar Drawings of Connected Planar Graphs, p. 9.

A Type Inference Algorithm for Pure Type Systems, p. 20.

A Quantitative Analysis of Iterated Local Search, p. 23.

Drawing Execution Graphs by Parsing, p. 10.

Preservation of Strong Normalisation for Explicit Substitution, p. 12.

Discrete Time Process Algebra, p. 20.
<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>95/10</td>
<td>R.C. Backhouse</td>
<td>MathJax: A System for On-Line Preparation of Mathematical Documents</td>
<td>p. 15</td>
</tr>
<tr>
<td></td>
<td>R. Verhoeven</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>O. Weber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95/11</td>
<td>R. Seljee</td>
<td>Deductive Database Systems and integrity constraint checking</td>
<td>p. 36</td>
</tr>
<tr>
<td>95/12</td>
<td>S. Mauw and M. Reniers</td>
<td>Empty Interworkings and Refinement Semantics of Interworkings Revised</td>
<td>p. 19</td>
</tr>
<tr>
<td>95/13</td>
<td>B.W. Watson and G. Zwaan</td>
<td>A taxonomy of sublinear multiple keyword pattern matching algorithms</td>
<td>p. 26</td>
</tr>
<tr>
<td>95/14</td>
<td>A. Ponte, C. Verhoef</td>
<td>De proceedings: ACP'95</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S.F.M. Vlijmen (eds.)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>