Models Whose Checks Don't Explode

R. P. Kurshan

AT&T Bell Laboratories, Murray Hill, New Jersey 07974

Abstract. Automata-theoretic verification is based upon the language containment test

$$\mathcal{L}(P_0 \otimes P_1 \otimes \cdots \otimes P_k) \subset \mathcal{L}(T)$$

where the P_i's are automata which together model a system with its fairness constraints, $\otimes$ is a parallel composition for automata and T defines a specification. The complexity of that test typically grows exponentially with k. This growth, often called "state explosion", has been a major impediment to computer-aided verification, and many heuristics which are successful in special cases, have been developed to combat it. While all such heuristics are welcome advances, it often is difficult to quantify benefit in terms of hard upper bounds. This paper gives a general algorithm for that language containment test which has complexity $O(k)$ when most of the P_i's are of a special type, which generalizes strong fairness properties. In particular, the algorithm and bound reduce to the natural generalization for testing the language emptiness of a nondeterministic Streett automaton, in which the normal acceptance condition is generalized to allow an arbitrary Boolean combination of strong fairness constraints (not just a conjunction), expressible in disjunctive normal form with k literals. The algorithm may be implemented either as a BDD-based fixed point routine, or in terms of explicit state enumeration.

1 Introduction

It is well-known that testing emptiness of language intersection

$$\cap_{i=1}^{k} \mathcal{L}(P_i) = \phi$$

for automata P_i, is PSPACE-complete [Koz77], [GJ79]. This is germane to automata-theoretic formal verification based on automata admitting of a parallel composition $\otimes$ which supports the *language intersection property*:

$$\mathcal{L}(P_1 \otimes \ldots \otimes P_k) = \cap_{i=1}^{k} \mathcal{L}(P_i)$$

as then the test

$$(*) \qquad \mathcal{L}(P_0 \otimes P_1 \otimes \cdots \otimes P_k) = \phi$$

is PSPACE-complete as well. This latter test enters into automata-theoretic verification when the P_i's model the components of a system together with its fairness constraints and the properties which are to be verified. As a result of this complexity barrier, many heuristics have been proposed for this test, including compositional techniques such as [GL91], [Lon93]. Many of these techniques are completely general and very powerful, leading to checks of $(*)$ which empirically seem to grow linearly with k, in many cases. In given problems, such compositional techniques thus often make the difference between computational tractability and intractability. However, for most interesting cases, there is no guarantee of a linear-time check.

A natural form of incremental check is to compute and then reduce each of the successive terms

$$P_0 \otimes P_1 \otimes \cdots \otimes P_i$$

for $i = 1, \ldots, k$, with the hope that internal cancellations will keep these successive terms small [GS91]. However, a commonly observed problem with this approach is that computing the middle terms (for $i \approx k/2$) very often involves an excessively large amount of computation – larger even than required to compute the final term $P_0 \otimes P_1 \otimes \cdots \otimes P_k$ directly (without benefit of successive reductions). The reason for this is that the middle terms model large and highly unconstrained systems which thus generate many states; many of these states, however, are unreachable in the complete, more constrained model.

The algorithm given in Section 3 also is based upon computations involving the successive terms $P_0 \otimes P_1 \otimes \cdots \otimes P_i$. However, each successive step involves computations in a model no larger than $O(|P_0 \otimes M|)$, where M is the largest P_i.

We assume as the underlying semantic model, the fully expressive type of ω-automata known as *L-process* [Kur90], for which the language-containment problem

$$\mathcal{L}(P_0 \otimes P_1 \otimes \cdots \otimes P_k) \subset \mathcal{L}(T)$$

may be solved in time linear in the number of edges of the specification model T. In [Kur90], this problem is transformed to the language emptiness problem $(*)$ for L-processes, in time linear in the number of edges of T. Here, we assume this transformation, and address the check $(*)$. We give a general algorithm for this check, which has complexity $O(k)$ when most of the L-processes P_i are of a special type, which generalizes strong fairness properties. In particular, the algorithm and bound reduce to the natural generalization for testing the language emptiness of a nondeterministic Streett automaton, in which the normal Streett acceptance condition is generalized to allow an arbitrary Boolean combination of strong fairness constraints (not just a conjunction), where this is expressible in disjunctive normal form with k literals. Moreover, the algorithm (but not the bound) also applies to the case where the P_i's are entirely arbitrary. The algorithm may be implemented either as a BDD-based fixed point routine, or in terms of explicit state enumeration.

More specifically, for a given L-process P, an L-process Q is defined to be *P-adic* if it possesses two properties of the Streett strong fairness acceptance condition: it is *P-faithful*, in the sense that the behavior of Q is a function of the state transitions of P, and it is *infinitary*, in the sense that its acceptance conditions depend only upon eventualities. The $O(k)$ bound applies to any P-adic L-processes. The essence of the algorithm is very simple, and closely related to a natural efficient test for emptiness of the language of a nondeterministic Streett automaton with k strong fairness constraints: each successive fairness constraint is tested against the set of fair behaviors defined by the previous constraints. Each set of fair behaviors corresponds to a set of strongly connected components of P. Therefore, for each successive constraint, it is necessary only to test the corresponding set of strongly connected components defined by the previous constraint. This gives rise to a recursive procedure which finds the strongly connected components of each strongly connected component defined by the previous constraint. Although each successive set of constraints is defined in terms of an L-process with a transition structure of its own, the fact that it is P-adic ensures that these constraints may be pulled back to the transition structure of P itself, without any ensuing blow-up of the state space.

2 Basics

Boolean Algebra

A Boolean algebra [Hal74] is a set L with distinguished elements $0, 1 \in L$, closed under the Boolean operations:

$$\begin{aligned} * &- \text{AND} \\ + &- \text{OR} \\ \sim &- \text{NOT} \end{aligned}$$

with universal element 1 and its complement 0. A Boolean algebra $L' \subset L$ is a *subalgebra* of L if L' and L share the same $0, 1$ and their operations agree. Every Boolean algebra contains the trivial 2-element Boolean algebra $\mathbb{B} = \{0, 1\}$ as a subalgebra. For $x, y \in L$, write $x \leq y$ if and only if $x * y = x$. $S(L)$— the *atoms* of L, are the nonzero elements of L, minimal with respect to $\leq$. For an arbitrary set V, define $\mathbb{B}[V]$ to be the Boolean algebra 2^V, with Boolean set operations. For notational simplicity, for $v \in V$, $\{v\} \in \mathbb{B}[V]$ may be denoted by v.

2.1 Definition For $L_1, \ldots, L_k \subset L$ subalgebras, define their (*interior*) *product*

$$\prod_{i=1}^{k} L_i = \left\{ \sum_{j \in J} x_{1j} * \cdots * x_{kj} \;\middle|\; x_{ij} \in L_i, \quad J \text{ finite} \right\} .$$

In [Sik69, §13] it is proved that for any Boolean algebras $L_1, \ldots, L_k$, there exists a Boolean algebra L such that (isomorphic copies of) $L_1, \ldots, L_k \subset L$ and $\prod_{i=1}^{k} L_i = L$. This is defined to be the *exterior* product of $L_1, \ldots, L_k$.

Transition Structure

2.2 Definition Let V be a nonempty set, and let M be a map

$$M : V^2 \to L \quad (V^2 = V \times V, \text{ the Cartesian product}).$$

Say M is an L-*matrix* with *vertices* or *state-space* $V(M) = V$, and *edges* or *transitions* $E(M) = \{e \in V^2 | M(e) \neq 0\}$. M provides the (static) transition function for automata. Note that $M(e) = \sum_{\substack{s \in S(L) \\ s \leq M(e)}} s$ (where each s is an "input letter"). For all $v \in V(M)$, define

$$s_M(v) = \sum_{w \in V(M)} M(v, w) .$$

2.3 Definition Let M, N be L-matrices with

$$V(M) \cap V(N) = \phi .$$

Their *direct sum* $M \oplus N$ is L-matrix with

$$V(M \oplus N) = V(M) \cup V(N) ,$$

defined by:

$$(M \oplus N)(v, w) = \begin{cases} M(v, w) \text{ if } v, w \in V(M) \\ N(v, w) \text{ if } v, w \in V(N) \\ \quad 0 \quad \text{otherwise} \end{cases}$$

Their *tensor product* $M \otimes N$ is L-matrix with

$$V(M \otimes N) = V(M) \times V(N) ,$$

where

$$(M \otimes N)((v, v'), (w, w')) = M(v, w) * N(v', w') .$$

2.4 Definition A *path* in M is a string $\mathbf{v} = (v_0, \ldots, v_n) \in V(M)^{n+1}$ for $n \geq 1$ such that $(v_i, v_{i+1}) \in E(M)$ for $i = 0, \ldots, n-1$. If $v_n = v_0$, $\mathbf{v}$ is a *cycle*. Say w is *reachable* from $v \in V(M)$ or $I \subset V(M)$ if there is a path $\mathbf{v}$ with $v_0 = v$ (resp., $v_0 \in I$) and $v_n = w$. Say $C \subset V(M)$ is *strongly connected* if for each $v, w \in C$, there is a path in C from v to w. (**NB**: by this definition, $\{v\}$ is strongly connected if and only if $(v, v) \in E(M)$.) A *directed graph* is a $\mathbb{B}$-matrix.

Automata

2.5 Definition An L-*process* P is a 5-tuple

$$P = (L_P,\ M_P,\ I(P),\ R(P),\ Z(P))$$

where L_P is a subalgebra of L (the *output* subalgebra), M_P is an arbitrary L-matrix, and

$$\begin{aligned} I(P) &\subset V(M_P) && (\textit{initial} \text{ states}) \\ R(P) &\subset E(M_P) && (\textit{recur} \text{ edges}) \\ Z(P) &\subset 2^{V(M_P)} && (\textit{cycle} \text{ sets}) . \end{aligned}$$

For an L-process P, write

$$V(P) \equiv V(M_P), \quad E(P) \equiv E(M_P), \quad P(v, w) \equiv M_P(v, w), \quad s_P(v) \equiv s_{M_P}(v) .$$

2.6 Definition The *selections* of an L-process P at $v \in V(P)$ are the elements of the set

$$S_P(v) = \{s \in S(L_P) \mid s * s_P(v) \neq 0\} .$$

The intended interpretation of "selection" is a set of (nondeterministic) outputs as a function of state. (These may be considered to be outputs either of the associated process or of a hidden internal process.) The nondeterministic nature of selection is an important facility for modelling abstraction: abstraction of function, achieved through modelling an algorithm by a nondeterministic choice of its possible outcomes, and abstraction of duration, achieved through modelling a specific sequence of actions by a delay of nondeterministic duration.

2.7 Definition Let M be an L-matrix and let $\mathbf{v} \in V(M)^\omega$. Set

$$\mu(\mathbf{v}) = \{v \in V(M) \mid v_i = v \text{ infinitely often}\},$$
$$\beta(\mathbf{v}) = \{e \in E(M) \mid (v_i, v_{i+1}) = e \text{ infinitely often}\} .$$

2.8 Definition Let P be an L-process. The *language* of P is the set $\mathcal{L}(P)$ of $\mathbf{x} \in S(L)^\omega$ such that for some run $\mathbf{v}$ of $\mathbf{x}$ in P with $v_0 \in I(P)$,

$$\beta(\mathbf{v}) \cap R(P) = \phi \quad \text{and} \quad \mu(\mathbf{v}) \cap (V(P) \setminus C) \neq \phi \;\; \forall C \in Z(P) .$$

Such a run $\mathbf{v}$ is called an *accepting* run of $\mathbf{x}$.

Note that if P is an L-process and L is a subalgebra of L', then P is an L'-process. However, the language of P as an L'-process is not the same as the language of P as an L-process (unless $L' = L$). In such cases, the context will make clear which language is meant.

2.9 Definition If P is an L-process and $W \subset V(P)$, define the *restriction of P to W* to be the L-process $P|_W$ with $L_{P|_W} = L_P$, $V(P|_W) = W$, $M_{P|_W}(e) = P(e)$ for all $e \in W^2$, $I(P|_W) = I(P) \cap W$, $R(P|_W) = R(P) \cap W^2$ and $Z(P|_W) = \{C \cap W \mid C \in Z(P)\}$.

2.10 Definition For an L-process P, let $W \subset V(P)$ be the states reachable from $I(P)$ through a path which may be extended to an accepting run of P, and set $P^* = P|_W$.

2.11 Lemma *P^* is an L-process and $\mathcal{L}(P^*) = \mathcal{L}(P)$.*

2.12 Definition Let $P_1, \ldots, P_k$ be L-processes. Then

$$\bigoplus_{i=1}^{k} P_i = \left(L_{P_1}, \bigoplus_i M_{P_i}, \bigcup_i I(P_i), \bigcup_i R(P_i), \bigcup_i Z(P_i) \right)$$
$$\bigotimes_{i=1}^{k} P_i = \left(\Pi_i L_{P_i}, \bigotimes M_{P_i}, \mathsf{X} I(P_i), \bigcup_i \Pi_i^{-1} R(P_i), \bigcup_i \Pi_i^{-1} Z(P_i) \right)$$

where $\Pi_i^{-1} Z(P_i) \equiv \{\Pi_i^{-1} C | C \in Z(P_i)\}$. Here, $\oplus P_i$ undefined unless $L_{P_1} = \cdots = L_{P_k}$).

2.13 Lemma *Let $P_1, \ldots, P_k$ be L-processes. Then*

$$\mathcal{L}\left(\bigoplus P_i\right) = \bigcup \mathcal{L}(P_i)$$
$$\mathcal{L}\left(\bigotimes P_i\right) = \bigcap \mathcal{L}(P_i) .$$

2.14 Lemma *For L-processes P, Q and $v \in V(P)$, $w \in V(Q)$,*

$$s_{P \otimes Q}(v, w) = s_P(v) * s_Q(w) .$$

2.15 Definition Let P be an L-process and let $Q = P|_W$ where W is the set of states of P reachable from $I(P)$. Define P^o to be the directed graph with edges $E(Q^*) \setminus R(Q^*)$. Let $\mathcal{B}(P)$ be the set of strongly connected components of P^o contained in no element of $Z(P)$.

3 P-adic Processes

The condition known as *strong-fairness*, although the foundation of the Streett automaton acceptance condition, often is conceived in purely logical terms.

3.1 Definition A *strong-fairness* constraint *on* the set S with designated set of *initial* states $I(S)$ is a pair (L, U) of subsets of S. Its *satisfaction* set is

$$\mathbf{SF}(L, U) = \{\mathbf{v} \in S^\omega \mid v_0 \in I(S), \mu(\mathbf{v}) \cap L \neq \phi \Rightarrow \mu(\mathbf{v}) \cap U \neq \phi\} .$$

For a strong-fairness constraint on the set of states $V(P)$ of an L-process P, it is to be understood that the designated set of initial states $I(V(P)) = I(P)$.

Suppose it is required to verify that an L-process P has empty language. It may be that this test fails, unless P is subject to a number of strong-fairness constraints (L_i, U_i) on $V(P)$. (This arises naturally if the system model represented here by P is presented as a Streett automaton.) The strong-fairness constraint (L_i, U_i) may be represented by a 4-state L-process, provided P "outputs its state": *i.e.*, provided the output subalgebra L_P of P contains enough information to determine the state of P from its selections. This always can be accomplished by augmenting the output subalgebra L_P so as to contain the state of P as a component, as in Example 3.2.

3.2 Example Suppose P is an L-process whose output subalgebra L_P is an exterior product of the form $L_P = L' \cdot \mathbb{B}[V(P)]$, and $s_P(v) \leq v$ for all $v \in V(P)$. Then the state of P is a component of its selection: every selection of P is of the form $x * v$ where $x \in S(L')$ and $v \in V(P)$. In this case, a strong-fairness constraint (L_i, U_i) on $V(P)$ may be represented by the 4-state L-process $Q_i = Q_i^L \oplus Q_i^U$ where Q_i^L and Q_i^U are defined as follows: for $X = L_i$ or U_i the respective transition matrix of Q_i^L or Q_i^U is

$$\begin{array}{cc} & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} 0 \\ 1 \end{array} & \begin{pmatrix} X' & X \\ X' & X \end{pmatrix} \end{array}$$

where $X' = V(P) \setminus X$, $I(Q_i^L) = I(Q_i^U) = \{0\}$, $R(Q_i^L) = \{(0,1),(1,1)\}$, $Z(Q_i^L) = \phi$, $R(Q_i^U) = \phi$, $Z(Q_i^U) = \{\{0\}\}$ and each has the trivial output subalgebra $\mathbb{B}$. Then Q_i is a $\mathbb{B}[V(P)]$-process and as such, for each run $\mathbf{v}$ of P,

$$\mathbf{v} \in \mathcal{L}(Q_i^L) \Leftrightarrow \mu(\mathbf{v}) \cap L_i = \phi \,,$$
$$\mathbf{v} \in \mathcal{L}(Q_i^U) \Leftrightarrow \mu(\mathbf{v}) \cap U_i \neq \phi$$

and

$$\mathcal{L}(Q_i) = \mathcal{L}(Q_i^L) \cup \mathcal{L}(Q_i^U) \,.$$

Thus,

$$\mathbf{v} \in \mathbf{SF}(L_i, U_i) \Leftrightarrow \mathbf{v} \in \mathcal{L}(Q_i) \,.$$

Consequently, for several strong-fairness constraints (L_i, U_i), the subset of $\mathcal{L}(P)$ whose runs all satisfy $\cap\, \mathbf{SF}(L_i, U_i)$ is precisely $\mathcal{L}(P \otimes Q)$ where $Q = \otimes Q_i$. Hence, to show that P subject to the strong-fairness constraints (L_i, U_i) has empty language, corresponds to showing

3.3 $\mathcal{L}(P \otimes Q) = \phi$.

The size of $P \otimes Q$ grows geometrically with the number of strong-fairness constraints. In fact, it can be shown that in the worst case, if $\mathcal{L}(P') = \mathcal{L}(P \otimes Q)$, then $2^k \le |V(P')|$ [HSB94]. Thus, it may seem that the computational complexity of testing (3.3) also should grow thus. However, in what follows, it is shown that this is not the case. In fact, for a class of L-processes Q_i which contains as a proper subset those L-processes derived from strong-fairness constraints (as above), the complexity of testing

3.4 $\mathcal{L}(P \otimes Q_1 \otimes \cdots \otimes Q_k) = \phi$

is only *linear* in the size of k. Moreover, we will see that it is not even necessary to test the Q_i's for membership in this special class: the algorithm to test (3.4) will have complexity which is linear in k when the Q_i's are of this class, but will test (3.4) for any L-processes Q_i whatsoever.

The next definition generalizes the context of Example 3.2.

3.5 Definition Let P, Q be L-processes. Say Q is *P-faithful* provided for all $v \in V(P)$, and $w, w' \in V(Q)$,

$$s_P(v) * Q(w, w') \neq 0 \Rightarrow s_P(v) \le Q(w, w') \,.$$

Thus, Q is P-faithful if whenever some selection of P at v enables a given transition of Q, then *every* selection at v enables that transition. In other words, Q cannot distinguish among the different selections of P at a given state, and the behavior of Q is a function of the state transitions of P.

3.6 Lemma *If Q is P-faithful and $\mathbf{x}$, $\mathbf{y} \in \mathcal{L}(P)$ share the same run of P, then $\mathbf{x} \in \mathcal{L}(Q) \Leftrightarrow \mathbf{y} \in \mathcal{L}(Q)$.*

3.7 Definition Let P be an L-process and let $L' \subset L$ be a subalgebra. Say L' is *P-faithful* provided that for any $x, y \in S(L')$, if $v \in V(P)$, $x * s_P(v) \neq 0$ and $y * s_P(v) \neq 0$, then $x = y$.

The prototype P-faithful subalgebra is the subalgebra $\mathbb{B}[V(P)]$ of Example 3.2. A P-faithful subalgebra L' is "faithful" to the state of P, inasmuch as distinct atoms $x, y \in S(L')$ correspond to distinct states of P. The atom $x \in S(L')$ "corresponds" to the state $v \in V(P)$ if $x * s_P(v) \neq 0$, and for each state v, this is true of exactly one element of $S(L')$.

3.8 Proposition *Let P be an L-process, $L' \subset L$ a P-faithful subalgebra and let Q be an L'-process. Then Q is P-faithful.*

Proof. Let $v \in V(P)$ and $w, w' \in V(Q)$. Suppose $\widehat{x} \equiv s_P(v) * Q(w, w') \neq 0$, and let $\widehat{y} = s_P(v) * \sim Q(w, w')$. By assumption, $\widehat{x} > 0$, so for some $x \in S(L')$, $x * \widehat{x} > 0$. If $\widehat{y} > 0$, then likewise for some $y \in S(L')$, $y * \widehat{y} > 0$. Thus, $x * s_P(v) \neq 0$ and $y * s_P(v) \neq 0$, so $x = y$. However, $x \leq Q(w, w')$ while $y \leq \sim Q(w, w')$, so $x = y = 0$, a contradiction. It follows that $\widehat{y} = 0$, so $s_P(v) \leq Q(w, w')$, that is, Q is P-faithful.

P-faithfulness is one half of a generalization of strong-fairness. If Q_i is the L-process constructed in Example 3.2 to implement the strong-fairness constraint (L_i, U_i), then by Proposition 3.8, Q_i is P-faithful. The other half of the generalization relates to the acceptance condition, as follows.

3.9 Definition An L-ω-language $\mathcal{L}$ is said to be *infinitary* if whenever $\mathbf{a}, \mathbf{a}' \in S(L)^*$ and $\mathbf{b} \in S(L)^\omega$, then

$$\mathbf{ab} \in \mathcal{L} \Rightarrow \mathbf{a}'\mathbf{b} \in \mathcal{L} .$$

An L-process P is *infinitary* if $\mathcal{L}(P)$ is.

Thus, a language $\mathcal{L}$ is infinitary if membership in $\mathcal{L}$ depends only upon eventualities. It is easily seen that each Q_i of Example 3.2 is infinitary. Thus, infinitary and P-faithful together generalize strong-fairness, allowing more general acceptance conditions and sequentiality (defined by the transition structure).

3.10 Definition Let P be an L-process. An L-process Q is said to be *P-adic* if Q is infinitary and P-faithful. Set

$$\mathcal{L}_P = \{\mathcal{L}(P \otimes Q) \mid Q \text{ is } P\text{-adic}\} ,$$

the *P-adic languages*.

3.11 Lemma *If Q_1, Q_2 are P-faithful (respectively, infinitary), then the same is true for $Q_1 \otimes Q_2$ and $Q_1 \oplus Q_2$. If $\mathcal{L}$ is infinitary, so is the complementary language $\mathcal{L}'$.*

Proof. Suppose Q_1, Q_2 is P-faithful. Obviously, $Q_1 \oplus Q_2$ is P-faithful. Let (w_1, w_2), $(w_1', w_2') \in V(Q_1 \otimes Q_2)$ and suppose

$$s_P(v) * (Q_1 \otimes Q_2)((w_1, w_2), (w_1', w_2')) \neq 0 .$$

Then $s_P(v) * Q_1(w_1, w_1') * Q_2(w_2, w_2') \neq 0$ so

$$s_P(v) \leq Q_1(w_1, w_1') * Q_2(w_2, w_2') = (Q_1 \otimes Q_2)((w_1, w_2), (w_1', w_2')) .$$

Suppose Q_1, Q_2 are infinitary and $ab \in \mathcal{L}(Q_1 \oplus Q_2)$. Then $ab \in \mathcal{L}(Q_i)$ for $i = 1$ or 2, so for any a', $a'b \in \mathcal{L}(Q_i) \subset \mathcal{L}(Q_1 \oplus Q_2)$. If $ab \in \mathcal{L}(Q_1 \otimes Q_2)$ then $ab \in \mathcal{L}(Q_i)$ for $i = 1$ and 2, so likewise $a'b$ for any a'.

If $\mathcal{L}$ is infinitary and $ab \in \mathcal{L}'$, let a' be chosen. If $a'b \in \mathcal{L}$ then also $ab \in \mathcal{L}$, which is impossible. Thus, $a'b \in \mathcal{L}'$.

3.12 Corollary *$\mathcal{L}_P$ is closed under union and intersection.*

3.13 Note Ken McMillan has shown that $\mathcal{L}_P$ is closed under relative complement, as well: that $\mathcal{L} \in \mathcal{L}_P \Rightarrow \mathcal{L}(P) \setminus \mathcal{L} \in \mathcal{L}_P$. For example, for Q_i as in Example 3.2, each Q_i is P-adic (as already observed). Setting $\widehat{Q}_i = \widehat{Q}_i^L \otimes \widehat{Q}_i^U$, where $\widehat{Q}_i^L$ and $\widehat{Q}_i^U$ are formed from Q_i^L and Q_i^U by interchanging the cycle set and recur edges ($Z(Q_i^L) = \{\{0\}\}$, $R(Q_i^U) = \{(0,1),(1,1)\}$, $R(Q_i^L) = Z(Q_i^U) = \phi$), gives $\mathcal{L}(\widehat{Q}_i) = \mathcal{L}(Q_i)'$. By Lemma (3.11), $\widehat{Q}_i$ is P-adic as well. Incidentally, even if P is as in Example 3.2, it is not the case that $\mathcal{L}_P = \{\mathcal{L}_f \mid f \in \mathcal{F}\}$ where $\mathcal{F}$ is the set of all Boolean combinations of satisfaction sets of strong-fairness constraints on $V(P)$, and for $f \in \mathcal{F}$, $\mathcal{L}_f = \{\mathbf{x} \in S(L)^\omega \mid \mathbf{x}$ has a run in $f\}$, although it is true that $\{\mathcal{L}_f \mid f \in \mathcal{F}\}$ is closed under complementation. The reason is that strong fairness alone cannot capture sequentiality. For example, let P be the $\mathbb{B}$-process with $V(P) = I(P) = \{0,1\}$, $R(P) = Z(P) = \phi$ and $P(i,j) = j$ for $i, j \in \{0,1\}$. Then $\mathcal{L} = (0+1)^+(01)^\omega \in \mathcal{L}_P$ but $\mathcal{L} \neq \mathcal{L}_f$ for any $f \in \mathcal{F}$.

Let P, $Q_1, \ldots, Q_k$ be arbitrary L-processes. Set $G_0 = \{V(P)\}$ and for $i \geq 1$ set

$$G_i = \left\{ \Pi_{V(P)} C \;\middle|\; C \in \mathcal{B}((P \otimes Q_i)|_{D \times V(Q_i)}),\ \ D \in G_{i-1} \right\}$$

(where $(P \otimes Q_i)|_{D \times V(Q_i)}$ is the restriction (2.9) of $P \otimes Q_i$ to $D \times V(Q_i) \subset V(P \otimes Q_i)$, and $\Pi_{V(P)} C$ is the projection of C to $V(P)$).

The following theorem shows that (3.4) may be tested in time linear in k provided each Q_i is P-adic. The algorithm consists of consecutively testing for emptiness the k sets G_i. This test has complexity $O(km)$ where $m = \max_i |E(Q_i)|$, which, incidentally, is the same complexity as testing emptiness for a deterministic Streett automaton P with k fairness constraints [Saf88].

3.14 Theorem *For P, Q_i and G_i as above,*

a) $G_k = \phi \Rightarrow \mathcal{L}(P \otimes Q_1 \otimes \cdots \otimes Q_k) = \phi$;
b) $\mathcal{L}(P \otimes Q_1 \otimes \cdots \otimes Q_k) = \phi \Rightarrow G_k = \phi$, *provided each Q_i is P-adic.*

Proof.
a) Suppose $\mathbf{x} \in \mathcal{L}(P \otimes Q_1 \otimes \cdots \otimes Q_k)$ has an accepting run $\mathbf{v}$. Then $\mathbf{v}$ has the form

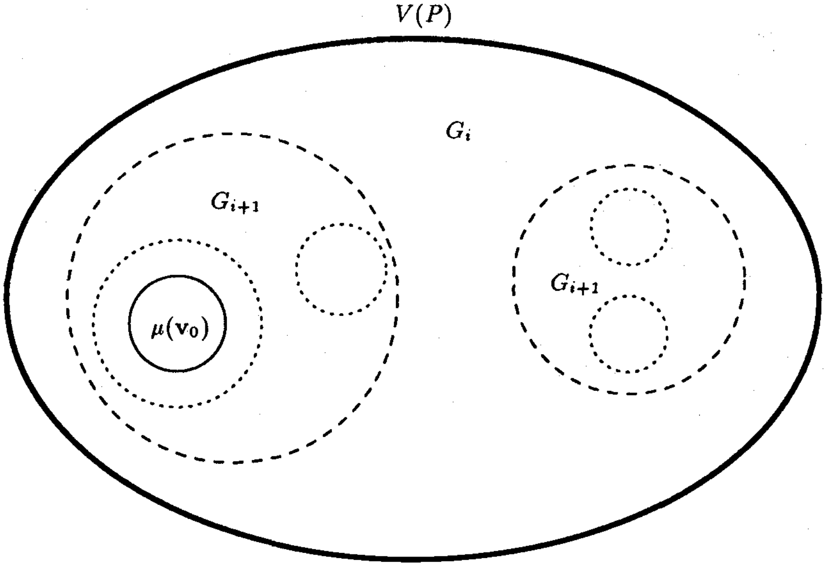

Fig. 1. Situation in the proof of (3.14)

$\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_k)$ for $\mathbf{v}_0$ an accepting run of $\mathbf{x}$ in P and $\mathbf{v}_i$ an accepting run of $\mathbf{x}$ in Q_i, for $1 \le i \le k$. Thus, for some $K \in \mathcal{B}(P \otimes Q_1 \otimes \cdots \otimes Q_k)$, $\mu(\mathbf{v}) \subset K$ and so, in particular, for some $C_1 \in \mathcal{B}(P \otimes Q_1)$, $\phi \ne \mu(\mathbf{v}_0, \mathbf{v}_1) \subset \Pi_{V(P \otimes Q_1)} K \subset C_1$ and thus

$$\mu(\mathbf{v}_0) \subset \Pi_{V(P)} K \subset \Pi_{V(P)} C_1 \in G_1 .$$

Moreover, if $\mu(\mathbf{v}_0) \subset D_i \subset D_{i-1} \subset \cdots \subset D_1$ with $D_j \in G_j$ for $1 \le j \le i$, $\Pi_{V(P)} K \subset D_i$ and $i < k$, then, since $\Pi_{V(P \otimes Q_{i+1})} K$ is strongly connected and contained in $V(P \otimes Q_{i+1})^0$, for some $C_{i+1} \in \mathcal{B}((P \otimes Q_{i+1})|_{D_i \times V(Q_{i+1})})$,

$$\mu(\mathbf{v}_0, \mathbf{v}_{i+1}) \subset \Pi_{V(P \otimes Q_{i+1})} K \subset C_{i+1}$$

and thus $D_{i+1} \equiv \Pi_{V(P)} C_{i+1} \in G_{i+1}$ and $\mu(\mathbf{v}_0) \subset D_{i+1} \subset D_i$. Hence, by induction on k, $G_k \ne \phi$.
(b) Suppose $D_k \in G_k$. Then there is some $\mathbf{x} \in \mathcal{L}(P \otimes Q_k)$ with a run $(\mathbf{v}_0, \mathbf{v}_k)$ such that $\mu(\mathbf{v}_0, \mathbf{v}_k) \subset C \in \mathcal{B}((P \otimes Q_k)|_{D_{k-1} \times V(Q_k)})$ for some C with $\Pi_{V(P)} C = D_k$ and some $D_{k-1} \in G_{k-1}$. Thus, $\mu(\mathbf{v}_0) \subset D_k \subset D_{k-1}$. Now, suppose $\mu(\mathbf{v}_0) \subset$

$D_k \subset \cdots \subset D_i$ with $D_j \in G_j$, for $i \le j \le k$, and $(\mathbf{v}_0, \mathbf{v}_{i+1}, \ldots, \mathbf{v}_k)$ is an accepting run of $\mathbf{x}$ in $P \otimes Q_{i+1} \otimes \cdots \otimes Q_k$, for some i, $1 < i < k$. Since $D_i \in G_i$, there exists some accepting run $(\mathbf{w}_0, \mathbf{w}_i)$ in $P \otimes Q_i$ of (say) $\mathbf{y} \in \mathcal{L}(P \otimes Q_i)$, with $\mu(\mathbf{w}_0) \subset D_i$. Since $D_i = \Pi_{V(P)} C$ for some $C \in \mathcal{B}((P \otimes Q_i)|_{D_{i-1} \times V(Q_i)})$ where $D_{i-1} \in G_{i-1}$, it follows that $D_i \subset D_{i-1}$. Since $D_k \subset D_i$ and D_i is strongly connected, we may suppose that in fact, for some n, $w_{0j} = v_{0j}$ for $j \ge n$ (redefining w_{0j} as necessary). Thus, for $j \ge n$, $x_j \le P(v_{0j}, v_{0j+1}) \le s_P(v_{0j})$, while $y_j \le P(v_{0j}, v_{0j+1}) * Q_i(w_{ij}, w_{ij+1})$. In particular, $y_j \le s_P(v_{0j})$ and $y_j \le Q_i(w_{ij}, w_{ij+1})$, so $s_P(v_{0j}) * Q_i(w_{ij}, w_{ij+1}) \ne 0$, for all $j \ge n$. Since Q_i is P-faithful, $s_P(v_{0j}) \le Q_i(w_{ij}, w_{ij+1})$, whereas $x_j \le s_P(v_{0j})$, and thus $x_j \le Q_i(w_{ij}, w_{ij+1})$ for all $j \ge n$. Since Q_i is infinitary, $\mathbf{x} \in \mathcal{L}(Q_i)$. Let $\mathbf{v}_i$ be an accepting run of $\mathbf{x}$ in Q_i. Then $(\mathbf{v}_0, \mathbf{v}_i)$ is an accepting run of $\mathbf{x}$ in $P \otimes Q_i$ and so $(\mathbf{v}_0, \mathbf{v}_i, \mathbf{v}_{i+1}, \ldots, \mathbf{v}_k)$ is an accepting run of $\mathbf{x}$ in $P \otimes Q_i \otimes \cdots \otimes Q_k$. It follows by induction on k that $\mathbf{x} \in \mathcal{L}(P \otimes Q_1 \otimes \cdots \otimes Q_k)$.

This theorem gives a way to check (3.4) for arbitrary Q_i's (irrespective of whether each Q_i is P-adic). The algorithm is as follows:

```
i = 0
while i < k :
        i → i + 1
        if G_i = φ, report (3.4) holds;  EXIT
find    x ∈ L(P) with accepting run³ v, μ(v) ⊂ D ∈ G_k
i = 1
while i < k :
        if x ∉ L(Q_i), repeat⁴ algorithm with P ⊗ Q_i
              in place of P, for {Q_j | j ≠ i}
        i → i + 1
report (3.4) fails — x ∈ L(P ⊗ Q_1 ⊗ ··· ⊗ Q_k)
```

The complexity of this algorithm is $O(m^k)$ for $m = \max_i |E(Q_i)|$, but reduces to $O(km)$ in case the Q_i's are P-adic. Moreover, even in the general case, the empirical complexity often may look like $O(km)$. The algorithm can be implemented either through explicit state enumeration, or in terms of a BDD fixed point routine, as in [TBK91].

4 Conclusion

We have described a general algorithm for testing that a model P defined in terms of L-processes satisfies its specification. This algorithm has complexity which is linear in the number of component L-processes, when most of these

[3] It always is possible to find $\mathbf{x}$ of the form $\mathbf{x} = \mathbf{y}' \cdot \mathbf{y}^\omega$ for $\mathbf{v}$ of the form $\mathbf{v} = \mathbf{w}' \cdot \mathbf{w}^\omega$ with $\mathbf{w}, \mathbf{w}' \in V(P)^*$. By Lemma (3.6), if the Q_i's are P-faithful, then the choice of $\mathbf{x}$ for a given $\mathbf{v}$ is immaterial.

[4] If the Q_i's are all P-adic, then this recursive call is unreachable.

L-processes are P-adic, a class which generalizes strong fairness with sequential constraints. Currently, this algorithm is being implemented into the verification tool COSPAN [HK90]; however, as this implementation is not complete, there are no concrete results to report. Nonetheless, the linear bound speaks for itself.

Acknowledgement The author thanks Ken McMillan for his careful reading and helpful comments.

References

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, 1979.

[GL91] O. Grumberg and D. E. Long. Model Checking and Modular Verification. In *Proc. CONCUR'91*, volume 527 of *Lec. Notes Comput. Sci. (LNCS).* Springer-Verlag, 1991.

[GS91] S. Graf and B. Steffen. Compositional Minimization of Finite State Systems. *Lec. Notes Comput. Sci. (LNCS)* **531**, pages 186–196, (1991).

[Hal74] P. Halmos. *Lectures on Boolean Algebras.* Springer-Verlag, 1974.

[HK90] Z. Har'El and R. P. Kurshan. Software for Analytical Development of Communications Protocol. *AT&T Tech. J.* **69**, pages 45–59, (1990).

[HSB94] R. Hojati, V. Singhal, and R. K. Brayton. Edge-Street/Edge-Rabin Automata Environment for Formal Verification Using Language Containment. LICS (to appear), 1994.

[Koz77] D. Kozen. Lower Bounds for Natural Proof Systems. *Proc 18th Symp. Found. Comput. Sci. (FOCS)*, pages 254–266, (1977).

[Kur90] R. P. Kurshan. Analysis of Discrete Event Coordination. *Lec. Notes in Comput. Sci. (LNCS)* **430**, pages 414–453, (1990).

[Lon93] D. E. Long. *Model Checking, Abstraction, and Compositional Verification.* PhD thesis, CMU, 1993.

[Saf88] S. Safra. On the complexity of ω-automata. In *Proc. 29th Found. Comput. Sci. (FOCS)*, pages 319–327, 1988.

[Sik69] R. Sikorski. *Boolean Algebras.* Springer-Verlag, 1969.

[TBK91] H. Touati, R. Brayton, and R. P. Kurshan. Testing Language Containment for ω-Automata Using BDD's. *Lec. Notes in Comput. Sci. (LNCS)*, 1991.