Model-checking Trace Event Structures

P. Madhusudan*
University of Pennsylvania

Abstract

*Given a regular collection of Mazurkiewicz traces, which can be seen as the behaviours of a finite-state concurrent system, one can associate with it a canonical regular event structure. This event structure is a single (often infinite) structure that captures both the concurrency and conflict information present in the system. We study the problem of model-checking such structures against logics such as first-order logic (*FOL*), monadic second-order logic (*MSOL*) and a new logic that lies in between these two called monadic trace logic (*MTL*).* MTL *is a fragment of* MSOL *where the quantification is restricted to sets that are conflict-free. While it is known that model-checking such event structures against* MSOL *is undecidable, our main results are that* FOL *and* MTL *admit effective model-checking procedures. It turns out that* FOL *captures previously known decidable temporal logics on event structures.* MTL *is more powerful and can express interesting branching-time properties of event structures, and when restricted to a sequential setting, can express the standard logic* CTL* *over trees.*

1. Introduction

Systems and communication protocols often comprise of several machines that work concurrently at different speeds and interact with each other. Model-checking, which has been successful in handling industrial size verification problems, is the paradigm where systems are abstracted into finite-state models that are then automatically checked for correctness against formal specifications. Concurrency amongst the components is largely ignored and the model is "flattened" into a large global system where concurrency is replaced by nondeterminism using an interleaving semantics.

There has been a large body of work, however, that has tried to identify what the true semantics of a concurrent system is. Notions of partially ordered behaviour have been defined that preserve the concurrency present in the system. A popular notion is that of (Mazurkiewicz) traces, which is a class of restricted labelled partial orders. One starts with a concurrent alphabet, which is a pair (Σ, I), where Σ is a finite alphabet and I is a binary independence relation over Σ that identifies which actions in Σ are independent of each other. A trace over (Σ, I) is a partial order which is labelled with letters in Σ and where the labelling respects the independence relation I. Such a trace describes a single execution of a concurrent system and plays the role strings play in the world of sequential systems. This class of partial orders has very robust properties like that of words and analogues of regularity, recognizability and monadic second-order definability have been well studied.

An argument for the practical motivation for the study of such behaviours is that checking these partial orders, instead of checking all the linearizations of it, could alleviate the state-space explosion problem. This is usually referred to as the "partial-order reduction technique" and has been implemented, for example in SPIN [7], with reasonable success.

Trees serve as the standard semantics for describing the branching behaviour of sequential systems. The unfolding of a system into a tree is a single structure that captures all the executions of the system and depicts also the branching aspects of it. *Event structures* [21] are the analogue of trees for concurrent systems—an event structure is a single object that preserves *both* the concurrency and conflict (branching) information in the system. Event structures are partial orders equipped with a *conflict* relation which relate conflicting events of the system (events that cannot belong to the same execution of the system). Event structures have been well studied and associations between the event-structure semantics of various concurrency models like Petri nets and distributed transition systems are well understood [21].

Event structures hence seem a natural behaviour model to study *branching-time logics* for concurrent systems. The problem, however, is that even the event structures that correspond to simple finite-state concurrent systems are complex structures and are difficult to reason with. Earlier work has focussed on trying to find restricted modal logics on restricted classes of event structures that have a decidable model-checking problem. For example, Penczek [15] iden-

*This research was supported by NSF award CCR99-70925, and NSF award ITR/SY 0121431.

tifies a modal logic on the class of *free-choice event structures* for which the model-checking problem is decidable.

In this paper, we consider the problem of model checking an event structure against classical (non-temporal) logics such as first-order logic (FOL) and monadic second-order logic (MSOL) specifications. The class of event structures we consider are *regular trace event structures*. This class of event structures can be presented by a regular prefix-closed language $\mathcal{L}$ of traces over a distributed alphabet. Such a language can be finitely presented (say as a finite automaton accepting the language) and we can associate with it a canonical labelled event structure such that the "executions" of the event structure is precisely $\mathcal{L}$. This event structure is regular where the notion of regularity is a natural one that intuitively says that there are only a finite number of non-isomorphic "suffixes" in the event structure. It is also a *trace* event structure in the sense that the labels on the events respect the dependency constraints in the concurrent alphabet.

The class of regular trace event structures is very powerful and encompasses behaviours of all natural finite-state models of concurrent systems. For example, for any 1-safe Petri net one can define a distributed alphabet such that the event structure corresponding to the net is a regular trace event structure. In fact, in [16, 13] it is conjectured that any regular event structure (which is also "bounded branching") must be a regular trace event structure.

Regular trace event structures are however difficult to verify. As pointed out in [16] (attributed to Walukiewicz), the model-checking problem for regular trace event structures against monadic second-order logic (MSOL) (which is the canonical decidable logic over trees) is undecidable. This means that these are not tree-interpretable and is perhaps the primary reason why a powerful decidable logic over event structures has been difficult to come up with.

The main result in this paper is that a restriction of monadic second-order logic, called monadic trace logic (MTL) can be effectively model-checked on regular trace event structures. MTL has the same syntax as MSOL but its semantics allow second-order variables to quantify only over *conflict-free* sets of events in the event structure. In other words, the elements of a set quantified with a variable lie along a single execution (trace) of the event structure.

We first show that first-order logic FOL over regular trace event structures is decidable. The main idea of the proof stems from the automata-theoretic proof of decidability of Presburger arithmetic and that of FOL model-checking of *automatic structures* (also called *synchronized rational graphs*). Automatic structures [1] are structures where the set of elements can be represented by a regular set of words and relations can be captured by synchronized rational relations [5] over the words representing the related elements. The crux of showing FOL decidability is to show that every FOL-definable relation over an automatic structure corresponds to a synchronized rational relation over words (see [19] for an excellent survey). We extend this methodology from words to traces. First, we show how a tuple of traces can be *braided* into a single trace over a new alphabet. This allows us to represent relations over traces as a language of braided traces over the new alphabet. We then prove that any FOL-definable relation corresponds to a regular set of braided traces.

The results are then extended to model-check MTL properties, where the main idea behind the proof is that sets of events that are conflict-free can be represented using appropriately annotated *infinite* traces. This calls for extending the procedure for FOL above to deal with infinite traces and regular relations involving finite and infinite traces.

FOL turns out to be quite an expressive logic in itself. (Note that in our structures we consider the causality relation and not the *immediate* causality relation; otherwise, FOL would be very weak as it would not be able to causally relate events unboundedly far away from each other.) The only work we know that explicitly deals with model checking event structures is that reported in [15]. One can show that all modal logics over event structures present in the literature (including the modal logic of [15] and the tense logics in [11]) can easily be translated to FOL. While in [15] a modal logic is shown only to be decidable over the restricted class of free-choice event structures, our result generalizes this to a larger class.

The logic MTL is more powerful and can express properties such as "for every finite trace of the system, there is an extension of it that satisfies the MSOL property φ on traces" which is the analogue of the CTL* property $\mathsf{AG\ EF}\ \varphi$.

When the independence relation is empty (i.e., when there is no concurrency), traces degenerate to words and event-structures degenerate to trees. The logic MTL degenerates to monadic path logic (MPL) over trees, which has been studied in the literature. In [6], it was shown that MPL can capture CTL* over trees and hence is quite a powerful logic. We therefore believe that MTL is a very expressive logic and that any reasonable temporal logic over event structures should be embeddable into it.

2. Preliminaries

Let us fix some basic notation. For an alphabet Σ, let Σ^* denote the set of finite words of Σ, Σ^ω denote the set of ω length words over Σ and let $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. For any $k \in \mathbb{N}$, let $[k]$ denote the set $\{i \mid 1 \leq i \leq k\}$.

Let $(S, \leq)$ be a partially ordered set (i.e. $\leq$ is a partial order on S). Then $<$ denotes the corresponding relation which is not reflexive, i.e. $< = \leq \setminus \{(s,s) \mid s \in S\}$. For any $s \in S$, $\downarrow s$ is the set of elements in S that are below s, i.e. the set $\{s' \in S \mid s' \leq s\}$. Also, let $\lessdot$ denote the immediate partial-order relation: $s \lessdot s'$ iff $s < s'$ and there

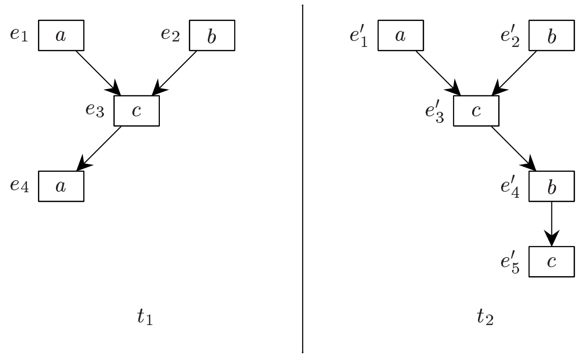

Figure 1. Two traces t_1 and t_2 over $(\{a, b, c\}, \{(a, b), (b, a)\})$

is no s'' such that $s < s'' < s'$. Note that it is not necessary that the reflexive and transitive closure of $<\!\cdot$ be the same as $\leq$; however, if for every $s \in S$, $\downarrow s$ is finite, then this is indeed the case. The partially ordered sets we deal with will all have this property and we assume this without explicitly mentioning each time.

Traces

A *trace alphabet* is a pair (Σ, I) where Σ is an alphabet containing a finite set of actions and $I \subseteq \Sigma \times \Sigma$ is an irreflexive and symmetric relation over Σ called the *independence* relation. The induced relation $D = (\Sigma \times \Sigma) \setminus I$ is called the *dependence* relation.

A *(Mazurkiewicz) trace* over the trace alphabet (Σ, I) is a Σ-labelled poset $t = (E, \leq, \lambda)$ where E is a finite or countable set of events, $\leq$ is a partial order on E, called the causal order, and $\lambda : E \to \Sigma$ is a labelling function such that the following hold:

(T1) $\forall e \in E, \downarrow e$ is finite

(T2) $\forall e, e' \in E,\ e <\!\cdot\, e' \Rightarrow \lambda(e) D \lambda(e')$

(T3) $\forall e, e' \in E,\ \lambda(e) D \lambda(e') \Rightarrow e \leq e'$ or $e' \leq e$

(T2) says that events that are immediately causally related must correspond to dependent actions and (T3) says that any two events with dependent labels must be causally related. When $I = \emptyset$, all events are causally related and a trace degenerates to a word over Σ. See [2] for further details on traces.

For example, Figure 1 depicts two traces t_1 and t_2. In the diagrams, we show only the immediate causal relation using arrows; the labels of events are shown inside the box while the event name is shown outside.

Let $\mathcal{T}r(\Sigma, I)$ denote the set of all traces over (Σ, I), and let $\mathcal{T}r^f(\Sigma, I)$ and $\mathcal{T}r^\omega(\Sigma, I)$ denote the set of finite and infinite traces over (Σ, I) (i.e. where the event set E is finite or infinite, respectively). A trace $t = (E, \leq, \lambda)$ is isomorphic to a trace $t' = (E', \leq', \lambda')$, denoted $t \approx t'$, if they are structurally isomorphic, i.e. there is a one-to-one correspondence between E and E' that preserves the partial order relations and the labelling functions. We do not distinguish between isomorphic traces.

If $t = (E, \leq, \lambda)$ is a trace, and $e \in E$, let $\Downarrow e$ be the sub-trace defined by $\downarrow e$, i.e. $\Downarrow e = (\downarrow e, \leq', \lambda')$ where $\leq'$ and λ' are $\leq$ and λ, respectively, restricted to $\downarrow e$. It is easy to see that $\Downarrow e$ is a trace and that if $e, e' \in E$ with $\Downarrow e \approx \Downarrow e'$, then $e = e'$.

Let us define concatenation over traces, which is basically obtained by taking the union of the two traces and causally relating events of the first trace to the events in the second trace that have dependent labels. Let $t_1 \in \mathcal{T}r^f(\Sigma, I)$, $t_2 \in \mathcal{T}r(\Sigma, I)$, with $t_1 = (E_1, \leq_1, \lambda_1)$ and $t_2 = (E_2, \leq_2, \lambda_2)$. Assume without loss of generality that E_1 and E_2 are disjoint. Then $t_1 \cdot t_2 = (E, \leq, \lambda)$ where $E = E_1 \cup E_2$, $\leq = (\leq')^*$, the reflexive and transitive closure of $\leq'$, where $\leq' = \leq_1 \cup \leq_2 \cup \{(e_1, e_2) \mid e_1 \in E_1, e_2 \in E_2, \lambda(e_1) D \lambda(e_2)\}$, and the labelling function is given by $\lambda(e) = \lambda_i(e)$ if $e \in E_i$, $i \in \{1, 2\}$. It is easy to verify that $t_1 \cdot t_2$ is a trace in $\mathcal{T}r(\Sigma, I)$.

We also define a prefix relation on traces: let $\sqsubseteq$ be the relation over traces over (Σ, I) such that $t_1 \sqsubseteq t_2$ iff there is some trace $t \in \mathcal{T}r(\Sigma, I)$ such that $t_1 \cdot t = t_2$. Note that $\sqsubseteq$ is a partial order on $\mathcal{T}r(\Sigma, I)$.

Two finite traces are said to be in conflict if they have no possible common extension. That is, t_1 is in conflict with t_2, denoted $t_1 \not\uparrow t_2$, if there is no trace $t \in \mathcal{T}r(\Sigma, I)$ such that $t_1 \sqsubseteq t$ and $t_2 \sqsubseteq t$. The conflict relation is clearly symmetric and irreflexive. Intuitively, conflicting traces describe two finite behaviours that are truly branching. For example, in Figure 1, t_1 and t_2 are in conflict.

For a trace $t = (E, \leq, \lambda)$, a *linearization* of t is a linearization (at most an ω-linearization) of its events that respects the partial order, i.e. a structure $(E, \leq', \lambda')$ where $\leq'$ is an (at most ω) linear order with $\leq\ \subseteq\ \leq'$. But a linearly ordered Σ-labelled structure is really a word over Σ. Hence the set of linearizations of t, denoted $lin(t)$, can be seen as a subset of Σ^*, if t is finite, or a subset of Σ^ω if t is infinite. Note that any infinite trace t has at least one ω linearization.

The conditions (T1) and (T2) make traces a very restricted kind of labelled partial order. In fact, linearizations of traces have as much information as the trace itself. One can verify for example that every word $\sigma \in \Sigma^\infty$ is a linearization of a unique trace over (Σ, I). This hence induces a natural equivalence relation over Σ^∞: $\sigma \equiv_I \sigma'$ iff σ and σ' are linearizations of the same trace. We often refer to a trace using the notation $[\sigma]$, i.e. using the equivalence class of linearizations of the trace. For example, in Figure 1, $t_1 = [abca]$ since $abca$ is a linearization of t_1.

A trace language over (Σ, I) is a set of traces over (Σ, I).

There are several ways to define regular languages of finite traces. The following definition suits us best: A language of finite traces $\mathcal{L}$ over (Σ, I) is *regular* if the set of its linearizations, $lin(\mathcal{L})$ is a regular subset of Σ^*. (Regular languages of words are the standard notion and we assume the reader's familiarity with this.) Note that regular trace languages over (Σ, I) are closed under union, intersection and complement (since regular word languages are closed under these operations).

Event Structures

A Σ*-labelled (prime) event structure*, introduced first in [12], is a structure $\mathcal{ES} = (Ev, \preceq, \#, \eta)$ where Ev is a finite or countable set of *events*, $\preceq$ is a partial order on Ev called the *causality relation*, $\# \subseteq Ev \times Ev$ is an irreflexive and symmetric relation called the *conflict relation*, and $\eta : Ev \to \Sigma$ is a labelling function such that the following hold:

(ES1) If $e_1, e_2, e \in Ev$ and $e_1 \# e_2 \preceq e$, then $e_1 \# e$

(ES2) For every $e \in Ev$, $\downarrow e$ is finite.

(ES1) says that if two events are in conflict, then their future events are also in conflict. This also ensures that if $e \leq e'$, then $\neg(e \# e')$, i.e. causally related events are not in conflict. Hence $\downarrow e$, for any $e \in E$, is a conflict-free set. Note that we call elements of both traces as well as event-structures "events"; this will cause no confusion, however, as it will be clear from context what me mean. We refer the reader to [21] for the literature on event structures.

For an event structure $(Ev, \preceq, \#, \eta)$, let $\#_\mu$ denote the relation relating the minimally conflicting events: $e_1 \#_\mu e_2$ iff $e_1 \# e_2$ and for every $e_1' \preceq e_1$, $e_2' \preceq e_2$, if $e_1' \# e_2'$ then $e_1' = e_1$ and $e_2' = e_2$.

Now consider a set $\mathcal{L}$ of traces over (Σ, I) which is prefix closed (i.e. $\forall t \in \mathcal{L}, \forall t' \sqsubseteq t, t' \in \mathcal{L}$). Then $\mathcal{L}$ can be seen as an event structure. In order to define this, we first need to define prime traces. A trace t is *prime* if it has exactly one maximal element. Let $pr(\mathcal{L})$ denote the set of prime traces in $\mathcal{L}$. Also, for a prime trace t, let $max(t)$ denote its maximal element.

If $\mathcal{L} \subseteq \mathcal{T}r(\Sigma, I)$ is a prefix-closed language of traces, let $\mathcal{ES}_\mathcal{L} = (Ev, \preceq, \#, \eta)$ where $Ev = pr(\mathcal{L})$, $\preceq = \sqsubseteq \cap (Ev \times Ev)$, $\# = \not\uparrow \cap (Ev \times Ev)$, and $\eta(t) = \lambda(max(t))$. In other words, we take the set of prime traces to be the events of the event structure, take the prefix and conflict relation on traces as the causality and conflict relations, respectively, and label each event of the event structure, which is a prime trace, using the label of the maximal element of the prime trace. The event structure $\mathcal{ES}_\mathcal{L}$ is strongly related to $\mathcal{L}$: for example, one can verify that $\downarrow t$ in the event structure is isomorphic to t and hence $\mathcal{L}$ is recoverable from $\mathcal{ES}_\mathcal{L}$. See [16, 21] for a formal correspondence between $\mathcal{ES}_\mathcal{L}$ and $\mathcal{L}$.

Event structures thus obtained from languages of traces are special in the sense that the labels respect the causal and conflict relations. Let us identify this formally. A *trace event structure* over a trace alphabet (Σ, I) is a Σ-labelled event structure $\mathcal{ES} = (Ev, \preceq, \#, \eta)$ which satisfies the following: for every $e, e' \in Ev$,

(TES1) If $e \#_\mu e'$, then $\eta(e) \neq \eta(e')$

(TES2) If $e \prec\!\!\cdot\; e'$ or $e \#_\mu e'$, then $\eta(e) D \eta(e')$

(TES3) If $\eta(e) D \eta(e')$, then $e \preceq e'$ or $e' \preceq e$ or $e \# e'$.

Condition (TES1) is a determinacy condition that intuitively says that each prime trace is represented by at most one event (if $e \#_\mu e'$ and $\eta(e) = \eta(e')$, then $\downarrow e$ and $\downarrow e'$ define the same prime trace). (TES2) and (TES3) reflect the obvious connection between the labels, the causality and conflict relation to the independence relation.

We then have:

Proposition 2.1 *([16]) For any language $\mathcal{L}$, $\mathcal{ES}_\mathcal{L}$ is a trace event structure.*

Though this is all we need, $\mathcal{L}$ and $\mathcal{ES}_\mathcal{L}$ are more strongly related. (see [16, 21] for details).

Finally, a *regular trace event structure* is an event structure $\mathcal{ES}$ such that $\mathcal{ES} = \mathcal{ES}_\mathcal{L}$ for some *regular* trace language $\mathcal{L}$. Regular trace event structures can thus be presented by a finite automaton over Σ^* that accepts $lin(\mathcal{L})$. We assume this presentation in the formulation of decidability results below.

For example, consider the alphabet $\Sigma = \{a, b, c\}$ and where aIb. Consider the language of traces $\mathcal{L}_1 = ([abc] + [aabbc])^*$ (where regular expressions over traces have the obvious interpretation). Let $\mathcal{L}$ be the prefix-closure of $\mathcal{L}_1$. Note that traces such as $a.b$ are not prime traces in $\mathcal{L}$. The event structure corresponding to $\mathcal{L}$ is then shown in Figure 2. In the diagram, we denote the immediate causal relation using arrows and denote only the minimal conflict relation using a dashed edge.

Logics over event structures

For any logical relational structure $(U, R_1, \ldots, R_n)$, where U is the universe and each R_i is a relation over U, the first order logic over the structure is the standard one that allows quantification over U, checking equality, checking the relations R_i ($i \in [n]$) between quantified elements and closed under boolean operations.

A labelled event structure $(Ev, \preceq, \#, \eta)$ can be viewed as a structure $(Ev, \preceq, \#, \eta_{a_1}, \ldots, \eta_{a_n})$ where $\Sigma = \{a_1, \ldots, a_n\}$ and each η_{a_i} is a predicate that identifies the elements labelled a_i.

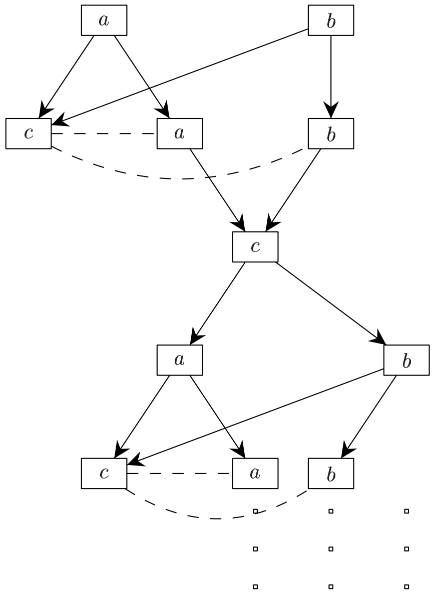

Figure 2. Event structure corresponding to $\mathcal{L}$

FOL over event structures is then the logic:

$$\varphi ::= x = x' \mid x \preceq x' \mid x \# x' \mid \eta_a(x) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists x.\varphi(x)$$

where x, x' are variables and $a \in \Sigma$.

Similarly, monadic second-order logic (MSOL) extends FOL in that it also allows quantification over *sets* of events:

$$\begin{aligned} \text{MSOL} : \varphi \quad ::= \quad & x = x' \mid x \preceq x' \mid x \# x' \mid \eta_a(x) \mid x \in X \\ & \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists x.\varphi(x) \mid \exists X.\varphi(X) \end{aligned}$$

The semantics of FOL and MSOL are the standard semantics on structures.

On event structures, we consider a new logic called *monadic trace logic* (MTL) which is an extension of FOL and a restriction of MSOL. The syntax of MTL is the same as that of MSOL; however, semantically, set-quantifiers are not interpreted over *all* possible sets of events but only over *conflict-free* sets of events. In other words, a variable X can be interpreted as a set of events $F \subseteq Ev$ only if F has no two events that are in conflict. Hence one can find a single trace such that all the events in F belong to the trace (and hence the name monadic trace logic).

For any of these logics $\text{LOG} \in \{\text{FOL}, \text{MSOL}, \text{MTL}\}$, the model-checking problem for regular trace event structures against specifications in LOG is the following: Given a finite presentation of a regular trace event structure $\mathcal{ES}_{\mathcal{L}}$ and a formula $\varphi \in \text{LOG}$, does $\mathcal{ES}_{\mathcal{L}} \models \varphi$?

Logics over event structures can state interesting properties of the underlying system. For example, we can state properties involving the knowledge of agents: "whenever agent A does an action a, it knows whether agent B will eventually do an action b or not". Such properties cannot be stated (as regular properties) in a sequential setting over trees. We refer the reader to [14] for various logics on traces and event structures.

It has been observed that MSOL is too expressive a logic and leads to undecidability (from [16], attributed to Walukiewicz):

Proposition 2.2 ([16]) *The model-checking problem for regular trace event structures against* MSOL *specifications is undecidable.*

The idea in the proof of the above proposition is to consider $\Sigma = \{a, b, c\}$, $I = \{(a, b), (b, a)\}$ and the prefix-closure of the language $\{[w] \mid w \in a^*b^*c\}$. The c-events then stick-out at the grid points defined by a and b, and an MSOL formula can encode a tiling problem.
We can now state our main result:

Theorem 2.3 *The model-checking problem for regular trace event structures against* FOL *and* MTL *specifications is decidable.*

In the next section, we show how to model-check FOL specifications and in Section 4 we extend this procedure to model-check MTL.

3. Model checking FOL properties

We consider model-checking FOL specifications on trace event structures. FOL is actually powerful enough to express most modal logics on event structures studied in the literature [11, 15]. Modal logics typically have modalities that refer to either future events, past events, conflicting events, minimally conflicting events or concurrent events. All such modalities can easily be expressed in FOL. Also, the "gossip" modality is sometimes used especially when the event structure explicitly comprises n communicating sequential agents [9, 20]; this modality can also be expressed in FOL. In fact, it seems to be hard to come up with any natural modality that is not expressible in FOL.

The proof of decidability of model checking FOL properties is shown using the following rough sketch. First, we introduce a way of representing a k-tuple of traces as a single trace, called a *braided trace* (over a different alphabet). This then allows us to represent any k-ary relation over traces as a language of braided traces over the new alphabet. This notion is similar to the way synchronous relations are defined for words [5].

We then consider a regular trace event structure given by a regular trace language $\mathcal{L}$ over (Σ, I). Note that each event in this event structure is represented by a prime trace in $\mathcal{L}$.

Now let $\varphi(x_1, \ldots x_k)$ be any FOL formula with free variables $x_1, \ldots, x_k$. Then φ induces a k-ary relation over the events in the event structure, and hence a relation $R^{\mathcal{L}}_{\varphi}$ over the prime traces in $\mathcal{L}$. The crux of the proof is to show that the braided traces that represent $R^{\mathcal{L}}_{\varphi}$ forms an effective regular trace language. This is shown by structural induction on φ. This approach finally reduces the problem of model-checking $\mathcal{ES}_{\mathcal{L}}$ against φ to the emptiness problem for automata. In this section, all traces that we consider will be finite traces.

Braiding traces

Let us fix for this section a trace alphabet (Σ, I). If $\mathcal{L}'$ is a prefix-closed regular trace language, then is easy to see that $pr(\mathcal{L}')$ is also regular. So let the event structure be presented as $\mathcal{L} = pr(\mathcal{L}')$. Let $\mathcal{ES}_{\mathcal{L}} = (Ev, \preceq, \#, \eta)$ be the associated regular trace event structure. Note that $Ev = \mathcal{L}$.

A k-ary finitary trace relation is a k-ary relation over finite traces, $R \subseteq (\mathcal{T}r^f(\Sigma, I))^k$. A formula $\varphi(x_1, \ldots, x_k)$ defines a k-ary trace relation $R^{\mathcal{L}}_{\varphi}$ over $\mathcal{L}$: $(t_1, \ldots, t_k) \in R^{\mathcal{L}}_{\varphi}$ iff each t_i is in $\mathcal{L}$ and $\mathcal{ES}_{\mathcal{L}}$ satisfies φ under the valuation where each x_i is assigned the event $t_i \in Ev$. We will represent such relations defined by formulas using a trace language over a new alphabet.

Let $S = \{x_1, \ldots, x_k\}$ be a set of k variables, where $k \geq 1$. Then (Σ_S, I_S) is the trace alphabet where $\Sigma_S = \{(a, Y) \mid a \in \Sigma, \emptyset \neq Y \subseteq S\}$ and $I_S = \{((a, Y), (b, Y')) \mid aIb \text{ or } Y \cap Y' = \emptyset\}$.

Let $\bar{u} = \langle t_1, \ldots, t_k \rangle$ be a k-tuple of traces, with each $t_i = (E_i, \leq_i, \lambda_i)$. Without loss of generality, assume the event sets are disjoint and let $E = \bigcup_{i=1}^{k} E_i$. Define the following equivalence relation on E which relates two events if the traces they define are isomorphic: $\forall e, e' \in E, e \sim e'$ iff $e \in E_i, e' \in E_j$, for some i, j, and $\Downarrow_i e \approx \Downarrow_j e'$. Let $[\![e]\!]$ denote the equivalence class which e belongs to, according to the above relation. Note that if $e \sim e'$ with $e \in E_i$ and $e' \in E_j$, then $\lambda_i(e) = \lambda_j(e')$; further, if $i = j$, then $e = e'$. Hence the size of any equivalence class $[\![e]\!]$ is bounded by k.

Let $\bar{S} = \langle x_1, \ldots, x_k \rangle$ be an ordering of the variables in S. Then the tuple of traces $\bar{u}$ can be represented using the trace $br(\bar{u}, \bar{S}) = (F, \leq, \lambda)$ over (Σ_S, I_S) where

- $F = \{(a, [e]) \mid [e] \in E/\sim, e \in E_i, i \in [k], a = \lambda_i(max(e))\}$,
- $\leq$ is given by:
  $(a_1, [\![e_1]\!]) \leq (a_2, [\![e_2]\!])$ iff there is an $i \in [k]$ and events $e'_1 \in [\![e_1]\!]$ and $e'_2 \in [\![e_2]\!]$ such that $e'_1, e'_2 \in E_i$ and $e'_1 \leq_i e'_2$;
- the labelling function is given by:
  $\lambda(a, [\![e]\!]) = (a, Y)$ where $Y = \{x_i \mid \exists e' \in E_i, e' \in [\![e]\!]\}$.

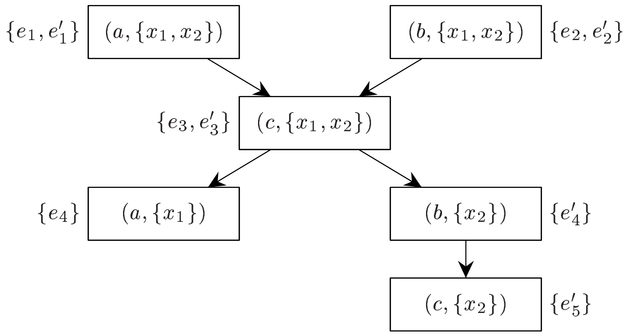

Figure 3. The braided trace $br(\langle t_1, t_2 \rangle, \langle x_1, x_2 \rangle)$

The function br hence braids a k-tuple of traces over (Σ, I) into a single trace over (Σ_S, I_S). Intuitively, the braiding of a tuple of traces merges these traces as much as possible while annotating the events to reflect which traces they came from.

For example, consider the two traces t_1 and t_2 over the alphabet $\{a, b, c\}$ with aIb, shown in Figure 1. Then Figure 3 represents the braided trace $br(\langle t_1, t_2 \rangle, \langle x_1, x_2 \rangle)$.

Traces formed by braiding a tuple of traces form a special class of traces over (Σ_S, I_S). Let us identify this class and its properties. A *braided trace* with respect to (Σ, I) and S is a trace over (Σ_S, I_S), $b = (F, \leq, \lambda)$, such that:

(BT) For every $f_1, f_2 \in F$, if $f_1 \leq f_2$ with $\lambda(f_1) = (a_1, Y_1)$ and $\lambda(f_2) = (a_2, Y_2)$, then $Y_2 \subseteq Y_1$.

(BT) says that in a braided trace, along a dependent chain of events, the second component is monotonically decreasing. The following is straightforward to prove:

Lemma 3.1 *Let $\bar{S} = \langle x_1, \ldots, x_k \rangle$ be a tuple of variables and b be a trace over (Σ_S, I_S). Then,*

- *If $\langle t_1, \ldots, t_k \rangle$ is a tuple of traces over (Σ, I), then $br(\bar{u}, \bar{S})$ is a braided trace with respect to (Σ, I) and S.*
- *If b is a braided trace, then there is a unique tuple $\langle t_1, \ldots, t_k \rangle$ of traces over (Σ, I) such that $b = br(\bar{u}, \bar{S})$.*

Let $B(\Sigma_S, I_S)$ denote the set of all braided traces with respect to (Σ, I) and S. Then since the property (BT) can be verified by a finite automaton, we have:

Proposition 3.2 *For any S, $B(\Sigma_S, I_S)$ is a regular trace language.*

A k-ary relation can thus be represented as a collection of braided traces: if $R \subseteq (\mathcal{T}r^f(\Sigma, I))^k$ and $\bar{S}$ is a k-tuple of variables, then let $br(R, \bar{S}) = \{br(\bar{u}, \bar{S}) \mid \bar{u} \in R\}$. Trace relations are thereby encoded as trace languages.

The following three relations capturing equality, prefix and conflict among traces will be useful. Also, note that the sets of braided traces corresponding to these relations are regular trace languages. In other words, if $\bar{S} = \langle x, x' \rangle$ is a pair of variables, then for each of the following relations R, $br(R, \bar{S})$ is regular. In each of the cases below, we take the intersection with an automaton that accepts $B(\Sigma_S, I_S)$, the set of braided traces over (Σ_S, I_S).

- $R_=$ is the equality relation, $\{([t], [t]) \mid [t] \in \mathcal{T}r(\Sigma, I)\}$. To show that $br(R_=, \bar{S})$ is regular, observe that the automaton working over Σ_S^* simply has to check if each letter is of the form (a, S).

- $R_\sqsubseteq$ is the prefix relation on traces, $\{([t], [t']) \mid [t] \sqsubseteq [t'], [t], [t'] \in \mathcal{T}r(\Sigma, I)\}$. The set $br(R_\sqsubseteq, \bar{S})$ is regular since the automaton working over Σ_S^* just has to check that each letter is of the form (a, Y), where $Y = \{x, x'\}$ or $Y = \{x'\}$.

- $R_{\not\uparrow}$ is the conflict relation over traces, $\{([t], [t']) \mid [t] \not\uparrow [t'], [t], [t'] \in \mathcal{T}r(\Sigma, I)\}$. To see that $br(R_{\not\uparrow}, \bar{S})$ is regular, observe that the automaton working over Σ_S^* can accept this by checking if there is an occurrence of a letter of the form $(a_1, \{x\})$ and an occurrence of a letter of the form $(a_2, \{x'\})$, for some $a_1, a_2 \in \Sigma$ such that $a_1 D a_2$.

Let $b = (F, \leq, \lambda)$ be a braided trace over (Σ_S, I_S). If $S' \subseteq S$, then b projected to S', denoted $b \downharpoonleft S'$, is the trace $(F', \leq', \lambda')$ where $F' = \{f \in F \mid \lambda(f) = (a, Y), Y \cap S' \neq \emptyset\}$, $\leq'$ is $\leq$ restricted to F', and for every $f \in F'$, $\lambda'(f) = (a, Y')$ where if $\lambda(f) = (a, Y)$ then $Y' = Y \cap S'$. Using the properties of braided traces, it is not hard to verify that $b \downharpoonleft S'$ is indeed a trace and is in fact a braided trace over $(\Sigma_{S'}, I_{S'})$. We extend the $\downharpoonleft$ operator to sets of braided traces. It is also straightforward to verify that if $\mathcal{B}$ is a regular set of braided traces over (Σ_S, I_S), then $\mathcal{B} \downharpoonleft S'$ is also regular (the automaton working over $\Sigma_{S'}^*$ guesses online a word whose projection to S' is the word being read; the automaton checks if this guessed word is a linearization of $\mathcal{B}$).

Let us also define an "expansion" operator. Let $b = (F, \leq, \lambda)$ be a braided trace over (Σ_S, I_S) and let S' be a finite superset of S. Then $b \upharpoonleft S'$ is the set of all braided traces b' over $(\Sigma_{S'}, I_{S'})$ such that $b' \downharpoonleft S = b$. Again, we extend this to languages by defining $\mathcal{B} \upharpoonleft S' = \bigcup_{b \in \mathcal{B}} (b \upharpoonleft S')$. Observe that if $\mathcal{B}$ is regular, then so is $\mathcal{B} \upharpoonleft S'$ (the automaton working on $\Sigma_{S'}^*$ checks if the input word is a braided trace and also checks if the linearization of its projection to S is in $\mathcal{B}$.)

We are now ready to prove the main theorem.

Theorem 3.3 *The model-checking problem for regular trace event structures against* FOL *specifications is decidable.*

Proof: Let $\mathcal{L}$ be the finitely presented trace language over (Σ, I) and let $\mathcal{ES}_\mathcal{L} = (Ev, \preceq, \#, \eta)$ be the corresponding regular trace event structure. Let φ_0 be the given FOL formula. In the sequel, to make the notation and proof uniform, whenever we consider a subformula φ of φ_0, we include a special variable x (which does not occur in φ_0 at all) as a free-variable in φ. This is useful as it even makes a formula φ with with no free variables define a 1-ary relation—this monadic relation contains all the traces in $\mathcal{L}$ if φ is true in the event structure and is the empty relation if φ is false.

For any subformula $\varphi(x_1, \ldots, x_k)$ of φ_0, if the induced relation over traces is $R_\varphi^\mathcal{L}$ and $\bar{S} = \langle x_1, \ldots, x_k \rangle$, then let $\mathcal{B}_{\varphi,\bar{S}}^\mathcal{L}$ denote the set $br(R_\varphi^\mathcal{L}, \bar{S})$, which is a set of braided traces over (Σ_S, I_S). We show the following claim:

Claim 3.4 *Let $\varphi(x_1, \ldots, x_k)$ be a subformula of φ_1 and let $\bar{S} = \langle x_1, \ldots, x_k \rangle$. Then $\mathcal{B}_{\varphi,\bar{S}}^\mathcal{L}$ is a regular trace language.*

Proof of claim: First note that if $R_k = \{\langle t_1, \ldots, t_k \rangle \mid \forall i \in [k], t_i \in \mathcal{L}\}$, which is the k-ary relation relating any k-tuple of traces as long as they are in $\mathcal{L}$, then for any $\bar{S} = \langle x_1, \ldots, x_k \rangle$, $br(R_k, \bar{S})$ is regular (the automaton just has to check that each t_i is in $\mathcal{L}$). Let us denote this language as $\mathcal{B}_{\bar{S}}$.

We show the claim by structural induction over the subformulas of φ_0. For the base cases, note that if $\bar{S} = \langle x, x_1, x_2 \rangle$ and $\varphi(x, x_1, x_2)$ is of the form $(x_1 = x_2)$, $(x_1 \preceq x_2)$ or $(x_1 \# x_2)$, then $\mathcal{B}_{\varphi,\bar{S}}^\mathcal{L} = (br(R, \langle x_1, x_2 \rangle) \cap \mathcal{B}_{\langle x_1, x_2 \rangle}) \upharpoonleft S$, where R is $R_=$, $R_\sqsubseteq$ or $R_{\not\uparrow}$, respectively. Hence $\mathcal{B}_{\varphi,\bar{S}}^\mathcal{L}$ is regular.

If $\varphi(\bar{S}) = \neg\varphi_1(\bar{S})$, then $\mathcal{B}_{\varphi,\bar{S}}^\mathcal{L} = (\mathcal{T}r^f(\Sigma_S, I_S) \setminus \mathcal{B}_{\varphi_1,\bar{S}}^\mathcal{L}) \cap \mathcal{B}_{\bar{S}}$, and is hence regular.

Now, let $\bar{W}$, $\bar{Y}$, and $\bar{Z}$ be disjoint sets of variables and let $\varphi(\bar{W}, \bar{Y}, \bar{Z}) = \varphi_1(\bar{W}, \bar{Y}) \wedge \varphi_2(\bar{W}, \bar{Z})$. Let $\bar{S} = \bar{W}\bar{Y}\bar{Z}$ be the concatenation of these tuples. Then $\mathcal{B}_{\varphi,\bar{S}}^\mathcal{L} = (\mathcal{B}_{\varphi_1,\bar{W}\bar{Y}}^\mathcal{L} \upharpoonleft S) \cap (\mathcal{B}_{\varphi_2,\bar{W}\bar{Z}}^\mathcal{L} \upharpoonleft S)$, and is again regular.

Finally, let $\varphi(\bar{S}) = \exists x_1 . \varphi_1(x_1, \bar{S})$. Let $\bar{Y} = x_1.\bar{S}$. Then $\mathcal{B}_{\varphi,\bar{S}}^\mathcal{L} = \mathcal{B}_{\varphi_1,\bar{Y}}^\mathcal{L} \downharpoonleft S$, and is therefore regular.

End of claim.□

Hence, for $\varphi_0(x)$, using the claim, we get that $\mathcal{B}_{\varphi_0,\{x\}}^\mathcal{L} \neq \emptyset$ iff $\mathcal{ES}_\mathcal{L} \models \varphi$. By checking for emptiness of the automaton accepting $\mathcal{B}_{\varphi_0,\{x\}}^\mathcal{L}$, we can decide whether $\mathcal{ES}_\mathcal{L}$ satisfies φ. □

4. Model-checking MTL properties

In this section, we show how to model check regular trace event structures against MTL specifications. Recall that in MTL one can quantify over sets of events, but any set of events that is quantified must be conflict-free. It is easy to see that if F is a conflict-free set of events of an event structure, then $\bigcup_{f \in F} \downarrow F$ is a subset of events that corresponds to a trace t_F, perhaps an infinite one, that contains all the events in F as subtraces.

In the decidability proof for FOL, recall that each first-order variable quantified over an event, which was a trace in $\mathcal{L}$. The crux of the decidability proof for MTL is that each second-order variable quantifies over a set of conflict free events that can be represented by an infinite trace (with additional markings).

A subset of finite and infinite words, $L \subseteq \Sigma^\infty$ is said to be a regular mixed language if the set of finite words in L is a regular subset of Σ^* and the set of ω-words in L is a regular subset of Σ^ω. (Regular subsets of Σ^ω is the usual notion [18].) Note that mixed regular languages are closed under union, intersection and complement (since regular subsets of Σ^* and Σ^ω are). A set of finite and infinite traces, called a mixed set of traces, $\mathcal{L}' \subseteq \mathcal{T}r(\Sigma, I)$ is said to be regular if $lin(\mathcal{L}')$ is a regular subset of Σ^∞. Mixed regular trace languages are hence also closed under union, intersection and complement.

Let $\mathcal{L} \subseteq \mathcal{T}r^f(\Sigma, I)$ be the given regular language of finite traces, which is $\sqsubseteq$-closed and which defines the event structure $\mathcal{ES}_{\mathcal{L}}$. Let us assume, in order to simplify the presentation, that for every trace $t \in \mathcal{L}$, there is an extension t' of it (i.e. $t \sqsubseteq t', t \neq t'$), which is also in $\mathcal{L}$.

Now, let $\Sigma' = \Sigma \times \{\top, \bot\}$ and let $I' = \{((a,r),(a',r')) \mid aIa',\ r,r' \in \{\top,\bot\}\}$. If v is a trace over (Σ', I'), let $\lfloor v \rfloor$ be the trace obtained by replacing every label of the form (a,r) by a — $\lfloor v \rfloor$ is then a trace over (Σ, I). Let $lim(\mathcal{L})$ be the collection of *infinite traces* v over (Σ', I') such every finite prefix of $\lfloor v \rfloor$ is in $\mathcal{L}$. Intuitively, $lim(\mathcal{L})$ is just obtained by taking the limit of traces in $\mathcal{L}$ and tagging a $\top/\bot$ symbol to each label.

The point now is that every trace v in $lim(\mathcal{L})$ denotes a set of prime traces $Set(v)$ in $\mathcal{L}$: a prime trace t is in $Set(v)$ iff there is a prime trace v_1 over (Σ', I') such that $v_1 \sqsubseteq v$, the maximal event of v_1 is of the form $(a, \top)$ (for some $a \in \Sigma$), and $\lfloor v_1 \rfloor$ is isomorphic to t. Intuitively, $Set(v)$ denotes the set of prime traces which are prefixes of $\lfloor v \rfloor$ that are tagged with the $\top$ symbol.

Also, it is easy to see that if F is any set of conflict-free set of prime traces in $\mathcal{L}$, then there is a trace $v \in lim(\mathcal{L})$ such that $Set(v) = F$.

We now define relations involving both events and conflict-free sets of events. A formula $\varphi(x_1, \ldots, x_k, X_{k+1}, \ldots, X_{k+l})$ defines a relation $R_\varphi \subseteq (\mathcal{T}r^f(\Sigma, I))^k \times (\mathcal{T}r^\omega(\Sigma', I'))^l$, in the obvious way, given the interpretation above of how infinite traces over (Σ', I') represent conflict-free sets of events.

We need to define braiding of a tuple of finite and infinite traces. Let $S_x = \{x_1, \ldots, x_k\}$ be a set of first-order variables, $S_X = \{X_{k+1}, \ldots, X_l\}$ a set of second-order variables and let $S = S_x \cup S_X$. Let $\mathcal{G}_Y$, where $Y \subseteq S_X$, denote the set of functions from Y to $\{\top, \bot\}$. Then, define $\Sigma_S = \{(a, Z, g) \mid a \in \Sigma, Z \subseteq S, g \in \mathcal{G}_{Z \cap S_X}\}$ and let the independence relation be defined as: $(a_1, Z_1, g_1) I (a_2, Z_2, g_2)$ iff $a_1 I a_2$ or $Z_1 \cap Z_2 = \emptyset$. This is an extension of the definition of Σ_S and I_S in the previous section where it was defined when S was a set of first-order variables.

Let $\bar{u} = \langle t_1, \ldots t_k, v_{k+1}, \ldots, v_l \rangle$ be a tuple of traces where each $t_i = (E_i, \leq_i, \lambda_i)$, $i \in \{1, \ldots, k\}$, is a finite trace over (Σ, I) and each $v_j = (E_j, \leq_j, \lambda_j)$, $j \in \{k+1, \ldots, l\}$, is an infinite trace over (Σ', I'). Let the event sets be disjoint and let $E = \bigcup_{i=1}^{l} E_i$. Let us extend the definition of $\lfloor t \rfloor$ to the case when t is a trace over (Σ, I) by defining $\lfloor t \rfloor = t$. Now define the following relation on E: $e \sim' e'$ iff $\lfloor \downarrow e \rfloor \approx \lfloor \downarrow e' \rfloor$, i.e. two events e and e' are related if $\downarrow e$ and $\downarrow e'$ are isomorphic, modulo the $\{\top, \bot\}$-labels. Let $[\![e]\!]$ denote the equivalence class to which e belongs under the above equivalence. Let $\bar{S} = \langle x_1, \ldots, x_k, X_{k+1}, \ldots X_l \rangle$ be an ordered set of first-order and second-order variables. Then we define the braiding of the tuple $\bar{u}$, $br(\bar{u}, \bar{S})$ as $(F, \leq, \lambda)$ where:

- $F = \{(a, [\![e]\!]) \mid e \in E_i, i \in [l]$, and either $\lambda_i(e) = a$ or $\lambda_i(e) = (a, r)$, for some $r \in \{\top, \bot\}$
- $\leq$ is given by:
  $(a_1, [\![e_1]\!]) \leq (a_2, [\![e_2]\!])$ iff there is an $i \in [l]$, $e_1', e_2' \in E_i$ such that $e_1' \in [\![e_1]\!]$, $e_2' \in [\![e_2]\!]$ and $e_1 \leq_i e_i'$.
- λ is defined as follows:
  $\lambda((a, [\![e]\!])) = (a, Z, g)$ where $Z = \{x_i \mid \exists e' \in E_i, i \in \{1, \ldots, k\}, e' \in [\![e]\!]\} \cup \{X_i \mid \exists e' \in E_i, i \in \{k+1, \ldots, l\}, e' \in [\![e]\!]\}$ and for any $X_j \in Z \cap S_X$, $g(X_j) = r$, if the event $e' \in E_j$ such that $e' \in [\![e]\!]$ in v_j is labelled (a, r).

The definition of braided traces with respect to (Σ, I) and S is also extended to be any trace, finite or infinite, over (Σ_S, I_S) that satisfies:

(BT') For every $f_1, f_2 \in F$, if $f_1 \leq f_2$ with $\lambda(f_1) = (a_1, Z_1, g_1)$ and $\lambda(f_2) = (a_2, Z_2, g_2)$, then $Z_2 \subseteq Z_1$.

Again, we can show that $br(\bar{u}, \bar{S})$ is always a braided trace and every braided trace represents a unique tuple $\bar{u}$ of finite and infinite traces. Also, the set of all braided traces with respect to (Σ, I) is a mixed regular trace language. We hence represent relations over finite and infinite traces as a collection of braided traces.

We also extend the notion of projection, $\downharpoonleft$, and expansion, $\upharpoonleft$, in the obvious way to infinite braided traces. Again, regularity is preserved under projection and expansion.

The proof of decidability of model-checking against MTL specifications is similar to that for FOL. We show, by induction, that if $\varphi(x_1, \ldots, x_k, X_{k+1}, \ldots, X_l)$ is any subformula of the specification, then this defines a relation over a tuple of traces—finite traces over (Σ, I) and infinite traces over (Σ', I'). We then show that the braided traces that encode this relation is a mixed regular trace language.

The only additional thing we need to do is to handle the atomic formula $x \in X$. This can be done as follows: an automaton reading the braided trace can verify that if the maximal event in the trace with a label of the kind (a, Z, g) such that $x \in Z$ is e, then e has a label (a', Z', g') where $X \in Z'$ and $g(X) = \top$.

Existential set quantification can be handled in a similar manner as first-order quantification, since regular braided traces are closed under projection. The problem again reduces to checking the emptiness of automata and we have:

Theorem 4.1 *The model-checking problem for regular trace event structures against* MTL *specifications is decidable.*

5. Extensions

In the FOL decidability result, note that for the proof to go through, one just needed that the set of traces representing events is regular and that the relations corresponding to atomic formulas are such that the braided language of traces corresponding to them are regular. We can hence generalize this to any class of structures that have the above properties.

Let $\mathcal{U} = (U, R_1, \ldots, R_l)$ be any structure where U is the universe and each R_i is a k_i-ary relation over U. We say that $\mathcal{U}$ is a *trace automatic structure* if there is a distributed alphabet (Σ, I), a regular trace language $\mathcal{L}$ over (Σ, I) ($\mathcal{L}$ need not contain only prime traces) and languages $\mathcal{R}_1, \ldots \mathcal{R}_l$ where each $\mathcal{R}_i$ is a regular trace language over (Σ_{S_i}, I_{S_i}) (where each S_i is a set of k_i variables $\{x_1, \ldots, x_{k_i}\}$ and (Σ_{S_i}, I_{S_i}) is defined as in Page 6), and also there is a 1-1 correspondence $f : U \to \mathcal{L}$ such that

- for every $i \in [l]$, for every $\langle u_1, \ldots, u_{k_i}\rangle \in U^{k_i}$, $\langle u_1, \ldots, u_{k_i}\rangle \in R_i$ iff $br(\langle u_1, \ldots, u_{k_i}\rangle, \langle x_1, \ldots, x_{k_i}\rangle) \in \mathcal{R}_i$

A trace automatic structure can hence be presented by giving the languages $\mathcal{L}$ and $\mathcal{R}_1, \ldots, \mathcal{R}_l$. Then, using a proof similar to that of Theorem 3.3, one can show that given a trace automatic structure and an FOL formula φ, one can decide whether φ holds in the structure. (A similar generalization of MTL seems awkward as quantification over conflict-free traces, though natural for event structures, is not natural in a general setting).

Note that it is known that automatic structures (where elements are represented by finite words) have a decidable FOL theory [1]. Actually, it turns out that regular trace event structures are in fact automatic structures. However, we have chosen not to present them in this manner because the encoding to make them automatic turns out to be awkward and complex. In particular, encoding each event by a (canonical) linearization of the trace representing the event does not make the structures automatic. We believe that using traces to represent elements is a much more clean and intuitive way to represent structures involving traces. Also, we do not know whether the class of trace automatic structures is larger than the class of automatic structures, but we conjecture that it is not.

Another point to note is that in the logic MTL, we can write properties of the form $AG\ EF\ \varphi$, as mentioned in the introduction, where φ is a *monadic second-order formula over traces*. The reason is that once we quantify over a trace using a variable X in MTL, we can quantify over arbitrary subsets of it as they are bound to be conflict-free.

6. Conclusions

We have shown that FOL and MTL are both decidable logics for model-checking regular trace event structures. We believe that MTL or a variant of it is likely to be powerful enough to capture any natural branching time logic over event structures. The notion of regular trace relations introduced here is also of independent interest.

We can easily extend the logics to handle an extra level of labelling of the events, where the labelling is an arbitrary function $\gamma : \Sigma \to \Pi$. Doing this would allow us, for example, to model-check event structures corresponding to labelled finite 1-safe Petri nets.

Note that we have not considered the *satisfiability* problem for any of these logics. There are modal logics over event structures studied in the literature for which satisfiability is decidable. However, we can show, using a variant of the MSOL model-checking undecidability result (Proposition 2.1), that the satisfiability problem for trace event structures for FOL specifications is already undecidable.

As for complexity, our procedures have non-elementary time complexity. While it is clear that we cannot do better than this in terms of the dependence on the size of the formula, we do not know the exact complexity of the model-checking problem when the formula is *fixed*.

One important aspect of our proof of FOL is that it is "symbolic" as it uses finite automata to capture the relations of FOL-definable relations. Since these automata can be minimized each time, it is possible that the automata stay small in size in practice and leads to efficient decision procedures. The tool Mona [4] has had considerable success using such minimization of automata and it would be interesting to see how this fares in our setup.

A related area is that of *unfoldings of Petri nets* [10]. Unfoldings are quite similar to the event structures associated with a Petri net and model-checking unfoldings of Petri-nets using logics has been studied [20]. It would be interesting to find their counterparts in the event structure world and compare them to FOL and MTL.

We can also see our results being related to decidable theories for *trace rewriting systems*. In [8], the authors show that the FOL theory of *one-step* trace rewriting is decidable; this does not however encompass our FOL decidability result as the one-step rewrite rule cannot capture the prefix relation $\sqsubseteq$ on traces.

In this work, we have reduced the expressive power of the logic to get decidability. Another approach is to restrict the class of event structures to get MSOL-decidability. Thiagarajan has recently identified a class of event structures (those arising from communicating cyclic processes), which can be interpreted on trees and hence have a decidable MSOL model-checking problem [17].

Finally, from a practical point of view, it would be interesting to come up with a simple branching-time temporal logic on event structures akin to CTL over trees and, perhaps using the symbolic decision procedure presented here, try to develop efficient model-checking procedures for it.

Acknowledgements:

I would like to thank Wolfgang Thomas and P.S. Thiagarajan for interesting discussions that helped shape this paper.

References

[1] A. Blumensath and E. Grädel. Automatic structures. In *15th Annual IEEE Symposium on Logic in Comp. Sc.*. IEEE Computer Society, 2000.

[2] V. Diekert and G. Rozenberg, editors. *Book of Traces*. World Scientific, Singapore, 1995.

[3] W. Ebinger and A. Muscholl. Logical definability of infinite traces. In *Proc. of ICALP*, volume 700 of *LNCS*. Springer, 1993.

[4] J. Elgaard, N. Klarlund, and A. Möller. Mona 1.x: New techniques for WS1S and WS2S. In *CAV, Proc. 10th Int. Conference*, volume 1427 of *LNCS*, pages 516–520. Springer, Berlin, 1998.

[5] C. Frougny and J. Sakarovitch. Synchronized rational relations of finite and infinite words. *Theoretical Comp. Sc.*, 108:45–82, 1993.

[6] T. Hafer and W. Thomas. Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree. In *Proc. 14th ICALP*, volume 267 of *LNCS*, pages 269–279. Springer, 1987.

[7] G.J. Holzmann. The model checker SPIN. *IEEE Trans. on Software Engg.*, 23(5):279–295, May 1997. Special issue on Formal Methods in Software Practice.

[8] D. Kuske and M. Lohrey. On the theory of one-step rewriting in trace monoids. In *Proc. of ICALP*, volume 2380 of *LNCS*. Springer, 2002.

[9] K. Lodaya, R. Ramanujam, and P.S. Thiagarajan. Temporal Logics for Communicating Sequential Agents: I. In *International Journal of Foundations of Computer Science*, Vol. 3 (2):117–159, 1992.

[10] K.L. McMillan. Using unfolding to avoid the state explosion problem in the verification of asynchronous circuits. In *Proc. 4th CAV*, volume 663 of *LNCS*, pages 164–174, 1992. Springer.

[11] M. Mukund and P.S. Thiagarajan. A logical characterization of well branching event structures. *Theoretical Comp. Sc.*, 96(1):35–72, 1992.

[12] M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, Part 1. *Theoretical Comp. Sc.*, 13(1):85–108, 1981.

[13] M. Nielsen and P.S. Thiagarajan. Regular event structures and finite petri nets: The conflict-free case. In *Proc., ICATPN*, volume 2360 of *LNCS*, 2002.

[14] W. Penczek. Branching time and partial order in temporal logics. In L. Bolc and A. Szalas, eds., Time and Logic: A Computational Approach, pages 179–228. UCL Press, 1995.

[15] W. Penczek. Model-checking for a subclass of event structures. In *Proc. of TACAS, (TACAS)*, volume 1217 of *LNCS*. Springer, 1997.

[16] P.S. Thiagarajan. Regular trace event structures. Technical Report RS-96-32, BRICS Tech report, Denmark, 1996. 34 pp.

[17] P.S. Thiagarajan. *Private communication*, 2003.

[18] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 165–191, 1990.

[19] W. Thomas. A short introduction to infinite automata. In *Proc. of DLT, 2001*, volume 2295 of *LNCS*. Springer, 2001.

[20] F. Wallner, M. Huhn, P. Niebert. Verification based on local states. In *Proc. of TACAS*, volume 1384 of *LNCS*. Springer, 1998.

[21] Glynn Winskel and Mogens Nielsen. Models for concurrency. *Handbook of logic in comp. sc. (vol. 4)*, pages 1–148, 1995.