Electronic Communications of the EASST
Volume 47 (2012)

Proceedings of the 11th International Workshop on Graph Transformation and Visual Modeling Techniques (GTVMT 2012)

Model Checking Communicating Processes: Run Graphs, Graph Grammars, and MSO

Alexander Heußner

15 pages

Guest Editors: Andrew Fish, Leen Lambers
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Model Checking Communicating Processes: Run Graphs, Graph Grammars, and MSO

Alexander Heußner

Université Libre de Bruxelles – Belgium

Abstract: The formal model of recursive communicating processes (RCPS) is important in practice but does not allows to derive decidability results for model checking questions easily. We focus a partial order representation of RCPS's execution by graphs—so called *run graphs*, and suggest an under-approximative verification approach based on a bounded-treewidth requirement. This allows to directly derive positive decidability results for MSO model checking (seen as partial order logic on run graphs) based on a context-freeness argument for restricted classes run graphs.

Keywords: Pushdown Systems, Hyperedge Replacement Grammars, MSO

Communicating ProcesseS (CPS) allow to model an important class of distributed processes that asynchronously communicate, e.g., via TCP-based channels or MPI, over a network topology. Hence, synchronization takes place on unbounded, lossless fifo channels. As CPS subsume communicating finite-state machines, they inherit their well-known undecidability results, e.g., for reachability [BZ83]. Local infinite data and unbounded recursion suggests to model each process by a finite control flow automaton with a pushdown stack leading to *recursive CPS* (RCPS). The latter allow to reduce the Post correspondence problem or the emptiness-testing of the intersection of two context free languages to RCPS's reachability question [Ram00], and thus give rise to a second source of undecidability. Hence, classical verification questions become a challenging problem on these models, especially when leaving behind simple reachability questions.

Our Contribution Extending positive decidability results from the (control-state) reachability problem of a (semantically) restricted class of RCPS introduced in [HLMS11] to the model checking problem of linear temporal logics (LTL) proves to be impossible [Ram00], as one can (ab)use LTL interpreted on an interleaving run to locally synchronize events that take place on different processes and that are a priori independent. Thus, we propose to leave CPS's interleaving-based semantics behind. We represent the underlying partial order of events of a CPS's (finite) execution as graphs—so called *run graphs*—and interpret monadic second order logic (MSO) thereupon. However, it is well-known that MSO is undecidable on general graphs. We avoid this intricacy by extending some of our previous insights, e.g., the context-freeness of runs of eager RCPS on non-confluent architectures [HLMS11], to the graph setting. We show that one can represent the run graphs of several (semantically) restricted classes of finite CPS/RCPS with the help of context-free graph grammars, in our case *hyperedge replacement grammars* [DKH97]. As graphs generated by these grammars have bounded treewidth, we can easily transfer the fundamental positive decidability result on MSO model checking of bounded-treewidth graphs [CE11] to our setting. The latter was already proposed in [Heu10] but is first detailed in this article. This gives rise to a new way of approaching CPS from a partial-order perspective, and proposes an unifying view on known semantic boundedness conditions via graph measures.

Overview In the following section, we introduce the formal model of (R)CPS as well as give an overview of known restrictions and decidability results for a suite of reachability questions. Next, we recall some notions on MSO interpreted on graphs, and hyperedge replacement grammars, before introducing run graphs. Last, we give decidability results for MSO interpreted on run graphs of two important classes of (R)CPS. We refer to the appendix for additional details/proofs.

Notation Given a set S, let $|S|$ denote its cardinality. For an I-indexed family of sets $(S^i)_{i\in I}$, we write elements of $\prod_{i\in I} S^i$ in bold face, i.e., $\mathbf{s} \in \prod_{i\in I} S^i$. The i-component of $\mathbf{s}$ is written $s^i \in S^i$, and we identify $\mathbf{s}$ with the indexed family of elements $(s^i)_{i\in I}$. An *alphabet* Σ is a finite set of *letters*. We write Σ^* for the set of all *finite words* over Σ. Let ε denote the empty word. Let $w = w_1 w_2 \cdots \in \Sigma^*$, then its *forgetful projection* on $\Gamma \subseteq \Sigma$ is the word $w' = w'_1 w'_2 \ldots$ with $w_i = w'_i$ if $w_i \in \Gamma$, else $w_i = \varepsilon$ (for $i \in \{1,2,\ldots\}$). A *labeled transition system* (LTS) $\mathcal{S} = \langle S, s_0, Act, \to\rangle$ is given by a set of *states* S, an initial state s_0, an *action* alphabet Act, and a (labeled) *transition relation* $\to \subseteq S \times Act \times S$. We usually write $s \xrightarrow{a} s'$ in place of $(s,a,s') \in \to$. A *pushdown system* is given by a tuple $\langle Z, z_0, Act, \Gamma, \Delta\rangle$ where Z is a finite set of (control) states, $z_0 \in Z$ the initial state, Γ a finite stack alphabet, and $Act = \{push(\gamma), pop(\gamma) | \gamma \in \Gamma\}$ the set of pushdown actions where transition rules are given by $\Delta \subseteq Z \times Act \times Z$. The semantics of a pushdown system is given by an infinite transition system over configurations in $Z \times \Gamma^*$, i.e., tuples of a control state and a word representation of the stack's content. Note that a $push(\gamma)$ action adds the letter γ to the top of the current stack content, and that a $pop(\gamma)$ action blocks if the topmost letter is not equal to γ.

1 Verifying (Recursive) Communicating Processes

A *topology* $\mathcal{T}$ is a directed graph whose vertices are processes $Proc$ and whose edges are channels Ch. The set of *possible communication actions* of a process $p \in Proc$ are defined by $Com^p(\mathcal{T}, M) = \{c!m \mid c \in Ch, src(c) = p, m \in M\} \cup \{c?m \mid c \in C, dst(c) = p, m \in M\}$. We denote by $c!m$ the sending of message m into channel c, and by $c?m$ the reception of message m from c.

Definition 1 A *system of communicating processes* (CPS) $\mathcal{Q} = \langle \mathcal{T}, M, (\mathcal{S}^p)_{p\in Proc}\rangle$ is given by a topology $\mathcal{T}$, a *message* alphabet M, and an LTS $\mathcal{S}^p$ for each process $p \in Proc$ of the form $\langle S^p, s_0^p, Act^p, \to^p\rangle$ such that:

- the action alphabets Act^p, $p \in Proc$, are pairwise disjoint, and
- $Act^p_{com} = Act^p \cap (C \times \{!,?\} \times M) \subseteq Com^p(\mathcal{T}, M)$ for each $p \in Proc$.

States $s^p \in S^p$ are called *local states* of p. We write $S = \prod_{p\in Proc} S^p$ for the set of *global states*. Note that the sets S^p, and hence S, may be infinite. A *finite* CPS is a CPS where for all $p \in Proc$ the S^p are finite.

The operational semantics of $\mathcal{Q}$ is defined by a global LTS $[\![\mathcal{Q}]\!] = \langle X, x_0, Act, \to\rangle$, where $X = S \times (M^*)^C$ is the set of *configurations*, $x_0 = (\mathbf{s_0}, (\varepsilon)_{c\in Ch})$ is the initial configuration (note that channels are initially empty), $Act = \bigcup_{p\in Proc} Act^p$ is the set of actions, and $\to \subseteq X \times Act \times X$ is

the transition relation; the latter is defined as follows: for $a \in Act^p$, $(\mathbf{s_1}, \mathbf{w_1}) \xrightarrow{a} (\mathbf{s_2}, \mathbf{w_2})$ if the subsequent conditions are satisfied:

(i) $s_1^p \xrightarrow{a}{}^p s_2^p$ and $s_1^q = s_2^q$ for all $q \in Proc$ with $q \neq p$,

(ii) if $a \in Act_{loc}^p$ then $\mathbf{w_1} = \mathbf{w_2}$,

(iii) if $a = c!m$ then $w_2^c = w_1^c \cdot m$ and $w_2^d = w_1^d$ for all $d \in C$ with $d \neq c$,

(iv) if $a = c?m$ then $m \cdot w_2^c = w_1^c$ and $w_2^d = w_1^d$ for all $d \in C$ with $d \neq c$.

A (finite) *run* in the LTS $\mathcal{Q}$ is an alternating sequence $\rho = (x_0, a_1, x_1, \ldots, a_n, x_n)$ of configurations $x_i \in X$ and actions $a_i \in Act$ that satisfies, for all $1 \leq i \leq n$, $x_{i-1} \xrightarrow{a_i} x_i$. A configuration $x \in X$ is *reachable* in a CPS $\mathcal{Q}$ if there exists a finite run of $\mathcal{Q}$ from the initial configuration x_0 to x. We define the *reachability set* of $\mathcal{Q}$ as $Reach(\mathcal{Q}) = \{x \in X \mid x \text{ is reachable in } \mathcal{Q}\}$. For a given CPS $\mathcal{Q}$ and a global state $\mathbf{s} \in S$, the *state reachability problem* for CPS asks whether $Reach(\mathcal{Q}) \cap \{\mathbf{s}\} \times (M^*)^C$ is non-empty.

A pair of send/receive actions $a_i = c!m, a_j = c?m$ is called *matching* in ρ if $i < j$ and the number of receives on c within $a_i \cdots a_j$ equals the length of c in x_i. In the following we will mainly focus runs of a certain form, so called *eager runs*.

Definition 2 A run $\rho = (x_0, a_1, x_1, \ldots, a_n, x_n)$ is *eager* if for all a_i with $1 \leq i \leq n$ when a_i is a receive action then $i > 1$ and a_{i-1} is its matching send action.

A configuration $x \in X$ is *eager-reachable* in a CPS $\mathcal{Q}$ if there exists a finite, eager run from the initial configuration x_0 to x. The *eager-reachability set* of $\mathcal{Q}$ is the set $Reach_{eager}(\mathcal{Q})$ of eager-reachable configurations. We say that a CPS $\mathcal{Q}$ is *eager* when $Reach_{eager}(\mathcal{Q}) = Reach(\mathcal{Q})$. One can show that eagerness of CPS arises under some natural (and decidable) restriction on cyclic communication, called *mutex*; the latter is a generalization of the well-known half-duplex restriction from mutual channels to larger topologies. Deciding the eagerness of a finite CPS proves to be undecidable, however, deciding whether a given finite CPS is mutex is complete in PSPACE [HLMS11]. Eager CPS subsume globally existentially 1-bounded finite-state machines [LM04].

Recursive communicating processes (RCPS) $\mathcal{R} = \langle \mathcal{T}, M, (\mathcal{D}^p)_{p \in Proc} \rangle$ are CPS whose local transition systems $\mathcal{D}^p$ are pushdown systems. Formally, each $\mathcal{D}^p$ is given by $\langle Z^p, z_0^p, Act^p, \Gamma^p, \Delta^p \rangle$ where Z^p is a finite set of control states, z_0^p an initial state, Act^p an alphabet of actions, Γ^p a stack alphabet and $\Delta^p \subseteq Z^p \times Act^p \times Z^p$ the set of actions.

The (local) semantics of each $\mathcal{D}^p$ is the common one and we assert the stack operations to be expressed by $Act_{stack}^p = \{push(\gamma), pop(\gamma) \mid \gamma \in \Gamma^p\} \subseteq A_{loc}^p$; local configurations are written $(z^p, u^p) \in Z^p \times (\Gamma^p)^*$. Note that each *pop* action must have a *matching push* action before, such that the forgetful projection of a local run of $\mathcal{D}^p$ on Act_{stack}^p is a Dyck word, i.e., a word where the $push(\gamma)$ and $pop(\gamma)$ actions for $\gamma \in \Gamma$ are "well-nested".

Let $Z = \prod_{p \in Proc} Z^p$ the set of *global control states*. A *state* of $\mathcal{R}$ is written $s = (\mathbf{z}, \mathbf{u})$ where $s^p = (z^p, u^p)$ for each $p \in Proc$. For a given RCPS $\mathcal{R}$ and a global control state $\mathbf{z} \in Z$, the *state reachability problem* for RCPS asks whether $Reach(\mathcal{R}) \cap \{\mathbf{z}\} \times (\prod_{p \in Proc}((\Gamma^p)^*)) \times (M^*)^C$ is empty. We recall the following decidability results for (R)CPS:

Fact 1 *The state reachability problem is undecidable for finite* CPS*, even if we restrict the topology to two processes that are mutually connected by two channels. [BZ83]*

Fact 2 *The state eager-reachability problem for finite* CPS *is decidable. [HLMS11]*

Fact 3 *The* RCPS *state eager-reachability problem is undecidable when the underlying topology contains at least two pushdown processes that are connected by one channel. [HLMS11]*

Even if eagerness is sufficient for arriving at positive results for finite CPS, we need additional architectural constraints to gain the following positive reachability result for RCPS.

Fact 4 *A typed topology has a deciable* RCPS *state eager-reachability problem if and only if it is non-confluent. The problem is* EXPTIME*-complete in the latter case. [HLMS11]*

The previous result considers a semantic restriction against the direct synchronization of a RCPS's local pushdown systems, inspired by, e.g., [LMP08, CV09]. A *typed topology* restricts the usage of a (point-to-point) channels as follows: either the process sending on this channel can use its channel end only when its local stack empty, or the receiving process can read only when its local stack empty, or both of the previous two restrictions (i.e., sending and receiving requires the active process's local stacks to be empty).

A typed topology is *confluent* if there exists a sequence $(p_0, c_1, p_1, ..., c_n, p_n)$ of distinct processes $p_i \in Proc$ (for $0 \leq i \leq n$) and channels $c_j \in Ch$ ($1 \leq j \leq n$) such that p_0 and p_n can use their channel ends of c_1 and c_n without restriction. A *non-confluent* typed topology is not confluent. Informally, non-confluent topologies do not allow to let the information generated/read by two pushdown systems to "flow together".

The proof of Fact 4 relies on the following: for each eager run of a non-confluent RCPS one can efficiently construct an order-equivalent run whose forgetful projection onto Act_{stack} is globally a Dyck word, i.e., which can be simulated on *one* exponentially larger global pushdown process. We capture the latter by the following definition.

Definition 3 A finite run ρ of an RCPS is *one-stack* if the forgetful projection of $a'_1 \cdots a'_n$ on $\bigcup_{p \in P} Act^p_{stack}$ is a Dyck word.

Lemma 1 *All runs of eager* RCPS *over non-confluent architectures are one-stack.* □

Lemma 2 *Given an* RCPS*, it is undecidable whether* all *its runs are one-stack.* □

Remark 1 *Directly extending the previous positive results from safety/reachability verification to the model checking of ω-regular properties is however not possible. Let us take a look at an* RCPS *consisting of two local pushdown process with no channel in between, hence, an* RCPS *on a non-confluent architecture where all runs are eager. Interpreting linear temporal logics (LTL) on runs of this* RCPS *allows to express the synchronization of two pushdown processes via a logical formula [Ram00]. Restricting the logic to a "weaker" (from the standpoint of expressiveness) fragment does not help either, as the negative result can already derived for the X-less fragement of LTL interpreted only on the sequence of actions induced by a run. The*

culprit can be found in the possibility to express the synchronization of independent events via a formula that is interpreted on the global interleaving. Regarding our example: when there is no communication between the underlying local processes, all fireings of actions on both processes are assumed independent.

Thus we propose to abandon interleaving-based semantics in favor of a partial-order one and to consider (temporal) logics based on this structure. In the following, we will focus monadic second order logic (MSO) on a rendering of the underlying partial of a run of a CPS *into graphs.*

2 Run Graphs and MSO

Definition 4 For finite alphabets Λ and Π, a (Λ,Π)-*labeled directed* graph is a tuple $G = \langle V, A, arc, \lambda, \pi \rangle$ where V is the finite, non-empty set of vertices, A the set of arcs (often called edges), the function *arc* maps each arc to an ordered pair of vertices, as well as $\lambda : V \to 2^{\Lambda}$ labels each vertex by an element from 2^{Λ} and $\pi : A \to \Pi$ labels each arc by an element from Π.

For $arc(a) = (v, v')$ where $a \in A$ and $v, v' \in V$, we say that v is the *source* vertex of a and v' its *destination*. Given a graph $G = \langle V, A, arc, \lambda, \pi \rangle$, then a graph $G' = \langle V', A', arc', \lambda', \pi' \rangle$ is a *subgraph* of G if $V' \subseteq V$ and $A' \subseteq V$, and arc', λ', π' can be derived by restricting the original counterparts to V' and A'. An *induced subgraph* demands in addition that for all $a \in A$ with $arc(a) = (x, y)$ and $x, y \in V'$ it holds that $a \in A'$. For $a \in A$ let $G \setminus a$ be the graph obtained by removing a from G and by *contracting* the source and destination of a to one new vertex. A graph H is a *minor* of a graph G if H is isomorphic to a graph G'' that can be derived from G by finitely many arc contractions, as well as deletions of arcs and vertices (the latter implies the deletion of arcs connected to these vertices).

Given a graph G, we define the *treewidth* of G as usual via graph decompositions [RS86]. The treewidth measures by a natural number how "close" G is to a tree where trees have treewidth 1. (Recall that the treewidth of a graph corresponds to the size of a separator for the graph, i.e., the number of edges one has to delete to separate the graph into components, which equals 1 for trees.) A class of graphs has *bounded treewidth* if there exists a constant $k \in \mathbb{N}$ such that all graphs in this class have a treewidth lower or equal to k; else it has *unbounded treewidth*.

Fact 5 *For $k \in \mathbb{N}$, let G_k be the $k \times k$ grid; then (i) G_k has treewidth k, thus (ii) the class of all grids $\{G_k \mid k \in \mathbb{N}\}$ has unbounded treewidth; further, (iii) if H is a minor of G then the treewidth of H is lower or equal to the one of G. [Bod98]*

Monadic Second Order Logic on Graphs We introduce MSO_V interpreted over the previously introduced graphs where formulas are constructed as below for first-order variables x, y over vertices, and X is a second-order variable over sets of vertices. Correct MSO_V formulae are given by the following BNF expression:

$$\varphi ::= x = y \mid Lab_l(x) \mid Arc(x,y) \mid x \in X \mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists x(\varphi) \mid \exists X(\varphi)$$

Informally, we interpret $=$ as equality on V, the predicate $Lab_l(x)$ for $l \in 2^{\Lambda}$ is true if the vertex x is labeled by l, and $x \in X$ is true if x is in the set represented by the second order variable X. The core of MSO_V is the incidence predicate $Arc(x,y)$ that is true if there exists an arc whose

source vertex is represented by x and whose destination is represented by y. Note that we have existential quantification over both first and second order variables. (A formal introduction of the semantics can be found in [CE11].) We write $G \models \varphi$ if a graph G satisfies a formula φ. We add the usual additional syntactic sugar by introducing $(\varphi \vee \phi)$, $(\varphi \to \phi)$, and $(\forall x(\varphi))$.

MSO_A (often written MSO_2 or MSO-2 in the literature) is introduced analogously whereas we additionally allow first and second order variables over arcs, and first- and second-order quantification over arcs. The central difference between MSO_A and MSO_V is the ternary incidence predicate $Arc(x,y,z)$ in MSO_A which is true if x is an arc with source y and destination z. We extend $Lab_l(x)$ to include arc labellings when x is an arc and $l \in \Pi$, as well as demand the $=$-relation to extend to arcs.

Note that $\big(\exists x\big(Arc(x,y,z)\big)\big)$ is equivalent to the incidence predicate of MSO_V. Thus, we can directly translate any formula from MSO_V to MSO_A. The reverse direction only holds if we restrict ourselves to the class of graphs of bounded treewidth [CE11]. As this is the case in the following, we will not distinguish between MSO_V and MSO_A, thus only refer to "MSO".

Example 1 *The following MSO_A formula $\chi(x,y,l)$ (with free variables x,y,l) expresses a "causal" reachability problem: there exists an l-labeled path from vertex x to vertex y:*

$$\chi(x,y,l) \equiv \forall X \Big(x \in X \wedge \forall u,v \big(u \in X \wedge arc(z,u,v) \wedge Lab_l(z) \to v \in X \big) \to y \in X \Big)$$

Then again, we can characterize classes of graphs that are "incompatible" with MSO as follows:

Fact 6 *If a class of graphs contains for each $k \in \mathbb{N}$ a graph G that has G_k as induced subgraph, then MSO is undecidable on this class of graphs. [See75]*

Hyperedge Replacement Grammars We introduce hypergraphs as extension of "classical" graphs. A $((\Lambda,\Pi)$-labeled) *hypergraph* is tuple $\mathcal{H} = \langle V, A, arc, \lambda, \pi \rangle$, analogous to the previous definition of graphs, whereas A is a finite sorted set $A = \bigcup_{i \in \mathbb{N}} A_i$ (i.e., the A_i are disjoint), whose elements are called hyperarcs (or hyperedges), and we define $arity(a) = i$ if $a \in A_i$; further, arc maps an arc $a \in A_i$ to a sequence of nodes of size i, i.e., $arc(a) = (v_1, \cdots, v_n)$ for $n = arity(a)$. Note that the previously introduced graphs are hypergraphs where the only possible arity of (hyper-)arcs is 2.

Embeddable hypergraphs are a tuple $\langle \mathcal{H}, ext \rangle$ where ext is a sequence of distinct "external" vertices $(v_1, \cdots, v_k)$ of $\mathcal{H}$ (with $v_1, \cdots, v_k \in V$). We abuse notation and write $\mathcal{H}$ for the tuple $\langle \mathcal{H}, ext \rangle$ and $ext(\mathcal{H})$ for the sequence ext; further let $type$ map an embeddable hypergraph $\mathcal{H}$ to the size of its sequence $ext(\mathcal{H})$. Let us assert that all the hypergraphs in the remainder are over the same two set of labels Λ and Π. We depict external vertices by $\bullet$ and others by $\circ$.

We define the operation of *hyperedge replacement* as follows: let $\mathcal{H}$ be a hypergraph, $a \in A$ one of its arcs, and $\mathcal{H}'$ an embeddable hypergraph disjoint from $\mathcal{H}$ with $arity(a) = type(\mathcal{H}')$ (if they are not disjoint, we take disjoint copies the usual way), then $\mathcal{H}[a \leftarrow \mathcal{H}']$ is the hypergraph that is equivalent to the disjoint union of $\mathcal{H}$ and $\mathcal{H}'$ where we deleted the arc a in $\mathcal{H}$ and fused $arc(a) = (v_1, \cdots, v_n)$ componentwise with $ext(H') = (v'_1, \cdots, v'_n)$. Further, we adapted the labellings of vertices accordingly as disjoint union of the labellings of $\mathcal{H}$ and $\mathcal{H}'$ where we assign the union of the original labels to the fused vertices.

A *hyperedge replacement grammar* (HRG) is a tuple $\mathcal{G} = \langle N, T, \mathcal{P}, \mathcal{H}_0 \rangle$ where N and T are the sets of non-terminal and terminal hyperarcs. $\mathcal{H}_0$ is the initial hypergraph. $\mathcal{P}$ is a finite set of production rules that map a hyperedge $a \in N$ to an embeddable hypergraph $\mathcal{H}$ such that $arity(a) = type(\mathcal{H})$ and all vertices in $ext(\mathcal{H})$ are labeled by $\emptyset$. We write rules as $R : a \hookrightarrow \mathcal{H}$ where we add the syntactic sugar of an identifier R. The *width* of an HRG $\mathcal{G}$ is the maximal number of vertices of the hypergraphs at the right-hand sides of all rules in $\mathcal{P}$ minus one.

Given an HRG $\mathcal{G} = \langle N, T, \mathcal{P}, \mathcal{H}_0 \rangle$, we define the *derivation relation* $\Rightarrow_{\mathcal{G}}$ between two hypergraphs $\mathcal{H}, \mathcal{H}'$ as follows: $\mathcal{H} \Rightarrow_{\mathcal{G}} \mathcal{H}'$ iff $\mathcal{H}' = \mathcal{H}[a \leftarrow \mathcal{H}_i]$, and $a \hookrightarrow \mathcal{H}_i$ is a rule in $\mathcal{P}$ for $a \in N$ and an embedded hypergraph $\mathcal{H}_i$. Let $\mathcal{L}(\mathcal{G})$ be the class of hypergraphs that can be derived from $\mathcal{H}_0$ in a finite number of $\Rightarrow_{\mathcal{G}}$-derivations, and that do only contain arcs in T. We refer to [DKH97] for more details on HRG and HRG languages.

Fact 7 *Given an HRG $\mathcal{G}$ of width k (for $k \in \mathbb{N}$), then the treewidth of the class of graphs $\mathcal{L}(\mathcal{G})$ is bounded by k. [Lau88]*

As graphs generated by HRG have bounded treewidth, one can draw a bridge to the known fundamental connection of decidability of MSO on classes of graphs with bounded treewidth:

Fact 8 *Given a hyperedge replacement grammar $\mathcal{G}$ that describes a class of graphs, and an MSO formula φ, then checking whether there exists a graph G in this class with $G \models \varphi$ is decidable. (follows directly by combining Fact 7 with [CE11])*

From CPS to Run Graphs In Section 1 we focused an interleaving-based semantics by viewing CPS runs as alternating sequences of configurations and actions. Alternatively, we can regard runs as linear sequence of *events* that are labeled by actions. Formally, let E be a set of events, $\lambda : E \to Act$ their labelling by actions, then a finite run ρ is a word in E^*. Analogously to the partition of Act into Act^p for $p \in Proc$, we partition E into sets E^p.

Due to the underlying semantics of a CPS, the events in the sets E^p (for $p \in Proc$) are linearly ordered; interprocess dependencies arise via communication, i.e., via events labeled by matching pairs of send and receive actions. For RCPS, we also need to take a closer look on events labeled by matching *push*/*pop* actions. We thus introduce the three order relations $succ^p$, $<_m$, and $<_{pd}$ on E for $(e, e') \in E \times E$:

- $(e, e') \in succ^p$ for $p \in Proc$ if e' is a direct successor of e on process p
- $e <_m e'$ if (e, e') is labeled by a matching pair of send/receive actions
- $e <_{pd} e'$ if e and e' are labeled by a matching pair of *push*/*pop* actions.

Given a run $\rho = (x_0, a_1, x_1, \ldots, a_n, x_n)$ of a CPS, we derive the corresponding *natural order of events* $[\rho] = \langle \{1, \ldots, n\}, \leq, \lambda \rangle$ where $\leq$ is the transitive reflexive closure of $\bigcup_p succ_p \cup <_m$. Note that we do not need to include $<_{pd}$ explicitly as it is already covered by the transitive closure of direct successors, however it plays an important role in causal entanglements.

The *run graph* corresponding to a run ρ of a CPS $\mathcal{Q}$ is a (Λ, Π)-labeled graph where Λ equals $\{fst, lst\} \cup \bigcup_{p \in Proc}(\Delta^p \cup Z^p)$ and $\Pi = \{succ^p \mid p \in P\} \cup \{pd, m\}$ over the vertex set $E \cup \{e_0^p, e_{\mathcal{F}}^p \mid p \in Proc\}$ and arcs labeled $succ^p$ / pd / m if the underlying source and destination vertices's events

are related by $succ^p$ / $<_{pd}$ / $<_m$. In addition we assert an initial and final event vertex e_0^p / $e_{\mathcal{F}}^p$ per process which is connected by an $succ^p$ arc to the first and from the last event and which is labeled by *fst* / *lst*. We depict *m*-labeled arcs by ○ -m-> ○ (dashed) and *pd*-labeled by ○ ⋯pd⋯> ○ (dotted).

Given an CPS $\mathcal{Q}$, let $\mathcal{RG}(\mathcal{Q})$ the set of graphs that are run graphs of a possible run of $\mathcal{Q}$. We want to use MSO to represent propositions about run graphs. As the set Λ depends on the underlying CPS $\mathcal{Q}$, we parametrize the logic by the CPS and write $\varphi \in \mathrm{MSO}(\mathcal{Q})$ for an MSO formula φ over this signature of labels.

Example 2 *We can now transfer Example 1 to specify the behaviour of an* RCPS $\mathcal{R}$. *The following formula* $\phi_p(z^p) \in MSO(\mathcal{R})$ *expresses that process p of* $\mathcal{R}$ *must pass a local control state* $z^p \in Z^p$ *at least once along its run:*

$$\phi_p(z^p) \equiv \exists e, e' \left(\chi(e, e', succ^p) \wedge Lab_{fst}(e) \wedge \bigvee_{a \in Act^p, z \in Z^p} Lab_{(z,a,z^p)}(e') \right)$$

3 Representing Classes of Run Graphs by HRG

3.1 Bounded Finite CPS

Proposition 1 *Given a finite* CPS $\mathcal{Q}$ *and* $\varphi \in MSO(\mathcal{Q})$, *then it is undecidable whether there exists* $G \in \mathcal{RG}(\mathcal{Q})$ *such that* $G \models \varphi$.

Idea of Proof *Combining Fact 5(iii) and Fact 6 allows for the counterargument to the previous decision question, as one can embed any* $k \times k$*-grid into the run graphs of a finite* CPS *over the topology* p ⇄ q . *We will exemplify the derivation of* G_3 *as graph minor in Figure 1a. Contracting the gray marked vertices and deleting the dotted arcs leads to* G_3 *as proper minor. Analogous constructions are possible for all other* G_k *for* $k \in \mathbb{N}$.

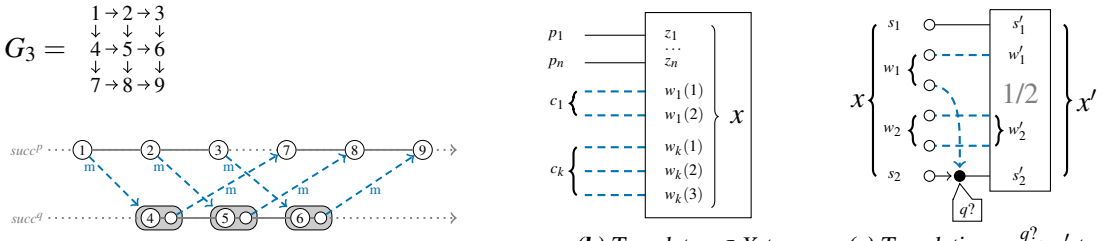
Thus, we need to focus on limitations for finite CPS. The well-established connection to MSO logics interpreted on "bounded" message sequence graphs in [GKM07, LM04] suggests the following restriction which we subsequently lift into our setting of run graphs:

Definition 5 A run of a CPS is (locally) *b*-bounded, if for all of its prefixes the number of unreceived messages on each channel is less than or equal to *b*. A CPS is (existentially locally) *b*-bounded for $b \in \mathbb{N}$, if for any run there exists an order-equivalent one that is *b*-bounded.

A *cut* of a run graph is a sequence $(e^p)_{p \in Proc}$ of vertices such that $e^p \in E^p$, and it holds that if there is a path in the run graph from e^p to e^q (for p, q distinct elements of *Proc*) then e^p is the only element of E^p in this path.

Definition 6 A run graph is *b*-bounded (for $b \in \mathbb{N}$) if for each cut $(e^p)_{p \in Proc}$ there is no channel $c \in Ch$ with $src(c) = p$ such that there are more than *b* send events on this channel before e^p that are not received before e^q with $dst(c) = q$.

We construct a HRG $\mathcal{G}(\mathcal{Q}, b) = \langle N, T, \mathcal{P}, \mathcal{H}_0 \rangle$ directly from the *finite* transition system $[\![\mathcal{Q}]\!]$ of a *b*-bounded finite CPS as follows (see below for a concrete example):

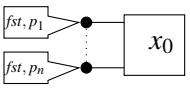


(a) *Embedding G_3 into runs on* $p \rightleftarrows q$.

(b) *Translate $x \in X$ to a Non-Terminal.*

(c) *Translating $x \xrightarrow{q?} x'$ to a Derivation Rule.*

Figure 1. Decidability Anti-Pattern via Embedding Grids / Positive Patterns via Grammar.

- $\mathcal{G}(\mathcal{Q},b)$ generates a run graph from left to right, i.e., a single $succ^p$-labeled path of E^p vertices for all $p \in Proc$ and appropriate m-labeled arcs;
- each non-terminal stores the capacity of the channel bound that is already used as well as stores unreceived messages in fifo order, i.e., each non-terminal represents a cut/configuration of $[\![\mathcal{Q}]\!]$ (see Figure 1b);
- thus we can directly translate each transition $x \xrightarrow{a} x'$ in $[\![\mathcal{Q}]\!]$ to a HRG rule (see Figure 1c);
- and the inital configuration maps to the following initial hypergraph 

We can interpret non-terminals as cuts: starting from the initial cut $\mathcal{H}_0$, where all processes are in its initial events, a derivation in $\mathcal{G}(\mathcal{Q},b)$ corresponds to a sequence of cuts that finally arrives at the last cut, where all processes are in *lst*-labeled events.

Example 3 *Figure 2 represents the HRG generation of a run graph of a 2-bounded finite* CPS *over the architecture* $p \rightleftarrows q$ *where $|M| = |S^p| = |S^q| = 1$, i.e., both processes only have one state and the message alphabet is of size one. Thus we label event vertices only by actions. (The detailed grammar can be found in the appendix.)*

Lemma 3 *For a given $b \in \mathbb{N}$ and a b-bounded finite* CPS $\mathcal{Q}$*, the class $\mathcal{L}(\mathcal{G}(\mathcal{Q},b))$ equals the class of run graphs $\mathcal{RG}(\mathcal{Q})$.*

Proof. For a $G \in \mathcal{RG}(\mathcal{Q})$, we assume that ρ is a b-bounded run conform to G. Then we can generate a sequence of cuts of G that left-to-right traces ρ. This sequence of cuts can be generated by $\mathcal{G}(\mathcal{Q},b)$.

For the other direction, a derivation of $\mathcal{G}(\mathcal{Q},b)$ is a sequence of cuts, such that each cut can be represented by a non-terminal. By construction, there are no more than b pending messages per channel at each non-terminal. Assert that there exists a cut in $G \in \mathcal{L}(\mathcal{G}(\mathcal{Q},b))$ that was not generated by the current derivation of G but which has more then b messages pending for a channel. As a the production rules of $\mathcal{G}(\mathcal{Q},b)$ are derived from the transition of $[\![\mathcal{Q}]\!]$ and as derivation in the HRG is equivalent to a run ρ in $[\![\mathcal{Q}]\!]$, we would have found a prefix of a run that has more than b pending messages, contradicting the boundedness of ρ. □

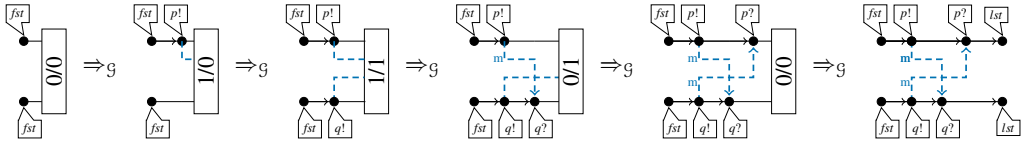


Figure 2. Possible Derivation of the HRG of Example 3 .

Corollary 1 *Given a b-bounded finite* CPS $\mathcal{Q}$*, then all run graphs in* $\mathcal{RG}(\mathcal{Q})$ *have treewidth bounded by* $|Proc| \cdot b \cdot |Ch|$.

Alternatively stated, the message sequence graphs (the graph rendering of the underlying message sequence charts) of b-bounded finite CPS have bounded treewidth.

Proposition 2 *Given a b-bounded finite* CPS $\mathcal{Q}$*, and* $\varphi \in MSO(\mathcal{Q})$*, then we can decide whether there exists a graph* $G \in \mathcal{RG}(\mathcal{Q})$ *such that* $G \models \varphi$.

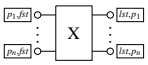
Proof. We reduce the given decision problem to deciding whether there exists a graph G in the HRG language $\mathcal{L}(\mathcal{G}(\mathcal{Q},b))$ such that $G \models \varphi$. The latter is decidable due to Fact 8, and, by Lemma 3, returns the demanded run graph of $\mathcal{Q}$. □

3.2 Eager 1-stack RCPS

Proposition 3 *Given a eager* RCPS $\mathcal{R}$ *and* $\varphi \in MSO(\mathcal{R})$*, then it is undecidable whether there exists* $G \in \mathcal{RG}(\mathcal{R})$ *such that* $G \models \varphi$.

Instead of focussing RCPS with the non-confluent restriction, we will restrict ourselves in the following to the larger class of eager RCPS where for each run there exists an order-equivalent one that is both eager and 1-stack (cf. Lemma 1). W.l.o.g. we assume all channels of eager RCPS to not enter their growing phase. However, we will not derive a HRG directly as in the previous case, but follow the subsequent three steps:

First, given an RCPS $\mathcal{R}$ over the set of processes $Proc = \{p_1, \cdots, p_n\}$, we construct an HRG $\mathcal{G}(\mathcal{R}) = \langle N, T, \mathcal{P}, \mathcal{H}_0 \rangle$ as follows:

- $\mathcal{H}_0$ equals  , where we additionally label vertices by process names;
- $N = \{X, I\}$ consists of two $(2 \cdot n)$-ary hyperedges;
- the production rules $\mathcal{P}$ presented in Figure 3 depend only on P;
- we depict the arc representing a matching send/receive related vertex pair in the case of eager communication by ○---^{m}---○ and interpret it as  with respect to graph properties like paths.

Second, we extend $\mathcal{G}(\mathcal{R})$ to an HRG $\widetilde{\mathcal{G}}(\mathcal{R})$ that generates run graphs with a vertex labeling over Δ as follows: we label the initial vertices for each process additionally by z_0^p, we transform a production rule of $\mathcal{G}(\mathcal{R})$ that generates a send/receive matching set of vertices for distinct $p_i, p_j \in P$ (below on the right) to a set of rules in $\widetilde{\mathcal{G}}(\mathcal{R})$ (left):

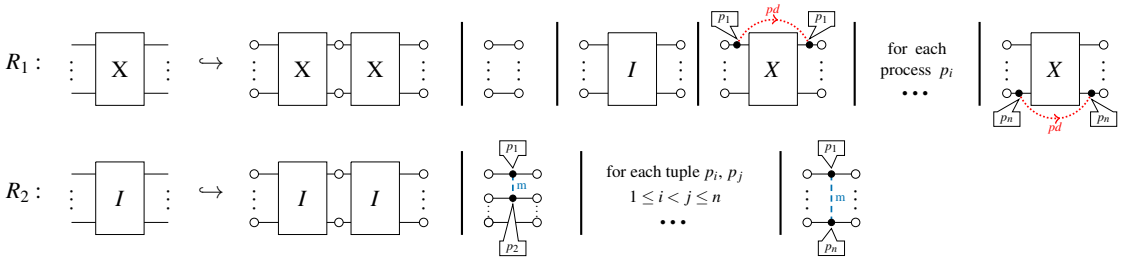
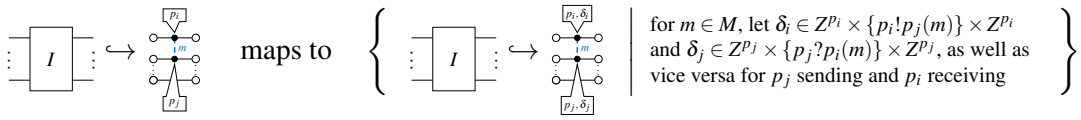
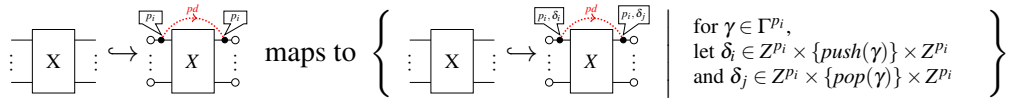


Figure 3. Production Rules of $\mathcal{G}(\mathcal{R})$ for 1-stack RCPS $\mathcal{R}$ over $Proc = \{p_1, \cdots, p_n\}$.



Thus, we *guess* the exchanged message $m \in M$, assure that both vertices are labeled by the appropriate matching actions, and additionally *guess* two control-states for each process.

We analogously translate production rules that generate matching push/pop vertex pairs to a set of rules that has the same right-hand-side hypergraph but additionally guesses the $\gamma \in \Gamma^p$; thus translating $\mathcal{G}$'s rules (right) to a set of rules in $\widetilde{\mathcal{G}}(\mathcal{R})$ (left):



Note that we assure that the same letter from the pushdown alphabet is assigned to a matching push/pop vertex pair.

Third, we *verify* that the previous guess of triples δ is correct with respect to the underlying $(\mathcal{A}^p)_{p \in Proc}$ by the MSO formula

$$\varphi_\Delta \equiv \forall v \left(\bigwedge_{p \in Proc} \left(\varphi^p_{\text{label}}(v) \wedge \varphi^p_{\text{trans}}(v) \right) \right)$$

which tests whether nodes are (i) correctly labeled by either a transition rule, or by *fst*/*lst*; and (ii) the labels of neighboring nodes mirror a correct transition of $\mathcal{R}$.

$$\varphi^p_{\text{label}}(v) \equiv Lab_{fst}(v) \vee Lab_{lst}(v) \vee \left(\bigvee_{(z,a,z') \in \Delta^p} Lab_{(z,a,z')}(v) \right)$$

$$\varphi^p_{\text{trans}}(v) \equiv \bigvee_{(z,a,z') \in \Delta^p} \left(Lab_{(z,a,z')}(v) \rightarrow Predecessor^p(v,z) \wedge Successor^p(v,z') \right)$$

Thus, the direct predecessor/successor of a node—if is not tagged by *fst*/*lst*—is labeled by a rule that ends/starts in the control state required by the current rule.

$$Predecessor^p(v,z) \equiv \exists v' \left(Arc_{succ^p}(v',v) \wedge \left(Lab_{fst}(v') \vee \bigvee_{a \in Act^p, z' \in Z^p} Lab_{(z',a,z)}(v') \right) \right)$$

$$Successor^p(v,z) \equiv \exists v' \left(Arc_{succ^p}(v,v') \wedge \left(Lab_{lst}(v') \vee \bigvee_{a \in Act^p, z' \in Z^p} Lab_{(z,a,z')}(v') \right) \right)$$

Lemma 4 *Given an* RCPS $\mathcal{R}$ *where all runs can be reordered into a run that is both eager and 1-stack, then the class of graphs, consisting of those in* $\mathcal{L}(\widetilde{\mathcal{G}}(\mathcal{R}))$ *that additionally fulfill* φ_Δ*, equals* $\mathcal{RG}(\mathcal{R})$.

The proof of the previous lemma relies on an argument close the proof of Fact 4 that allows to "cut" a run with respect to a context-free pattern. As before, we can directly apply Fact 8 on the formula $\varphi \wedge \varphi_\Delta$ to derive the following:

Proposition 4 *Given an* RCPS $\mathcal{R}$ *where all runs are order equivalent to a run that is both eager and 1-stack, and an MSO formula* φ *over the appropriate signature of run graphs, then we can decide whether there exists a graph* $G \in \mathcal{RG}(\mathcal{R})$ *such that* $G \models \varphi$.

4 Conclusion

This article proposes to leave behind the combination of linear temporal logic and CPS in favor for MSO on the underlying partial order of events. A graph-based representation of a run by a run graph proves ideal for a high-level view on MSO-based partial order logics. Describing the evolution of CPS by graph grammars is a versatile tool to (i) get better insights into the intricacies that lead to undecidability result (both [BZ83] and [Ram00] can be reduced as intermediary step to a simple semantic argument on grid graphs that allow to represent halting runs of Turing machines); as well as (ii) to naturally derive *semantic* boundedness conditions that include currently known ones (e.g., existential b-boundedness) and open the door for new ones; thus, (iii) this lifts the idea of semantic boundedness to the bounded width of graph grammars. The latter can also be seen as a different rendering of the fact that decidability results rely on the ability to parse the structure of a run by only storing a finite amount of information. However, the question of deriving *syntactic* restrictions that lead to the semantic ones found above remains often open, e.g, in the case of eager RCPS where there always exists an order-equivalent one-stack run (see Lemmata 1 & 2).

Related Works Generating graph representations of partial order runs via graph grammars is orthogonal to encoding the runs of transition systems itself into derivations of an appropriate graph grammar [Roz97]. The presented approach allows to specify/verify properties on runs, and not to only verify whether a configuration of a certain pattern is reachable.

Our run graphs are close in spirit to message sequence charts (MSC), and high-level MSC. The connection of MSO and (restricted classes of) MSC (Proposition 2) is well-known, e.g., [Mad03, GKM07]. However, Corollary 1 introduces the novel aspect of bounded treewidth. The relation of MSO and a nested-word representation of RCPS's runs was first considered in [MP11], whereas the focus remained on proving the bounded treewidth of RCPS and related models by sophisticated graph decomposition schemata. MSO for finite processes synchronizing over shared variables were already investigated in [MTY05] and extended in [BCGZ11] to recursive processes with a global phase boundedness restriction. The latter article was able to derive a (2)ExpTime complexity for model checking MSO on the underlying traces, based on an a priori known trace decomposition from [MP11]. Our considerations would give rise to a more gen-

eral approach based on a given graph grammar and its straightforward relation to tree automata recognizing term representations of run graphs [CE11].

A closer look onto the complexities of Propositions 2 and 4 reveals a tower of exponentials depending on the quantifier height of the MSO formula [CE11]. Recent results on MSO definable temporal logics on traces derive better complexities [BCGZ11, GK10] based on optimized translations to automata problems. Those fragments of MSO could, e.g., define an operator F^p (for "reachable for process $p \in Proc$") as MSO formula with one free variable $\lambda x(\phi(x))$ (see Example 2). Optimal algorithms for these MSO fragments on the classes of RCPS considered here remain an open question.

Outlook One can directly extend the previously introduced proof technique to other settings like bounded lock graphs [KW10] or the k-phase bounded RCPS [HLMS11]. A natural extension of our grammars would focus infinite run graphs or branching time properties. Besides MSO, we also started to take a look on interpreting linear temporal logics on Mazurkiewicz traces in our graph setting, which also would propose a propositional logic weaker than full MSO. Currently, a variety of other graph measures came into focus, like bounded path-width [DSG11] or the vertex cover number [PL11]; these could be the point of departure for deeper insights into novel semantic restrictions for other subclasses of (R)CPS.

Acknowledgements We thank the anonymous reviewers and the GT-VMT community for constructive feedback, as well as J. Leroux, A. Muscholl, and G. Sutre for stimulating discussions. This work was partly supported by the project ANR AVerISS (ANR-06-SETIN-001).

Bibliography

[BCGZ11] B. Bollig, A. Cyriac, P. Gastin, M. Zeitoun. Temporal Logics for Concurrent Recursive Programs: Satisfiability and Model Checking. In *Proc. of MFCS'11*. LNCS 6907, pp. 132–144. Springer, 2011.

[Bod98] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *TCS* 209(1–2):1–45, 1998.

[BZ83] D. Brand, P. Zafiropulo. On Communicating Finite-State Machines. *JACM* 30(2):323–342, 1983.

[CE11] B. Courcelle, J. Engelfriet. *Graph Structure and Monadic Second-Order Logic, a language theoretic approach*. Cambridge Univ. Press, 2012. (to be published).

[CF05] G. Cécé, A. Finkel. Verification of programs with half-duplex communication. *Information and Computation* 202(2):166–190, 2005.

[CV09] R. Chadha, M. Viswanthan. Deciding branching time properties for asynchronous programs. *TCS* 410(42):4169–4179, 2009.

[DKH97] F. Drewes, H.-J. Kreowski, A. Habel. Hyperedge Replacement Graph Grammars. In [Roz97]. pp. 95–162.

[DSG11] G. Delzanno, A. Sangnier, G. Zavatarro. On the Power of Cliques in the Parameterized Verification of Ad Hoc Networks. In *Proc. of FOSSACS'11*. LNCS 6604, pp. 441–455. Springer, 2011.

[GK10] P. Gastin, D. Kuske. Uniform satisfiability problem for local temporal logics over Mazurkiewicz traces. *Information and Computation* 208:797–816, 2010.

[GKM07] B. Genest, D. Kuske, A. Muscholl. On communicating automata with bounded channels. *Fundamenta Informaticae* 80:147–167, 2007.

[Heu10] A. Heußner. Run Graphs of Communicating Processes and MSO. In *CONCUR'10 YR Workshop*. pp. 75–78. 2010.

[HLMS11] A. Heußner, J. Leroux, A. Muscholl, G. Sutre. Reachability Analysis of Communicating Pushdown Systems. *LMCS*, 2011. to be published.

[KG07] V. Kahlon, A. Gupta. On the analysis of interacting pushdown systems. In *Proc of POPL'07*. Pp. 303–314. ACM, 2007.

[KW10] V. Kahlon, C. Wang. Universal Causality Graphs: A Precise Happens-Before Model for Detecting Bugs in Concurrent Programs. In *Proc. of CAV 2010*. LNCS 6174, pp. 434–449. 2010.

[Lau88] C. Lautemann. Decomposition Trees: Structured Graph Representation and Efficient Algorithms. In *Proc. of CAAP'88*. LNCS 299, pp. 28–39. Springer, 1988.

[LM04] M. Lohrey, A. Muscholl. Bounded MSC communication. *Information and Computation* 189(2):160–181, 2004.

[LMP08] S. La Torre, P. Madhusudan, G. Parlato. Context-Bounded Analysis of Concurrent Queue Systems. In *Proc. of TACAS'08*. LNCS 4963, pp. 299–314. Springer, 2008.

[Mad03] P. Madhusudan. Model-checking Trace Event Structures. In *Proc. of LICS*. Pp. 371–380. 2003.

[MP11] P. Madhusudan, G. Parlato. The Tree Width of Automata with Auxiliary Storage. In *Proc. of POPL 2011*. ACM, 2011.

[MTY05] P. Madhusudan, P. S. Thiagarajan, Yang. The MSO Theory of Connectedly Communicating Processes. In *Proc. of FSTTCS 2005*. LNCS 3821, pp. 201–212. Springer, 2005.

[PL11] M. Praveen, K. Lodaya. Parameterized Complexity Results for 1-safe Petri Nets. In *Proc. of CONCUR'11*. LNCS 6901, pp. 358–372. Springer, 2011.

[Ram00] G. Ramalingam. Context-sensitive synchronization-sensitive analysis is undecidable. *ACM Trans. Program. Lang. Syst.* 22(2):416–430, 2000.

[Roz97] G. Rozenberg. Handbook of Graph Grammars and computing by Graph Transformation. World Scientific, 1997.

[RS86] N. Robertson, P. D. Seymour. Graph minors III: Planar tree-width. *Journal of Combinatorial Theory, Series B* 41:92–114, 1986.

[See75] D. Seese. Ein Unentscheidbarkeitskriterium. *WZ HU Berlin Mathem.-Naturw. Reihe XXIV* 6:772–780, 1975.