

The MSO Theory of Connectedly Communicating Processes

P. Madhusudan¹, P.S. Thiagarajan², and Shaofa Yang²

¹ Dept. of Computer Science, University of Illinois at Urbana-Champaign
madhu@cs.uiuc.edu

² School of Computing, National University of Singapore
{thiagu,yangsf}@comp.nus.edu.sg

Abstract. We identify a network of sequential processes that communicate by synchronizing frequently on common actions. More precisely, we demand that there is a bound k such that if the process p executes k steps without hearing from process q —directly or indirectly—then it will never hear from q again. The non-interleaved branching time behavior of a system of connectedly communicating processes (CCP) is given by its event structure unfolding. We show that the monadic second order (MSO) theory of the event structure unfolding of every CCP is decidable. Using this result, we also show that an associated distributed controller synthesis problem is decidable for linear time specifications that do not discriminate between two different linearizations of the same partially ordered execution.

1 Introduction

Sequential systems can be represented as transition systems and their behaviors can be specified and verified using a variety of linear time and branching time logics. One can view the monadic second order (MSO) logic of 1-successor interpreted over strings as the canonical linear time logic and the MSO logic of n -successors interpreted over regular trees as the canonical branching time logic [12] for sequential systems. All other reasonable logics can be viewed as specializations of these two logics with expressive power often traded in for more efficient verification procedures.

In the case of concurrent systems the situation is similar in many respects. As for models, one can choose asynchronous transition systems or 1-safe Petri nets or some other equivalent formalism [15]. In the linear time setting, Mazurkiewicz traces—viewed as restricted labelled partial orders—constitute a nice generalization of sequences and the MSO logic of sequences can be smoothly extended to Mazurkiewicz traces [1]. In the branching time setting, it is clear that labelled event structures [15] are an appropriate extension of trees. Further, just as a transition system can be unwound into a (regular) tree, so can an asynchronous transition system or 1-safe Petri net be unwound into a (regular) labelled event structure [15]. One can also define a natural MSO logic for event structures in which the causality relation (a partial order) and the conflict relation are the

non-logical predicates and quantification is carried out over individual and subsets of events. But at this stage, the correspondence between the sequential and concurrent settings breaks down.

One can say that the MSO theory—of the branching time behavior—of a transition system is the MSO theory of the tree obtained as its unwinding. According to Rabin’s famous result [11], the MSO theory of *every* finite state transition system is decidable. In the concurrent setting, it is natural to say that the MSO theory—of the non-interleaved branching time behavior—of a finite asynchronous transition system is the MSO theory of the event structure obtained as its event structure unfolding. The trouble is, it is *not* the case that the MSO theory of every finite asynchronous transition system is decidable. Hence an interesting question is: what is the precise subclass of finite asynchronous transition systems for which the MSO theory is decidable?

We provide a partial answer to this question by exhibiting a subclass of finite asynchronous transition systems called, for want of a better term, Connectedly Communicating Processes (CCPs), whose MSO theories are decidable. As the name suggests, in a CCP, processes communicate with each other frequently. More precisely, there is a bound k such that if process p executes k steps without hearing from process q either directly or indirectly and reaches a state s , then starting from s it will never hear from q again, directly or indirectly. This class of systems *properly includes* two subclasses of 1-safe net systems that we know of, which have decidable MSO theories. These two subclasses are: the sequential net systems that do not exhibit any concurrency and dually, the conflict-free net systems which do not exhibit any branching—due to choices—in their behavior.

One motivation for studying branching time temporal logics in a non-interleaved setting has to do with distributed controller synthesis. More specifically, for distributed systems, where one is interested in strategies that are not dependent on global information—and hence can be synthesized in turn as a distributed controller—one needs to look at partial order based branching time behaviors. This is the case even if the controller must satisfy just a linear time specification. Here, as an application of our main result, we establish the decidability of a distributed controller synthesis problem where the plant model is based on a CCP and the specification is a robust (trace-closed) ω -regular language. By a robust language we mean one that does not discriminate between two different interleavings of the same partially ordered execution.

The communication criterion we impose is motivated by results in undecidability of distributed control. Most undecidability proofs in distributed control rely on the undecidability of *multi-player games with partial information* where the players (in our case processes) have an unbounded loss of information on the status of other players. Our restriction ensures that the processes communicate often enough so that this partial information stays bounded.

Our proof technique consists of extracting a regular tree from the event structure induced by a CCP with the nodes of this tree corresponding to the events of the event structure such that the causality relation is definable in the MSO theory of trees. This representation is obtained directly and broadly

preserves the structure of the event structure. Similar ideas have been used in other—tenuously related—settings [3, 4].

Turning to more directly related work, a variety of branching time logics based on event structures have been proposed in the literature (see for instance [9] and the references therein) but few of them deal directly with the generalization of Rabin’s result. In this context, a closely related work is [5] where it is shown, in present terms, that the MSO theories of *all* finite asynchronous transition systems are decidable *provided* set quantification is restricted to *conflict-free* subsets of events. It is however difficult to exploit this result to solve distributed controller synthesis problems.

Following the basic undecidability result reported in [10], positive results in restricted settings are reported in [7, 8, 14]. However, [7] considers processes communicating via buffers as also [14] in a more abstract form. On the other hand, [8] imposes restrictions on communication patterns that are much more severe than the property we demand here. Our notion of strategies considered in this paper are local in the sense that each process’s strategy is based on its local view of the global history, consisting of its own sequence of actions as well as the sequence of actions executed by other agents that it comes to know about through synchronizations. The work in [6] also considers view-based strategies, and shows that for simulations, the problem is undecidable. A more recent study that uses view-based strategies is [2]. This work is also based on asynchronous transition systems, but the restrictions placed on the plants concerned is in terms of the trace alphabet associated with the plant rather than the communication patterns. As a result, this subclass is incomparable with the subclass of CCPs. Finally, decentralized controllers have also been studied (see for instance [13] and its references) where the plant is monolithic but one looks for a set of controllers each of which can control only a subset of the controllable actions.

In the next section we formulate our model and in section 3 we show that the MSO theory of every CCP is decidable. We use this result in section 4 to solve a distributed controller synthesis problem. We discuss a number of possible extensions in the concluding part of the paper. Due to lack of space, many proofs are omitted and can be found in the technical report at www.comp.nus.edu.sg/~thiagu/fsttcs05.

2 Connectedly Communicating Processes

We fix a finite set of processes \mathcal{P} and let p, q , range over \mathcal{P} . For convenience, we will often write a \mathcal{P} -indexed family $\{X_p\}_{p \in \mathcal{P}}$ simply as $\{X_p\}$. A distributed alphabet over \mathcal{P} is a pair (Σ, loc) where Σ is a finite alphabet of actions and $loc : \Sigma \rightarrow 2^{\mathcal{P}} \setminus \{\emptyset\}$ identifies for each action, a nonempty set of processes (locations) that take part in each execution of the action. Σ_p is the set of actions that p participates in and it is given by $\{a \mid p \in loc(a)\}$. Fix such a distributed alphabet for the rest of the paper.

We will formulate our model in terms of deterministic asynchronous transition systems. We impose determinacy only for convenience. All our results

will go through, with minor complications, even in the absence of determinacy. An *asynchronous transition system* (ATS) over (Σ, loc) is a structure $\mathcal{A} = (\{S_p\}, s_{in}, \{\delta_a\}_{a \in \Sigma})$ where S_p is a finite set of p -states for each p and $s_{in} \in \prod_{p \in \mathcal{P}} S_p$. Further, $\delta_a \subseteq \prod_{p \in loc(a)} S_p \times \prod_{p \in loc(a)} S_p$ for each a . The ATS \mathcal{A} is deterministic if for each a , $(s_a, s'_a), (s_a, s''_a) \in \delta_a$ implies $s'_a = s''_a$. From now on we will implicitly assume that the ATSSs we encounter are deterministic. Members of $\prod_{p \in \mathcal{P}} S_p$ are referred to as global states. It will be convenient to view the global state s as a map from \mathcal{P} into $\bigcup S_p$ such that $s(p) \in S_p$ for every p . For the global state s and $P \subseteq \mathcal{P}$, we will let s_P denote the map s restricted to P . An example of an asynchronous transition system is shown in figure 1(i), where the locations of an action is assumed are the components in which it appears as a label of a local transition.

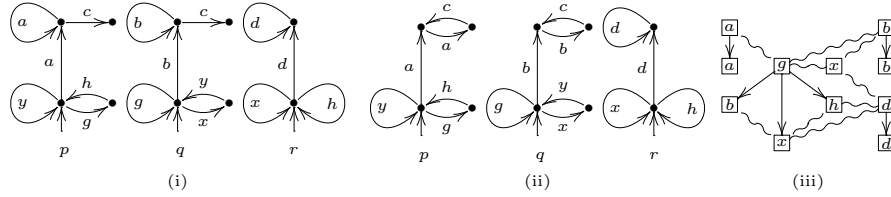


Fig. 1.

The dynamics of \mathcal{A} is given by a transition system $TS_{\mathcal{A}} = (RS_{\mathcal{A}}, s_{in}, \Sigma, \rightarrow_{\mathcal{A}})$ where $RS_{\mathcal{A}} \subseteq \prod_{p \in \mathcal{P}} S_p$, the set of reachable global states, and $\rightarrow_{\mathcal{A}} \subseteq RS_{\mathcal{A}} \times \Sigma \times RS_{\mathcal{A}}$ are least sets satisfying: Firstly, $s_{in} \in RS_{\mathcal{A}}$. Secondly, suppose $s \in RS_{\mathcal{A}}$ and $s' \in \prod_{p \in \mathcal{P}} S_p$ such that $(s_P, s'_P) \in \delta_a$ and $s_Q = s'_Q$ where $P = loc(a)$ and $Q = \mathcal{P} \setminus P$. Then $s' \in RS_{\mathcal{A}}$ and $s \xrightarrow{a}_{\mathcal{A}} s'$. We extend $\rightarrow_{\mathcal{A}}$ to sequences in Σ^* in the obvious way. We define $L(\mathcal{A}) = \{\sigma \in \Sigma^* \mid \exists s. s_{in} \xrightarrow{\sigma}_{\mathcal{A}} s\}$.

We shall use (Mazurkiewicz) trace theory to capture the notion of connectedly communicating. It will also come in handy for defining the event structure semantics of asynchronous transition systems. We first recall that a trace alphabet is a pair (Γ, I) where Γ is a finite alphabet set and $I \subseteq \Gamma \times \Gamma$ is an irreflexive and symmetric relation called the *independence relation*. The trace alphabet (Σ, I) induced by the distributed alphabet (Σ, loc) is given by $a I b$ iff $loc(a) \cap loc(b) = \emptyset$. Clearly I is irreflexive and symmetric. We let $D = (\Sigma \times \Sigma) \setminus I$ denote the dependency relation. The independence relation is extended to Σ^* via: $\sigma I \sigma'$ iff $a I b$ for every letter a that appears in σ and every letter b that appears in σ' . In what follows, we let σ, σ' range over Σ^* . As usual, \sim_I is the least equivalence relation contained in $\Sigma^* \times \Sigma^*$ such that $\sigma ab\sigma' \sim_I \sigma ba\sigma'$ whenever $a I b$. We let $\sigma \upharpoonright p$ be the Σ_p -projection of σ . It is the sequence obtained by erasing from σ all appearances of letters that are not in Σ_p . We define $|\sigma|_p = |\sigma \upharpoonright p|$ where $|\tau|$ denotes the length of the sequence τ . In what follows, we will often write \sim instead of \sim_I .

We say that two processes p and q are *separated in* σ if there exist $\tau, \tau' \in \Sigma^*$ such that $\sigma \sim \tau\tau'$ and $\tau I \tau'$ and $|\tau|_q = |\tau'|_p = 0$. Thus in the execution

represented by σ there can be no flow of information from q to p or conversely. The asynchronous transition system \mathcal{A} is *k-communicating* if for every $s \in RS_{\mathcal{A}}$ and every p, q , the following condition is satisfied: Suppose $s \xrightarrow{\sigma}_{\mathcal{A}} s'$ and $|\sigma|_p \geq k$ and $|\sigma|_q = 0$. Then p and q are separated in σ' for any $s' \xrightarrow{\sigma'}_{\mathcal{A}} s''$.

We shall say that \mathcal{A} is *connectedly communicating* iff it is *k-communicating* for some k . Clearly \mathcal{A} is connectedly communicating iff it is *K-communicating* where K is at most $|RS_{\mathcal{A}}|$. Furthermore, one can effectively determine whether \mathcal{A} is connectedly communicating. From now on we will often refer to a finite deterministic connectedly communicating ATS as a CCP. The ATS shown in figure 1(ii) is a CCP while the one shown in figure 1(i) is not. Note that the two ATSs are based on the same distributed alphabet.

3 Decidability

We wish to prove that the MSO theory of the unfolding of every CCP is decidable. To formulate this result we begin with a brief account of event structures.

An event structure (often called a prime event structure) is a triple $ES = (E, \leq, \#)$ where (E, \leq) is a poset such that for every $e \in E$, $\downarrow e = \{e' \in E \mid e' \leq e\}$ is a finite set. And $\# \subseteq E \times E$ is an irreflexive and symmetric relation such that, for every e_1, e_2 and e_3 , if $e_1 \# e_2$ and $e_2 \leq e_3$, then $e_1 \# e_3$. E is the set of events, \leq the causality relation and $\#$ the conflict relation. The minimal causality relation $<$ is defined as: $e < e'$ iff $e < e'$ and for every e'' , if $e \leq e'' \leq e'$, then $e'' = e$ or $e'' = e'$. A Σ -labelled event structure is a structure $(E, \leq, \#, \lambda)$ where $(E, \leq, \#)$ is an event structure and $\lambda : E \rightarrow \Sigma$ a labelling function.

The non-interleaved branching time behavior of \mathcal{A} is naturally given by its event structure unfolding [15]. This Σ -labelled event structure denoted $ES_{\mathcal{A}}$ is obtained as follows. We first note that $L(\mathcal{A})$ is a trace-closed subset of Σ^* in the sense if $\sigma \in L(\mathcal{A})$ and $\sigma \sim \sigma'$ then $\sigma' \in L(\mathcal{A})$ as well. For a non-null sequence $\sigma \in \Sigma^*$, let $last(\sigma)$ denote the last letter appearing in σ . In the present context, we shall view a (Mazurkiewicz) trace as a \sim -equivalence class of strings and denote the \sim -equivalence class containing the string σ as $[\sigma]_{\sim}$ and often drop the subscript \sim . The partial ordering relation \sqsubseteq over traces is given by: $[\sigma] \sqsubseteq [\sigma']$ iff there exists σ'' in $[\sigma']$ such that σ is a prefix of σ'' . A trace $[\sigma]$ is *prime* iff σ is non-null and for every σ' in $[\sigma]$, $last(\sigma) = last(\sigma')$. Thus for a prime trace $[\sigma]$, we can set $last([\sigma]) = last(\sigma)$. Now, $ES_{\mathcal{A}}$ is defined to be the structure $(E, \leq, \#, \lambda)$ where

- $E = \{[\sigma] \mid \sigma \in L(\mathcal{A}) \text{ and } [\sigma] \text{ is prime}\}$.
- \leq is \sqsubseteq restricted to $E \times E$.
- $\#$ is given by: $e \# e'$ iff there does *not* exist $\sigma \in L(\mathcal{A})$ such that $e \sqsubseteq [\sigma]$ and $e' \sqsubseteq [\sigma]$, for every $e, e' \in E$.
- $\lambda(e) = last(e)$, for every $e \in E$.

It is easy to check that $ES_{\mathcal{A}}$ is a Σ -labelled event structure. In fact, the labelling function λ will respect the dependency relation D in the sense that if $\lambda(e) D \lambda(e')$ then it will be the case that $e \leq e'$ or $e' \leq e$ or $e \# e'$. And this will endow $ES_{\mathcal{A}}$

with a great deal of additional structure. In particular, it will let us define its MSO theory using just the $<$ relation and the labelling function as it will turn out below. In what follows, we will often write $ES_{\mathcal{A}}$ as just ES .

In figure 1(iii) we show an initial fragment of the event structure unfolding of the system shown in figure 1(ii). As usual, directed arrows represent members of the $<$ relation and the squiggly edges represent the “minimal” members of the $\#$ relation. The relations \leq and $\#$ are to be deduced using the transitivity of \leq and the conflict inheritance axiom satisfied by an event structure.

We now define the syntax of the MSO logic over $ES_{\mathcal{A}}$ as:

$$MSO(ES_{\mathcal{A}}) ::= R_a(x) \mid x < y \mid x \in X \mid \exists x(\varphi) \mid \exists X(\varphi) \mid \sim \varphi \mid \varphi_1 \vee \varphi_2 ,$$

where $a \in \Sigma$, x, y, \dots are individual variables and X, Y, \dots are set variables. An interpretation \mathcal{I} assigns to every individual variable an event in E and every set variable, a subset of E . The notion of ES satisfying a formula φ under an interpretation \mathcal{I} , denoted $ES \models_{\mathcal{I}} \varphi$, is defined in the obvious way. For example, $ES \models_{\mathcal{I}} R_a(x)$ iff $\lambda(\mathcal{I}(x)) = a$; $ES \models_{\mathcal{I}} x < y$ iff $\mathcal{I}(x) < \mathcal{I}(y)$.

It is a standard observation that \leq can be defined in terms of $<$ in the presence of set quantification. We next observe that the conflict relation of $ES_{\mathcal{A}}$ admits an alternative characterization. Let the relation $\hat{\#}_D$ be given by: $e \hat{\#}_D e'$ iff $e \not\leq e'$ and $e' \not\leq e$ and $\lambda(e) D \lambda(e')$. Next define $\hat{\#}$ as: $e \hat{\#} e'$ iff there exist e_1 and e_1' such that $e_1 \hat{\#}_D e_1'$ and $e_1 \leq e$ and $e_1' \leq e'$. It is easy to verify that $\hat{\#} = \#$ and that $\hat{\#}$ is definable.

The MSO theory of ES is the set of sentences (formulas that do not have free occurrences of individual or set variables) given by: $\{\varphi \mid ES \models \varphi\}$. The MSO theory of ES is said to be decidable if there exists an effective procedure that determines for each sentence φ in $MSO(ES)$, whether $ES \models \varphi$. Finally, by the MSO theory of \mathcal{A} we shall mean the MSO theory of $ES_{\mathcal{A}}$. It is not difficult to show that the MSO theory of the asynchronous transition system in figure 1(i) is undecidable (as easily shown in our technical report). Our main result is:

Theorem 1. *Let \mathcal{A} be a CCP. Then the MSO theory of \mathcal{A} is decidable.*

Through the rest of this section, we assume \mathcal{A} is k -communicating where $k \leq |RS_{\mathcal{A}}|$. Let $TR = (\Sigma^*, \{succ_a\}_{a \in \Sigma})$ be the infinite Σ -tree, where $succ_a = \{(u, ua) \mid u \in \Sigma^*\}$. In what follows, we shall denote the standard MSO logic of n -successors ($|\Sigma| = n$) interpreted over TR as $MSO(TR)$. Its syntax is:

$$MSO(TR) ::= succ_a(x, y) \mid x \in X \mid \exists x(\varphi) \mid \exists X(\varphi) \mid \sim \varphi \mid \varphi_1 \vee \varphi_2 .$$

The semantics is the standard one [12]. We shall show that the structure $(E, <, \lambda)$ can be embedded in TR and that this embedding can be defined in $MSO(TR)$. This will at once yield theorem 1 by the result that $MSO(TR)$ is decidable [11].

Through the rest of the paper, we fix a total order lex on Σ . Often, we refer to this order implicitly, for example, by speaking of a being less than b . Clearly lex induces a total order over Σ^* which we shall refer to as the lexicographic order. For an event e in E with $e = [\sigma]$, we let $lin(e)$ be the lexicographically

least member in $[\sigma]$. Set $LEX_{\mathcal{A}} = \{ln(e) \mid e \in E\}$. In what follows, we will write $LEX_{\mathcal{A}}$ as just LEX . Clearly $LEX \subseteq \Sigma^*$ and hence members of LEX can be looked upon as distinguished nodes in the tree TR . A pleasant fact is that LEX is definable in $MSO(TR)$.

Lemma 2. *One can effectively construct a formula $\varphi_{LEX}(x)$ with one free individual variable x such that for any interpretation \mathcal{I} , $TR \models_{\mathcal{I}} \varphi_{LEX}(x)$ iff $\mathcal{I}(x) \in LEX$.*

Proof. It is easy to show that $L_{events} = \{\sigma \mid [\sigma] \in E\}$ is a regular trace-closed subset of Σ^* and is hence a regular trace language. It is known that the collection \widehat{L}_{lex} obtained by picking the lexicographically least member of each \sim -equivalence class of a regular trace language \widehat{L} is, in turn, a regular language [1]. Thus LEX is a regular subset of Σ^* and we can effectively construct from \mathcal{A} , a deterministic finite state automaton accepting LEX . Further, one can describe the successful runs of this automaton in the form of a formula $\varphi_{LEX}(x)$. \square

Define now the relation $\prec_{LEX} \subseteq LEX \times LEX$ by: $\sigma \prec_{LEX} \sigma'$ iff $[\sigma] \prec [\sigma']$ in $ES_{\mathcal{A}}$. Define also the map λ_{LEX} as $\lambda_{LEX}(\sigma) = last(\sigma)$ for every $\sigma \in LEX$. It now follows that $(LEX, \prec_{LEX}, \lambda_{LEX})$ is isomorphic to the structure (E, \prec, λ) . Hence if we show that \prec_{LEX} is definable in $MSO(TR)$ then we are done. In this light, the following result is crucial.

Lemma 3. *There exists a constant K (which can be effectively computed from \mathcal{A}) with the following property: Suppose $w = a_1 \dots a_m, w' = b_1 \dots b_n \in LEX$. Suppose further, $w \prec_{LEX} w'$ and w is not a prefix of w' . Then $|a_i a_{i+1} \dots a_m| \leq K$, where i is the least index such that $a_i \neq b_i$.*

Proof. Let $e = [w]$ and $e' = [w']$ so that $e \prec e'$. It follows from the definition of ES that $w' \sim w\tau$ for some τ in Σ^+ . Hence b_i is less than a_i . We show that $b_i I a_i a_{i+1} \dots a_m$. This will easily yield that $|a_i a_{i+1} \dots a_m| \leq k|\mathcal{P}|$, following the facts that \mathcal{A} is k -communicating and $[w']$ is prime and $w' \sim wb_i \tau'$ for some τ' in Σ^+ . Now suppose $b_i I a_i a_{i+1} \dots a_m$ does not hold. Let j ($i \leq j \leq m$) be the least index such that $a_j D b_i$. A basic property of traces is that if $a D b$ then the $\{a, b\}$ -projection of $\sigma 1$ is identical to the $\{a, b\}$ -projection of $\sigma 2$ whenever $\sigma 1 \sim \sigma 2$. It follows that $a_j = b_i$. But then b_i being less than a_i would imply that $\widehat{w} = a_1 \dots a_{i-1} b_i a_i \dots a_{j-1} a_{j+1} \dots a_m \sim w$ and clearly \widehat{w} is lexicographically less than w , a contradiction. \square

We can now show that \prec_{LEX} is expressible in $MSO(TR)$.

Lemma 4. *One can effectively construct a formula $\varphi_{\prec}(x, y)$ in $MSO(TR)$ with two free individual variables x and y such that, for any interpretation \mathcal{I} , $TR \models_{\mathcal{I}} \varphi_{\prec}(x, y)$ iff $\mathcal{I}(x), \mathcal{I}(y) \in LEX$ and $\mathcal{I}(x) \prec_{LEX} \mathcal{I}(y)$.*

Proof. Let $w, w' \in LEX$. Consider the condition $C1$ given by:

$$C1: w \text{ is a proper prefix of } w' \text{ and } last(w) D last(w') \\ \text{and } last(w) I w'' \text{ where } w' = ww''.$$

It is easy to see that if $C1$ is satisfied then $w \prec_{LEX} w'$ and moreover, $C1$ is definable in $MSO(TR)$. Let K be the constant established in lemma 3. Now consider the following conditions:

$$\begin{aligned} C2.1 : w &= w_0 a_1 a_2 \dots a_l \text{ with } l \leq K \text{ and} \\ &w' = w_0 w'_1 a_1 w'_2 a_2 \dots w'_l a_l w'_{l+1} \text{last}(w'). \\ C2.2 : w'_i &I a_j \text{ for } 1 \leq i \leq j \leq l \text{ and } a_l I w'_{l+1}. \\ C2.3 : a_l &D \text{last}(w'). \end{aligned}$$

Let $C2$ be the conjunction of $C2.1$, $C2.2$ and $C2.3$. It is easy to see that if $C2$ is satisfied then $w \prec_{LEX} w'$ and also that $C2$ is definable in $MSO(TR)$. What takes some work is showing that if $w \prec_{LEX} w'$ then $C1$ or $C2$ is satisfied. This can however be achieved by faithfully applying the definitions of LEX and \prec_{LEX} . \square

We can now establish theorem 1. Define the map $\|\cdot\|$ from $MSO(ES_{\mathcal{A}})$ into $MSO(TR)$ inductively: $\|R_a(x)\| = \exists y \text{succ}_a(y, x)$ and $\|x \prec y\| = \varphi_{\prec}(x, y)$ where $\varphi_{\prec}(x, y)$ is the formula established in lemma 4. Next we define $\|x \in X\| = x \in X$. Further, $\|\exists x(\Psi)\| = \exists x(\varphi_{LEX}(x) \wedge \|\Psi\|)$ and $\|\exists X(\Psi)\| = \exists X((\forall x \in X \varphi_{LEX}(x)) \wedge \|\Psi\|)$ where $\varphi_{LEX}(x)$ is the formula established in lemma 2. Finally, $\|\sim \Psi\| = \sim \|\Psi\|$ and $\|\Psi1 \vee \Psi2\| = \|\Psi1\| \vee \|\Psi2\|$. It is now easy to show that $ES_{\mathcal{A}} \models \Psi$ iff $TR \models \|\Psi\|$ for each sentence Ψ . It is also easy to see that our decision procedure for determining the truth of the sentence Ψ in $MSO(ES_{\mathcal{A}})$ is non-elementary in the size of Ψ but not in k .

4 Controller Synthesis

Our goal here is to define a distributed plant model based on deterministic ATSS and show the decidability of the controller synthesis problem for CCPs.

A plant is a structure $\mathcal{A} = (\{S_p^{env}\}, \{S_p^{sys}\}, s_{in}, \Sigma^{env}, \Sigma^{sys}, \{\delta_a\}_{a \in \Sigma})$ where $(\{S_p\}, s_{in}, \{\delta_a\}_{a \in \Sigma})$ is a deterministic ATS over (Σ, loc) , called the underlying ATS of \mathcal{A} with $S_p = S_p^{env} \cup S_p^{sys}$ and $S_p^{env} \cap S_p^{sys} = \emptyset$ for each p . Further, $\{\Sigma^{env}, \Sigma^{sys}\}$ is a partition of Σ such that for each a in Σ^{env} , $|loc(a)| = 1$. Finally, suppose $(s_a, s'_a) \in \delta_a$ and $p \in loc(a)$. Then $s_a(p) \in S_p^{env}$ iff $a \in \Sigma^{env}$ and hence $loc(a) = \{p\}$.

The sets S_p^{env}, S_p^{sys} are respectively the p -environment and p -system states. The sets Σ^{env} and Σ^{sys} are the environment (uncontrollable) and system (controllable) actions respectively. Each component interacts with its local environment and these interactions are enabled only when the component is in one of its environment states. We note that although the underlying ATS is deterministic, in general, a menu of controllable actions involving different processes will be available for the controller at each stage as the plant evolves. This will be the case even for the local strategies we define below. Through the rest of the section, we fix a plant \mathcal{A} as above. When talking about the behavioral aspects of \mathcal{A} , we shall identify it with its underlying ATS and will often drop the subscript \mathcal{A} . We will also say the plant is a CCP in case its underlying ATS is.

Members of $L(\mathcal{A})$ are referred to as *plays*. The set of infinite plays $L^\omega(\mathcal{A})$ is defined in the obvious way. We are interested in distributed strategies obtained by piecing together local strategies and the local views of a play will be instrumental in determining local strategies.

Let $\sigma = a_1 \dots a_n$ be a play in $L(\mathcal{A})$. The *p-view* of σ denoted $\downarrow^p(\sigma)$ is the subsequence $a_{h_1} \dots a_{h_m}$ such that $H = \{h_1, h_2, \dots, h_m\}$ is the least subset of $\{1, 2, \dots, n\}$ which satisfies: Firstly, h_m is the largest index in $\{1, 2, \dots, n\}$ such that $p \in \text{loc}(a_{h_m})$. Secondly, if $i \in H$ and $j < i$ and $a_j D a_i$, then $j \in H$. In other words, $\downarrow^p(\sigma)$ is the maximum amount of the current play that p knows about where this knowledge is gathered by its participation in the actions that have occurred in the play *and* the information it acquires as a result of synchronizations with other agents.

It will be convenient to define the set of actions that can potentially occur at a local state. For $u \in S_p$ we let $\text{act}(u)$ be the set given by: $a \in \Sigma_p$ is in $\text{act}(u)$ iff there exists (s_a, s'_a) in δ_a with $s_a(p) = u$. A *p-strategy* is a function $f : L(\mathcal{A}) \rightarrow 2^{\Sigma_p}$ which satisfies: Suppose $\sigma \in L(\mathcal{A})$ and $s_{in} \xrightarrow{\sigma} s$ with $s(p) = u$. Then $f(\sigma) \subseteq \text{act}(u)$ and moreover $f(\sigma) = \text{act}(u)$ in case $u \in S_p^{env}$. Thus a *p-strategy* recommends a subset of the structurally possible Σ_p -actions at the current *p*-state. It does so without restricting in any way the environment's choices.

The *p-strategy* f is said to be *local* if it satisfies: for every $\sigma, \sigma' \in L(\mathcal{A})$, $\downarrow^p(\sigma) \sim \downarrow^p(\sigma')$ implies $f(\sigma) = f(\sigma')$. Hence a local *p-strategy* depends only on the (partially ordered!) *p-view* of the play.

We now define a distributed strategy $Str = \{Str_p\}$ to be a family of local *p-strategies*, one for every *p*. From now, unless otherwise stated, we shall say “*p-strategy*” to mean “local *p-strategy*” and “strategy” to mean a distributed strategy.

Let $Str = \{Str_p\}$ be a strategy. The set of plays according to Str denoted $L(Str)$ is defined inductively by: Firstly, $\varepsilon \in L(Str)$. Secondly, if $\sigma \in L(Str)$ and $\sigma a \in L(\mathcal{A})$ such that $a \in Str_p(\sigma)$ for every $p \in \text{loc}(a)$, then $\sigma a \in L(Str)$. That is, an action a is allowed to execute only when it is recommended by every process taking part in a . In what follows, we will assume without loss of generality that $TS_{\mathcal{A}}$ has no deadlocks; more precisely, every reachable global state has a successor state reachable via a transition. Thus if a play according to a strategy cannot be extended it is only due to the local strategies not being able to agree on executing any system action. We will say that a strategy Str is *non-blocking* in case every play in $L(Str)$ can be extended to a longer play in $L(Str)$. This notion does not rule out the possibility of a play being extended indefinitely by just the execution of environmental actions. However one can rule out such plays by choosing the specification suitably.

To define specifications, we first define the set of infinite plays according to the strategy Str denoted $L^\omega(Str)$ in the obvious way. A *specification* is an ω -regular subset of Σ^ω which is assumed to be presented in a finite way, say, as a Büchi automaton. Let L_{spec} be a specification. A strategy Str is winning for L_{spec} iff Str is non-blocking and $L^\omega(Str) \subseteq L_{spec}$. A winning strategy for L_{spec}

is called a *controller* for the pair (\mathcal{A}, L_{spec}) . The controller synthesis problem we wish to solve is: given a pair (\mathcal{A}, L_{spec}) where \mathcal{A} is a CCP, determine whether there exists a controller for L_{spec} . We will be mainly interested in showing here that this problem is effectively solvable if the specification is *robust*.

To pin down robustness, we extend \sim to Σ^ω . This can be done in a number of equivalent ways. For our purposes it will do to define it as follows: Suppose $\sigma, \sigma' \in \Sigma^\omega$. Then $\sigma \sim \sigma'$ iff $\sigma \upharpoonright p = \sigma' \upharpoonright p$ for every p . We say that the specification L_{spec} is *robust* iff for every $\sigma, \sigma' \in \Sigma^\omega$, if $\sigma \in L_{spec}$ and $\sigma \sim \sigma'$, then $\sigma' \in L_{spec}$.

We can now state:

Theorem 5. *Given a CCP plant \mathcal{A} and a robust specification L_{spec} , one can effectively determine whether there exists a controller for (\mathcal{A}, L_{spec}) .*

In fact we can say much more as we point out in remarks following the proof of theorem 5.

4.1 Proof of Theorem 5

Throughout this subsection, we assume \mathcal{A} is a CCP and L_{spec} is robust. We shall show that the existence of a controller for (\mathcal{A}, L_{spec}) can be asserted in $MSO(ES_{\mathcal{A}})$. The required result will then follow at once from theorem 1.

In what follows, we let $ES_{\mathcal{A}} = (E, \leq, \#, \lambda)$ and often write ES instead of $ES_{\mathcal{A}}$. A *configuration* of ES is a subset $c \subseteq E$ such that $\downarrow c = c$ (where $\downarrow c = \cup_{e \in c} (\downarrow e)$) and $(c \times c) \cap \# = \emptyset$. Let c be a finite configuration. Then it is well-known that the Σ -labelled poset (c, \leq_c, λ_c) where \leq_c and λ_c are the obvious restrictions, represents a trace in the following sense. The set of linearizations of (c, \leq_c) (subjected to the point-wise application of λ_c) will be a trace, viewed as a \sim -equivalence class of strings. In fact finite and infinite configurations on the one hand and finite and infinite traces on the other hand, represent each other. It is not difficult to see that in $MSO(ES)$ one can construct a formula $infinite(X)$ with one free set variable X which asserts that X is an infinite set of events. Consequently, in $MSO(ES)$ one can define a formula $fin-conf(X)$ ($inf-conf(X)$) asserting that X is a finite (infinite) configuration.

Next we define, for $E' \subseteq E$, the p -view of E' denoted $p-view(E')$ to be the set of events given by: $e' \in p-view(E')$ iff there exists $e'' \in E'$ such that $e' \leq e''$ and $p \in loc(\lambda(e''))$. Again it is easy to see that we can define a formula $p-view(X, Y)$ asserting that Y is the p -view of X .

Now let Str be a strategy. From the definitions, it follows that $L(Str)$ is trace-closed. Hence for each $\sigma \in L(Str)$ we will have $[\sigma] \subseteq L(Str)$ and moreover, by the observation above, there will be a unique finite configuration in ES that corresponds to $[\sigma]$. We will say that E_{Str} is the *set of Str-events* and define it to be the set given by: $e \in E$ is in E_{Str} iff there exists $\sigma \in L(Str)$ such that $e = [\sigma]$. We will say that E' is *good* in case there exists a strategy Str such that E' is the set of Str -events. We can construct a formula $Good(X)$ which will assert that X is good. For arguing this, it will be convenient to assume the transition relation $\Rightarrow \subseteq C_{fin} \times E \times C_{fin}$ where C_{fin} is the set of finite configurations of ES

and \Rightarrow is given by: $c \xRightarrow{e} c'$ iff $e \notin c$ and $c' = c \cup \{e\}$. The formula $Good(X)$ will be a conjunction of the following properties all of which are easily definable in $MSO(ES)$.

- X is a nonempty set and for every finite configuration Y contained in X , if $Y \xRightarrow{e} Y'$ and $\lambda(e) \in \Sigma^{env}$ then $Y' \subseteq X$.
- If Y is a finite configuration contained in X then there exists a finite configuration Y' such that $Y \subset Y' \subseteq X$.
- Suppose Y is a finite configuration contained in X , and $Y \xRightarrow{e} Y'$. Suppose that for every p in $loc(a)$, where $a = \lambda(e)$, there exists $Y_p \subseteq X$ such that the p -view of Y_p is identical to the p -view of Y and $Y_p \xRightarrow{e1} Y'_p$ with $\lambda(e1) = a$ and $Y'_p \subseteq X$. Then $Y' \subseteq X$.

All we need now is to argue that we can assert that every infinite play belonging to a good set meets the specification. But this is easy to do since L_{spec} is robust. It follows that L_{spec} in fact is an ω -regular trace language and is hence definable in the Monadic Second Order logic of infinite traces interpreted over the set of infinite traces generated by our trace alphabet (Σ, I) [1]. Denoting this logical language by $MSO(\Sigma, loc)$, we can assume, without loss of generality, that its syntax is *exactly* that of $MSO(ES)$ but interpreted over infinite traces represented as Σ -labelled partial orders. In particular, the $<$ refers to the partial order of the trace rather than the positional order of a linearization of the trace.

Now let Φ_{spec} be a sentence in $MSO(\Sigma, loc)$ such that the ω -regular trace language defined by it is precisely L_{spec} . We can now assert in $MSO(ES)$ that there exists X such that X is a good set and moreover, for every infinite configuration Y contained in X , the trace corresponding to Y satisfies Φ_{spec} . It is routine to show that this sentence, say $\Psi_{controller}$, is satisfiable in $MSO(ES)$ iff there exists a controller for (\mathcal{A}, L_{spec}) .

5 Discussion

Here we informally sketch a number of additional results that one can derive for our ATSS. To start with, theorem 5 can be considerably strengthened. In case a controller exists then there exists a finite state one which can be effectively computed and synthesized as a (finite deterministic) CCP over (Σ, loc) . This controller will synchronize with the plant on common actions and the resulting controlled behavior will meet the specification. Developing this result however requires additional work and machinery. It also requires us to work with TR , the tree representation of ES , rather than with ES itself. By adapting the arguments developed in [10] we can also show quite easily that the controller synthesis problem is undecidable for CCP plants in case the specification is allowed to be non-robust.

Clearly we can assume the specification for the controller synthesis problem to be given as a sentence in $MSO(ES)$ and our argument for decidability will extend smoothly. One can also assume that the plant itself is not a CCP but require, for robust specifications, the controller be a CCP. More precisely, we say

that the strategy Str is k -communicating iff for every $\sigma \in L(Str)$, if $\sigma\sigma' \in L(Str)$ and $|\sigma'|_p \geq k$ and $|\sigma'|_q = 0$, then for every $\sigma\sigma'\sigma'' \in L(Str)$, p, q are separated in σ'' . We say Str is *connectedly communicating* iff Str is k -communicating for some integer k . We conjecture that the k -communicating controller synthesis problem for a given k is decidable and in case such a controller exists, a finite state one exists as well and it can be effectively synthesized. It is also interesting to determine if the connectedly communicating controller synthesis problem is decidable. In other words, given a plant and a robust specification determine if there exists a k -communicating controller for some k . We conjecture that this problem is undecidable.

References

- [1] V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995.
- [2] P. Gastin, B. Lerman, and M. Zeitoun. Distributed games with causal memory are decidable for series-parallel systems. In *FSTTCS '04, LNCS 3328*, pages 275–286. Springer, 2004.
- [3] D. Kuske. Regular sets of infinite message sequence charts. *Information and Computation*, 187:80–109, 2003.
- [4] P. Madhusudan. Reasoning about sequential and branching behaviours of message sequence graphs. In *ICALP '00, LNCS 2076*, pages 396–407. Springer, 2000.
- [5] P. Madhusudan. Model-checking trace event structures. In *LICS '03*, pages 371–380. IEEE Press, 2003.
- [6] P. Madhusudan and P.S. Thiagarajan. Controllers for discrete event systems via morphisms. In *CONCUR '98, LNCS 1466*, pages 18–33. Springer, 1998.
- [7] P. Madhusudan and P.S. Thiagarajan. Distributed control and synthesis for local specifications. In *ICALP '01, LNCS 2076*, pages 396–407. Springer, 2001.
- [8] P. Madhusudan and P.S. Thiagarajan. A decidable class of asynchronous distributed controllers. In *CONCUR '02, LNCS 2421*, pages 145–160. Springer, 2002.
- [9] W. Penczek. Model-checking for a subclass of event structures. In *TACAS '97, LNCS 1217*, pages 146–164. Springer, 1997.
- [10] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *FOCS '90*, pages 746–757. IEEE Press, 1990.
- [11] M. Rabin. Decidability of second order theories and automata on infinite trees. *Trans. of AMS*, 141:1–35, 1969.
- [12] W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Comp. Sci., Vol. B*. Elsevier, 1990.
- [13] S. Tripakis. Decentralized control of discrete event systems with bounded or unbounded delay communication. *IEEE Trans. on Automatic Control*, 49:1489–1501, 2004.
- [14] I. Walukiewicz and S. Mohalik. Distributed games. In *FSTTCS '03, LNCS 2914*, pages 338–351. Springer, 2003.
- [15] G. Winskel and M. Nielsen. Models for concurrency. In *Handbook of Logic in Comp. Sci., Vol. 3*. Oxford University Press, 1994.