Branching-Time Model Checking Gap-Order Constraint Systems (Extended Version)

Richard Mayr
University of Edinburgh, UK

Patrick Totzke
University of Warwick, UK

Abstract. We consider the model checking problem for Gap-order Constraint Systems (GCS) w.r.t. the branching-time temporal logic CTL, and in particular its fragments EG and EF. GCS are nondeterministic infinitely branching processes described by evolutions of integer-valued variables, subject to Presburger constraints of the form $x - y \geq k$, where x and y are variables or constants and $k \in \mathbb{N}$ is a non-negative constant. We show that EG model checking is undecidable for GCS, while EF is decidable. In particular, this implies the decidability of strong and weak bisimulation equivalence between GCS and finite-state systems.

1. Introduction

Counter machines [17] extend a finite control-structure with unbounded memory in the form of counters that can hold arbitrarily large integers (or natural numbers), and thus resemble basic programming languages. However, almost all behavioral properties, e.g., reachability and termination, are undecidable for counter machines with two or more counters [17]. For the purpose of formal software verification, various formalisms have been defined that approximate counter machines and still retain the decidability of some properties. E.g., Petri nets model weaker counters that cannot be tested for zero, and have a decidable reachability problem [15].

Gap-order constraint systems [19, 11, 4, 5] are another model that approximates the behavior of counter machines. They are nondeterministic infinitely branching processes described by evolutions of integer-valued variables, subject to Presburger constraints of the form $x - y \geq k$, where x and y are variables or constants and $k \in \mathbb{N}$ is a non-negative constant. Unlike in Petri nets, the counters can be tested for zero, but computation steps still have a certain type of monotonicity that yields a decidable reachability problem. In fact, control-state reachability is decidable even for the more general class of constrained multiset rewriting systems [1].

Previous work. Beyond reachability, several model checking problems have been studied for GCS and related formalisms. The paper [6] studies Integral Relational Automata (IRA), a model that is subsumed by GCS, that allows only constraints of the form $x \geq y$ or $x > y$, where y and x are variables or constants. It is shown that CTL model checking of IRA is undecidable, even for the restriction RCTL, that forbids next-state modalities. In contrast, model checking IRA remains decidable for the existential and universal fragments of CTL*. Models of equal expressivity include the monotonicity constraint systems (MCS) of [3] and $(\mathbb{Z}, <, =)$-automata in [8]. Demri and D'Souza [8] show that satisfiability and model checking LTL is decidable and PSPACE-complete.

Bozzelli and Pinchinat [4, 5] study the more general model of gap-order constraint systems (GCS), which strictly extend the models mentioned above. They show that model checking GCS is decidable and PSPACE-complete for the logic EGCCTL*, but undecidable for AGCCTL*, which are the existential and universal fragments of CTL*, respectively, extended with gap constraints as atomic propositions. Moreover, satisfiability is PSPACE-complete for both these fragments. EGCCTL* and AGCCTL* are not dual, since gap constraints are not closed under negation. Moreover, they are orthogonal to the fragments EF and EG considered in this paper, which allow nesting of negation and the operator *EF* (resp. *EG*). Checking fairness (the existence of infinite runs where a variable has a fixed value infinitely often) and thus termination (the non-existence of infinite runs), and also strong termination (the existence of a bound on the length of all runs) are decidable in polynomial space [5, 4]. An important ingredient for these results are effectively constructible under-approximations of the set of GCS runs induced by a given sequence of transitions, which preserve enabledness (Thm. 2 in [5]). This comes at the cost of losing information about the induced runs. In particular, it is impossible to recover (a representation of) the exact set of runs induced by a sequence of transitions from its approximation.

Our contribution. We study the decidability of model checking problems for GCS with fragments of computation-tree logic (CTL), namely EG and EF (see e.g. [9]).

We first show that EG-model checking is undecidable, even for the weaker model of IRA [6]. On the other hand, model checking GCS with respect to EF remains decidable. This positive result is based on the observation that one can use boolean combinations of gap constraints to represent the sets of variable valuations satisfying a given EF formula, and that such a representation can be effectively computed in a bottom-up fashion. An immediate consequence is that checking strong and weak bisimulation equivalence is decidable between GCS and finite-state systems.

2. Gap-Order Constraint Systems

Let $\mathbb{Z}$ and $\mathbb{N}$ denote the sets of integers and non-negative integers. A *labeled transition system* (LTS) is described by a triple $T = (V, Act, \longrightarrow)$ where V is a (possibly infinite) set of states, Act is a finite set of action labels and $\longrightarrow \subseteq V \times Act \times V$ is the labeled transition relation. We use the infix notation $s \xrightarrow{a} s'$ for a transition $(s, a, s') \in \longrightarrow$, in which case we say T makes an *a-step* from s to s'. For a set $S \subseteq V$ of states and $a \in Act$ we define the set of a-predecessors by $Pre_a(S) = \{s' | s' \xrightarrow{a} s \in S\}$. We write $\longrightarrow^*$ for the transitive and reflexive closure of $\longrightarrow$ and let $Pre^*(S) = \{s' | s' \longrightarrow^* s \in S\}$.

We fix a finite set Var of *variables* ranging over the integers and a finite set $Const \subseteq \mathbb{Z}$ of constants. Let Val denote the set of variable *valuations* $\nu : Var \rightarrow \mathbb{Z}$. To simplify the notation, we will extend the domain of valuations to constants, where they behave as the identity, i.e., $\nu(c) = c$ for all $c \in \mathbb{Z}$.

Definition 2.1. (Gap Constraints)
A *gap clause* over $(\mathit{Var}, \mathit{Const})$ is an inequation of the form

$$(x - y \geq k) \tag{1}$$

where $x, y \in \mathit{Var} \cup \mathit{Const}$ and $k \in \mathbb{Z}$. A clause is called *positive* if $k \in \mathbb{N}$. A (positive) gap *constraint* is a finite conjunction of (positive) gap clauses. A *gap formula* is an arbitrary boolean combination of gap clauses.

A valuation $\nu : \mathit{Var} \to \mathbb{Z}$ satisfies the clause $\mathcal{C} : (x - y) \geq k$ (write $\nu \models \mathcal{C}$) if it respects the prescribed inequality. That is,

$$\nu \models (x - y) \geq k \iff \nu(x) - \nu(y) \geq k. \tag{2}$$

We define the satisfiability of arbitrary gap formulae inductively in the usual fashion and write $\mathit{Sat}(\varphi) = \{\nu \in \mathit{Val} \mid \nu \models \varphi\}$ for the set of valuations that satisfy the formula φ. In particular, a valuation satisfies a gap constraint iff it satisfies all its clauses. A set $S \subseteq \mathit{Val}$ of valuations is called *gap definable* if there is a gap formula φ with $S = \mathit{Sat}(\varphi)$.

We will consider processes whose states are described by valuations and whose dynamics is described by stepwise changes in these variable valuations, according to positive gap constraints.

Let $\mathit{Var}' = \{x' \mid x \in \mathit{Var}\}$ be the set of primed copies of the variables. These new variables are used to express constraints on how values can change when moving from one valuation to another: x' is interpreted as the next value of variable x. A *transitional* gap clause (-constraint, -formula) is a gap clause (-constraint, -formula) with variables in $\mathit{Var} \cup \mathit{Var}'$. The combination $\nu \oplus \nu' : \mathit{Var} \cup \mathit{Var}' \to \mathbb{Z}$ of two valuations $\nu, \nu' : \mathit{Var} \to \mathbb{Z}$ maps variables $x \in \mathit{Var}$ to $\nu \oplus \nu'(x) = \nu(x)$ and $x' \in \mathit{Var}'$ to $\nu \oplus \nu'(x') = \nu'(x)$.

Transitional gap clauses can be used as conditions on how valuations may evolve in one step. For instance, ν may change to ν' only if $\nu \oplus \nu' \models \varphi$ for some gap clause φ.

Definition 2.2. A *Gap-Order Constraint System* (GCS) is given by a finite set of *positive* transitional gap constraints together with a labeling function. Formally, a GCS is a tuple $\mathcal{G} = (\mathit{Var}, \mathit{Const}, \mathit{Act}, \Delta, \lambda)$ where $\mathit{Var}, \mathit{Const}, \mathit{Act}$ are finite sets of variables, constants and action symbols, Δ is a finite set of *positive* transitional gap constraints over $(\mathit{Var}, \mathit{Const})$ and $\lambda : \Delta \to \mathit{Act}$ is a labeling function. Its operational semantics is given by an infinite LTS with states Val where

$$\nu \xrightarrow{a} \nu' \iff \nu \oplus \nu' \models \mathcal{C} \tag{3}$$

for some constraint $\mathcal{C} \in \Delta$ with $\lambda(\mathcal{C}) = a$. For a set $M \subseteq \mathit{Val}$ of valuations we write $\mathit{Pre}_{\mathcal{C}}(M)$ for the set $\{\nu \mid \exists \nu' \in M. \nu \oplus \nu' \models \mathcal{C}\}$ of $\mathcal{C}$-predecessors.

Observe that a positive gap constraint $(x - 0 \geq 0) \wedge (0 - x \geq 0)$ is satisfied only by valuations assigning value 0 to variable x. Similarly, one can test if a valuation equates two variables. Also, it is easy to simulate a finite control in a GCS using additional variables.[1] What makes this model computationally non-universal is the fact that we demand *positive* constraints: while one can easily demand an increase or decrease of variable x by *at least* some offset $k \in \mathbb{N}$, one cannot demand a difference of *at most* k (nor exactly k).

[1] In fact, [5, 4] consider an equivalent notion of GCS that explicitly includes a finite control.

Example 2.3. Consider the GCS with variables $\{x, y\}$ and single constant $\{0\}$ with two constraints $\Delta = \{\mathcal{C}X, \mathcal{C}Y\}$ for which $\lambda(\mathcal{C}X) = a$ and $\lambda(\mathcal{C}Y) = b$.

$$\mathcal{C}X = ((x - x' \geq 1) \wedge\ (y' - y \geq 0)\ \wedge\ (y - y' \geq 0)\ \wedge\ (x' - 0 \geq 0)) \tag{4}$$

$$\mathcal{C}Y = ((y - y' \geq 1)\ \wedge\ (x' - x \geq 0)\ \wedge\ (y' - 0 \geq 0)). \tag{5}$$

This implements a sort of lossy countdown where every step strictly decreases the tuple (y, x) lexicographically: $\mathcal{C}X$ induces a-steps that decrease x while preserving the value of y and $\mathcal{C}Y$ induces b-steps that increase x arbitrarily but have to decrease y at the same time. The last clauses in both constraints ensure that x and y never change from a non-negative to a negative value.

In the sequel, we allow ourselves to abbreviate constraints for the sake of readability. For instance, the constraint $\mathcal{C}X$ in the previous example could equivalently be written as $(x > x' \geq 0)\ \wedge\ (y = y')$.

3. Branching-Time Logics for GCS

We consider (sublogics of) the branching-time logic CTL over processes defined by gap-order constraint systems, where atomic propositions are gap clauses. The denotation of an atomic proposition $\mathcal{C} = (x - y \geq k)$ is $[\![\mathcal{C}]\!] = Sat(\mathcal{C})$, the set of valuations satisfying the constraint. Well-formed CTL formulae are inductively defined by the following grammar, where $\mathcal{C}$ ranges over the atomic propositions and $a \in Act$ over the action symbols.

$$\psi ::= \mathcal{C}\ \mid\ true\ \mid\ \neg\psi\ \mid\ \psi \wedge \psi\ \mid\ \langle a \rangle \psi\ \mid\ \mathit{EF}\psi\ \mid\ \mathit{EG}\psi\ \mid\ \mathit{E}(\psi \mathit{U} \psi) \tag{6}$$

To define the semantics, we fix a GCS $\mathcal{G}$. Let $Paths^{\omega}(\nu_0)$ be the set of infinite derivations

$$\pi = \nu_0 \xrightarrow{a_0} \nu_1 \xrightarrow{a_1} \nu_2 \ldots \tag{7}$$

of $\mathcal{G}$ starting with valuation $\nu_0 \in\ Val$ and let $\pi(i) = \nu_i$ denote the i-th valuation ν_i on π. Similarly, we write $Paths^*(\nu_0)$ for the set of finite derivations starting from ν_0. The denotation of formulae, with respect to the fixed GCS $\mathcal{G}$, is defined in the standard way.

$$[\![\mathcal{C}]\!] = Sat(\mathcal{C}) \tag{8}$$

$$[\![true]\!] = Val \tag{9}$$

$$[\![\neg\psi]\!] = Val \setminus [\![\psi]\!] \tag{10}$$

$$[\![\psi_1 \wedge \psi_2]\!] = [\![\psi_1]\!] \cap [\![\psi_2]\!] \tag{11}$$

$$[\![\langle a \rangle \psi]\!] = Pre_a([\![\psi]\!]) \tag{12}$$

$$[\![\mathit{EF}\psi]\!] = \{\nu \mid \exists \pi \in Paths^*(\nu).\ \exists i \in \mathbb{N}.\ \pi(i) \in [\![\psi]\!]\} \tag{13}$$

$$[\![\mathit{EG}\psi]\!] = \{\nu \mid \exists \pi \in Paths^{\omega}(\nu).\ \forall i \in \mathbb{N}.\ \pi(i) \in [\![\psi]\!]\} \tag{14}$$

$$[\![\mathit{E}(\psi_1 \mathit{U} \psi_2)]\!] = \{\nu \mid \exists \pi \in Paths^*(\nu).\ \exists i \in \mathbb{N}.\pi(i) \in [\![\psi_2]\!] \wedge \forall j < i.\pi(j) \in [\![\psi_1]\!]\} \tag{15}$$

We use the usual syntactic abbreviations $false = \neg true$, $\psi_1 \vee \psi_2 = \neg(\neg\psi_1 \wedge \neg\psi_2)$.

The sublogics EF and EG are defined by restricting the grammar (6) defining well-formed formulae: EG disallows subformulae of the form $\mathit{E}(\psi_1 \mathit{U} \psi_2)$ and $\mathit{EF}\psi$ and in EF, no subformulae of the form $\mathit{E}(\psi_1 \mathit{U} \psi_2)$ or $\mathit{EG}\psi$ are allowed. The *Model Checking Problem* is the following decision problem.

INPUT: A GCS $G = (\mathit{Var}, \mathit{Const}, \mathit{Act}, \Delta, \lambda)$, a valuation $\nu : \mathit{Var} \to \mathbb{Z}$
and a formula ψ.

QUESTION: $\nu \models \psi$?

Cerans [6] showed that general CTL model checking is undecidable for gap-order constraint systems. This result holds even for restricted CTL without *next* operators $\langle a \rangle$. In the following section, we show a similar undecidability result for the fragment EG. On the other hand, model checking GCS with the fragment EF turns out to be decidable; cf. Section 5.

4. Undecidability of EG Model Checking

Theorem 4.1. The model checking problem for EG formulae over GCS is undecidable.

Proof:
By reduction from the halting problem of deterministic 2-counter Minsky Machines (2CM). 2-counter machines consist of a deterministic finite control, including a designated halting control-state $halt$ and two integer counters that can be incremented and decremented by one and tested for zero. Checking if such a machine reaches the halting state from an initial configuration with control-state $init$ and counter values $x_1 = x_2 = 0$ is undecidable [17]. Given a 2CM M, we will construct a GCS together with an initial valuation ν_0 and a EG formula ψ such that $\nu_0 \models \psi$ iff M does not halt.

First of all, observe that we can simulate a finite control of n states using one additional variable $state$ that will only ever be assigned values from 1 to n. To do this, let $[p] \leq n$ be the index of state p in an arbitrary enumeration of the state set. Now, a transition $p \longrightarrow q$ from state p to q introduces the constraint $(state = [p] \wedge state' = [q])$. We will abbreviate such constraints by $(p \longrightarrow q)$ in the sequel and simply write p to mean the clause $(state = [p])$.

We use two variables x_1, x_2 to act as integer counters. Zero-tests can then directly be implemented as constraints $(x_1 = 0)$ or $(x_2 = 0)$. It remains to show how to simulate increments and decrements by exactly 1. Our GCS will use two auxiliary variables y, z and a new state err. We show how to implement increments by one; decrements can be done analogously.

Consider the x_1-increment $p \xrightarrow{x_1 = x_1 + 1} q$ that takes the 2CM from state p to q and increments the counter x_1. The GCS will simulate this in two steps, as depicted in Figure 1 below. The first step can arbitrarily increment x_1 and will remember (in variable y) the old value of x_1. The second step does not change any values and just moves to the new control-state. However, incrementing by more than one in

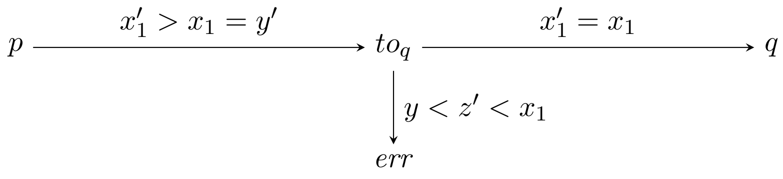


Figure 1. Forcing faithful simulation of x_1-increment. All steps contain the additional constraint $x_2' = x_2$, which is not shown, to preserve the value of the other counter x_2.

the first step enables an extra move to the error state err afterwards. This error-move is enabled if one can assign a value to variable z that is strictly in between the old and new value of x_1, which is true iff the increment in step 1 was not faithful. The incrementing transition of the 2CM is thus translated to the following three constraints.

$$(p \longrightarrow to_q) \wedge (x_1' > x_1 = y') \wedge (x_2' = x_2) \tag{16}$$

$$(to_q \longrightarrow q) \wedge (x_1' = x_1) \wedge (x_2' = x_2) \tag{17}$$

$$(to_q \longrightarrow err) \wedge (y < z' < x_1). \tag{18}$$

If we translate all operations of the 2CM into the GCS formalism as indicated above, we end up with an over-approximation of the 2CM that allows runs that faithfully simulate runs in the 2CM but also runs which 'cheat' and possibly increment or decrement by more than one and still don't go to state err in the following step.

We enforce a faithful simulation of the 2CM by using the formula that is to be checked, demanding that the error-detecting move is never enabled. The GCS will only use a unary alphabet $Act = \{a\}$ to label constraints. In particular, observe that the formula $\langle a \rangle err$ holds in every configuration which can move to state err in one step. Now, the EG formula

$$\psi = \textit{\textbf{EG}}(\neg halt \ \wedge \ \neg \langle a \rangle err) \tag{19}$$

asserts that there is an infinite path which never visits state $halt$ and along which no step to state err is ever enabled. This means ψ is satisfied by valuation $\nu_0 = \{state = [init], x_1 = x_2 = y = z = 0\}$ iff there is a faithful simulation of the 2CM from initial state $init$ with both counters set to 0 that never visits the halting state. Since the 2CM is deterministic, there is only one way to faithfully simulate it and hence $\nu_0 \models \psi$ iff the 2CM does not halt. Notice that the constructed GCS is in fact an IRA [6], since it only uses gap constraints of the form $x > y$ or $x = y$. □

5. Decidability of EF Model Checking

Let us fix sets Var and $Const$ of variables and constants, respectively. We will use an alternative characterization of gap constraints in terms of *monotonicity graphs* [2], which are finite graphs with nodes $Var \cup Const$. Monotonicity graphs are used to represent sets of variable valuations. We show that so represented sets are effectively closed under all logical connectors allowed in EF, and one can thus evaluate a formula bottom up.

Definition 5.1. (Monotonicity Graphs)
A *monotonicity graph* (MG) over $(Var, Const)$ is a finite, directed graph $M = (V, E)$ with nodes $V = Var \cup Const$ and in which each edge in E carries a *weight* in $\mathbb{Z} \cup \{-\infty, \infty\}$. The *degree* of M is the largest $k \in \mathbb{N}$ such that there is an edge with weight $-k$ in M or 0 if no edge has weight in $\mathbb{Z} \setminus \mathbb{N}$. The degree of a set $\{M_0, M_1, \ldots, M_j\}$ of MG is defined as the maximal degree of any MG M_k in the set.

[2] These were called *Graphose Inequality Systems* in [6] and *gap-graphs* in [19].

A valuation $\nu : \mathit{Var} \to \mathbb{Z}$ satisfies M (write $\nu \models M$) if for every edge $(x \xrightarrow{k} y)$ it holds that $\nu(x) - \nu(y) \geq k$. Let $Sat(M)$ denote the set of valuations satisfying M. A set $S \subseteq \mathit{Val}$ is *MG-definable* if there is a finite set $\{M_0, M_1, \ldots, M_j\}$ of MG such that

$$S = \bigcup_{0 \leq i \leq j} Sat(M_i) \tag{20}$$

and called MG^n-definable if there is such a set of MG with degree $\leq n$. We write *MG* and *MG*^{n} for the classes of MG- and MG^n-definable sets respectively.

For a monotonicity graph M, we write $M(x, y) \in \{-\infty, \infty\} \cup \mathbb{Z}$ for the least upper bound of the cumulative weight of all paths from node x to node y. Note that this is $-\infty$ if there is no such path. The *closure* $|M|$ is the unique complete MG with edges $x \xrightarrow{M(x,y)} y$ for all $x, y \in \mathit{Var} \cup \mathit{Const}$.

The following lemma and definition state some basic properties of monotonicity graphs that can easily be verified; see [6].

Lemma 5.2.

1. $Sat(M) = \emptyset$ holds for any monotonicity graph M that contains an edge with weight ∞ or a cycle with positive weight sum.
2. $|M|$ is polynomial-time computable from M and $Sat(M) = Sat(|M|)$.
3. If we fix sets Var, Const of variables and constants then for any gap constraint $\mathcal{C}$ there is a unique monotonicity graph $M_{\mathcal{C}}$, containing an edge $x \xrightarrow{k} y$ iff there is a clause $x - y \geq k$ in $\mathcal{C}$. Moreover, $Sat(M_{\mathcal{C}}) = Sat(\mathcal{C})$.

The last point of this lemma states that monotonicity graphs and gap constraints are equivalent formalisms. We call a MG *positive* if it has degree 0. Positive MG are equivalent to positive gap constraints. We thus talk about *transitional* monotonicity graphs over $(\mathit{Var}, \mathit{Const})$ as those with nodes $\mathit{Var} \cup \mathit{Var}' \cup \mathit{Const}$. We further define the following operations on MG.

Definition 5.3. Let M, N be monotonicity graphs over Var, Const and $V \subseteq \mathit{Var}$.

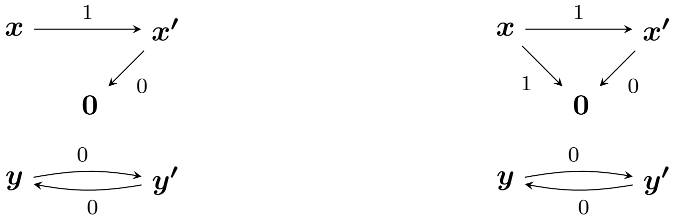
- The *restriction* $M|_V$ of M to V is the maximal subgraph of M with nodes $V \cup \mathit{Const}$.
- The *projection* $\mathit{Proj}(M, V) = |M|_V$ is the restriction of M's closure to V.
- The *intersection* $M \otimes N$ is the MG that contains an edge $x \xrightarrow{k} y$ if k is the maximal weight of any edge from x to y in M or N.
- The *composition* $G \circ M$ of a *transitional* MG G and M is obtained by consistently renaming variables in M to their primed copies, intersecting the result with G and projecting to $\mathit{Var} \cup \mathit{Const}$. $G \circ M := \mathit{Proj}(M_{[\mathit{Var} \mapsto \mathit{Var}']} \otimes G, \mathit{Var})$.

These operations are surely computable in polynomial time. The next lemma states important properties of these operations; see also [6, 5].

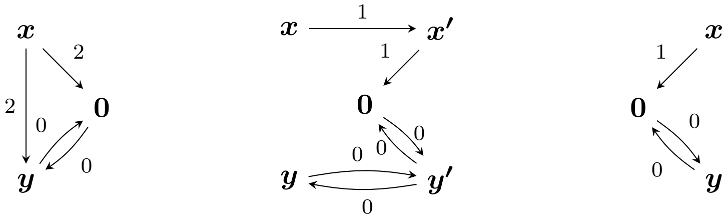
Lemma 5.4.

1. $Sat(Proj(M,V)) = \{\nu|_V : \nu \in Sat(M)\}$.
2. $Sat(M \otimes N) = Sat(M) \cap Sat(N)$
3. $Sat(G \circ M) = \{\nu \mid \exists \nu' \in Sat(M).\ \nu \oplus \nu' \in Sat(G)\} = Pre_G(M)$.
4. If M has degree n and G is a transitional MG of degree 0, then $G \circ M$ has degree $\leq n$.

Example 5.5. The monotonicity graph on the left below corresponds to the contraint $\mathcal{C}X$ in Example 2.3. On the right we see its closure (where edges with weight $-\infty$ are omitted). Both have degree 0.



Let us compute the $\mathcal{C}X$-predecessors of the set $S = \{\nu \mid \nu(x) > \nu(y) = 0\}$ which is characterized by the single MG on the right below.



If we rename variables x and y to x' and y' and intersect the result with $M_{\mathcal{C}X}$ we get the MG in the middle. We project into $Var \cup Const$ by computing the closure and restricting the result accordingly. This leaves us with the MG on the left, which characterizes the set $Pre_{\mathcal{C}X}(S) = \{\nu \mid \nu(x) \geq 2\ \wedge\ \nu(y) = 0\}$ as expected.

We have seen how to construct a representation of the $\mathcal{C}$-predecessors $Pre_{\mathcal{C}}(S)$ and thus $Pre_a(S)$ for a MG-definable set S, gap constraint $\mathcal{C}$ and action $a \in Act$. The next lemma is a consequence of Lemma 5.2, point 3 and asserts that we can do the same for complements.

Lemma 5.6. The class of MG-definable sets is effectively closed under complementation.

Proof:
By Lemma 5.2 we can interpret a finite set of MG $\mathcal{M} = \{M_0, M_1, \ldots, M_k\}$ as a gap formula in DNF. One can then use De Morgan's laws to propagate negations to atomic propositions, which are gap clauses of the form $x - y \geq k$. The negation is then expressible as $x - y < k$, which is equivalent to $y - x > -k$ and thus to the gap clause $y - x \geq -k + 1$. It remains to bring the formula into DNF again, which can then be described by finitely many MGs. □

Observe that complementation potentially constructs MG with increased degree. This next degree is bounded by the largest finite weight in the current graph minus one, but nevertheless, an increase of degree cannot be avoided. Therefore, classes of MG^n-definable sets are *not* closed under complementation.

Example 5.7. The set $S = \{\nu \mid \nu(x) - \nu(y) \geq 5\}$ corresponds to the gap-formula $\varphi = (x - y \geq 5)$. Its MG $\{(x \xrightarrow{5} y)\}$ is of degree 0. However, its complement is characterized by the MG $\{(y \xrightarrow{-4} x)\}$, which has degree 4.

It remains to show that we can compute $Pre^*(S)$ for MG-definable sets S. We recall the following partial ordering on monotonicity graphs and its properties [6].

Definition 5.8. Let M, N be MG over $(Var, Const)$. We say that M *covers* N (write $N \sqsubseteq M$) if for all $x, y \in Var \cup Const$ it holds that $N(x, y) \leq M(x, y)$.

Lemma 5.9.

1. If $N \sqsubseteq M$ then $Sat(N) \supseteq Sat(M)$.

2. For every $n \in \mathbb{N}$, $\sqsubseteq$ is a well-order on the set of MG over $(Var, Const)$ with degree $\leq n$.

Proof:
For the first claim, assume $\nu \in Sat(M) = Sat(|M|)$. Then, for every $x, y \in Var \cup Const$, we have $\nu(x) - \nu(y) \geq M(x, y) \geq N(x, y)$. So $\nu \in Sat(|N|) = Sat(N)$.

The second claim follows from Dickson's Lemma if we interpret each MG M with degree n as $|Var \cup Const|^2$-dimensional vector where the component for the pair (x, y) has value $n + M(x, y)$. □

Note that point 1 states that a $\sqsubseteq$-bigger MG is more restrictive and hence has a smaller denotation. Also notice that $\sqsubseteq$ is *not* a well order on the set of all MG due the lack of a bound on finite, negative weights: for instance, the sequence $(M_n)_{n \in \mathbb{N}}$ of MG, where for every n, the graph M_n has edges $x \xrightarrow{n} y \xrightarrow{-n} x$, is an infinite antichain.

Lemma 5.10. Let S be a MG^n-definable set of valuations. Then $Pre^*(S)$ is MG^n-definable and a representation of $Pre^*(S)$ can be computed from a representation of S.

Proof:
It suffices to show the claim for a set S characterized by a single monotonicity graph M_S, because $Pre^*(S \cup S') = Pre^*(S) \cup Pre^*(S')$. Assume that M_S has degree n.

We proceed by exhaustively building a finite tree of MG, starting in M_S. For every node N we compute children $G \circ N$ for all of the finitely many transitional MG G in the system. Point 4) of Lemma 5.4 guarantees that all intermediate representations have degree $\leq n$. By Lemma 5.9, point 2, any branch eventually ends in a node that covers a previous one and Lemma 5.9, point 1 allows us to stop exploring such a branch. We conclude that $Pre^*(M)$ can be characterized by the finite union of all intermediate MG. □

Finally, we are ready to prove our main result.

Theorem 5.11. EF model checking is decidable for Gap-order constraint systems. Moreover, the set $[\![\psi]\!]$ of valuations satisfying an EF-formula ψ is effectively gap definable.

Proof:
We can evaluate a formula bottom up, representing the sets satisfying subformulae by finite sets of MG. Atomic propositions are either $true$ or gap clauses and can thus be written directly as MG. For composite formulae we use the properties that MG definable sets are effectively closed under intersection (Lemma 5.4) and negation (Lemma 5.6), and that we can compute representations of $Pre_a(S)$ and $Pre^*(S)$ for MG-definable sets S by Lemmas 5.4 and 5.10.

The key observation is that although negation (i.e., complementing) may increase the degree of the intermediate MG, this happens only finitely often in the bottom up evaluation of an EF formula. Computing representations for modalities $\langle a \rangle$ and *EF* does not increase the degree. □

Remark 5.12. Since steps in GCS are described by positive transitional gap constraints, it is straightforward to extend this positive result of Theorem 5.11 to model checking GCS w.r.t. the slightly more general logic EF$_{\mathcal{C}}$, in which the next-state and reachability modalities $\langle a \rangle_{\mathcal{C}}$ and *EF*$_{\mathcal{C}}$ are subject to transitional gap clauses $\mathcal{C}$.

The exact complexity of the EF model checking problem for GCS is still open. However, a PSPACE lower bound already holds for reachability of the simpler model of *boolean programs*. Moreover, a rather crude Ackermannian upper complexity bound on EF model checking for GCS can be obtained by bounding “bad sequences” in our use of Dickson’s Lemma. In the remainder of this section we will elaborate on these reductions.

5.1. A PSPACE lower complexity bound

Several equivalent notions of boolean programs are used in different application domains (see e.g., [7, 2]). Essentially, they consist of a finite control unit that manipulates finitely many boolean variables.

We define *boolean programs* as finite-state machines with transitions of the form $s \xrightarrow{g(\vec{x})/a} t$, where s and t are control-states, a is an assignment $x = 0$ or $x = 1$ for some variable x, and $g(\vec{x})$ is a boolean formula with free variables $\vec{x}$. The semantics of boolean programs is given by the binary step relation $\longrightarrow$ over pairs of control-states and variable valuations. Let $Var = \{x_1, x_2, \ldots, x_k\}$ be the set of variables in the system and let $\nu, \nu' : Var \to \{0, 1\}$ be two valuations. A transition $s \xrightarrow{g(\vec{x})/y=b} t$ induces a step $s, \nu \longrightarrow t, \nu'$ if 1) $\nu \models g$, 2) $\nu'(y) = b$ and 3) $\nu'(x) = \nu(x)$ for $x \in Var \setminus \{y\}$.

The *state-reachability problem* for boolean programs asks, for two given control states s and t, whether there exists a valuation ν and a finite number of steps from s, ν_0 to t, ν. Here, $\nu_0 : x \mapsto 0$ assigns the value 0 to every variable. We will show that this problem is PSPACE hard, by reduction from the PSPACE-complete satisfiability problem for *quantified boolean formulae* (QBF). Notice that boolean programs can be directly simulated by gap-order constraint systems. The same lower bound thus holds for the reachability problem, and consequently also for the EF model checking problem for GCS.

Let $Q_1 x_1 Q_2 x_2 \ldots Q_k x_k \varphi$ be a QBF formula in prenex normal form. We construct a boolean program that contains control-states $eval_i$ and out_i as well as variables x_i for each $1 \leq i \leq k$. The program evaluates the formula top down: a subformula $\varphi_i = Q_i x_i, Q_{i+1} x_{i+1} \ldots Q_k x_k \varphi$ is verified by a run from control-state $eval_i$ to out_i. If the current valuation does not satisfy the subformula, the program deadlocks and out_i is not reachable.

The quantifier-free subformula φ is directly evaluated using a transition $eval_{k+1} \xrightarrow{\varphi/y_{k+1}=1} out_{k+1}$. Notice that a pair $eval_i, \nu$ is a deadlock unless $\nu \models \varphi$. For an existential quantifier Q_i, there are transitions

$$eval_i \xrightarrow{/x_i=0} eval_{i+1}, \quad eval_i \xrightarrow{/x_i=1} eval_{i+1}, \quad out_{i+1} \longrightarrow out_i$$

For a universal quantifier Q_i, there is an extra variable y_i and transitions

$$\begin{aligned} &eval_i \xrightarrow{/x_i=0} eval_{i+1}, && eval_i \xrightarrow{y_i=1/x_i=1} eval_{i+1},\\ &out_{i+1} \xrightarrow{/y_i=1} eval_i, && out_{i+1} \xrightarrow{y_i=1/y_i=0} out_i. \end{aligned}$$

Notice that the variable y_i serves as a flag to indicate that the subformula φ_i has been successfully verified for value $x_i = 0$. Just observe that for any valuation ν with $\nu(y_i) = 0$, there is a path from $eval_i, \nu$ to some out_i, ν' iff $\nu_{[x_i=0]} \models \varphi_i$ and $\nu_{[x_i=1]} \models \varphi_i$ Moreover, the existence of such a path implies that $\nu'(y_i) = 0$. An induction on i shows that the given formula is indeed satisfiable if, and only if, there exists ν such that $eval_1, \nu_0 \xrightarrow{*} out_1, \nu$.

5.2. An Ackermann upper complexity bound

Due to our use of Dickson's Lemma in Lemma 5.10, we can derive an Ackermannian upper bound for EF model checking using the approach of Schmitz et.al. [10]. We show how to to bound the size of our representation of $Pre^*(S)$ in terms of fast-growing functions. This implies that the space required by the procedure of Theorem 5.11 can be bounded by an Ackermannian function.

The family of *fast-growing functions* $F_n : \mathbb{N} \to \mathbb{N}$ is inductively defined as follows for all $x, k \in \mathbb{N}$.

$$F_0(x) = x + 1 \qquad \text{and} \qquad F_{k+1}(x) = F_k^{x+1}(x).$$

A variant of the Ackermann function is $F_\omega : \mathbb{N} \to \mathbb{N}$, defined as $F_\omega(x) = F_x(x)$.

Consider d-dimensional tuples of natural numbers with the pointwise ordering $\leq$. A sequence $x_0 x_1 \dots x_l \in (\mathbb{N}^d)^*$ of tuples is called *good* if there exist indices $0 \leq i < j \leq l$ such that $x_i \leq x_j$ and *bad* otherwise. I.e., a bad sequence is an antichain w.r.t. the ordering on the tuples. By Dickson's Lemma, every bad sequence is finite, but there exist bad sequences of arbitrary length, because there is no assumption on the increase in one dimension if another dimension decreases. The *norm* of $x \in \mathbb{N}^d$ is $\|x\|_\infty = \max\{x(i) \mid 0 \leq i \leq d\}$. The sequence is *t-controlled* by a function $f : \mathbb{N} \to \mathbb{N}$ if $\|x_i\|_\infty < f(i + t)$ for every index $0 \leq i \leq l$. Let $L_{d,f}(t)$ denote the maximal length of a bad sequence in $\mathbb{N}^d$ that is t-controlled by f.

Schmitz et.al. [10], show how to bound such controlled bad sequences in terms of fast-growing functions. It follows from their work that for every $d \geq 1$ and $c, k, x \in \mathbb{N}$,

$$L_{d,F_k^c}(t) \leq F_{k+d-1}^{(c+d+2)^d}(t). \tag{21}$$

We are now ready to bound the size of computed representations of $Pre^*(S)$ for a given MG-definable sets S. Fix a GCS with variables Var, constants $Const$ and δ-many transitional gap constraints. For the sake of readability we assume an unlabeled GCS; the bounds we provide directly apply for the labeled case as well. Recall that a satisfiable monotonicity graph M has the property that

$M(x,y) < \infty$ for all $x, y \in \mathit{Var} \cup \mathit{Const}$. We identify such a graph with the vector $v_M \in \mathbb{N}^d$ of dimension $d = |(\mathit{Var} \cup \mathit{Const})^2|$, where the component for the pair (x,y) has value 0 if $M(x,y) = -\infty$ and $n + M(x,y) + 1$ otherwise. In particular, notice that $\|M\|_\infty$ is bounded by $n + 1 + \max\{M(x,y) \mid x, y \in \mathit{Var} \cup \mathit{Const}\}$.

Let S be a MG-definable set of valuations represented by a single monotonicity graph and consider a branch of the tree constructed in the proof of Lemma 5.10. It provides a bad sequence $M_0 M_1 \ldots M_l$ where for each $0 < i$, the graph M_i is the result of composing its predecessor M_{i-1} with one of the transitional monotonicity graphs G of the system. Wlog., assume that all M_i are satisfiable, because otherwise it (and with it all M_j for $i \leq j \leq l$) represents the empty set and does not contribute to $\mathit{Pre}^*(S)$. By definition of compositions $G \circ M$ (see Definition 5.3) we therefore get $\|M_i\|_\infty \leq \|M_{i-1}\|_\infty + c$, for every $0 < i \leq l$, where c is the maximal constant in the system. Consequently, the branch is $\|M_0\|_\infty$-controlled by $f : x \mapsto xc$. Since f is dominated by F_1^c, equation (21) provides the bound

$$l \;\leq\; L_{d,f}(t) \;\leq\; L_{d,F_1^c}(t) \;\leq\; F_d^{(c+d+2)^d}(t) \tag{22}$$

on the length of the branch, where $t = \|M_0\|_\infty$. If we instead let $t = \max\{\|M_0\|_\infty, (c+d+2)^d + \delta\}$, then we can bound l by $F_d^{(c+d+2)^d}(t) \leq F_d^{t+1}(t) = F_{d+1}(t)$. In particular, this means that the norm $\|M_l\|_\infty$ is bounded by $c \cdot F_{d+1}(t) \leq F_{d+2}(t)$. Moreover, since $F_2^n(x) = x^n + x$, the total number of nodes in the tree is bounded by $\delta^l \leq F_2 F_{d+1}(t) \leq F_{d+3}(t)$. We have shown the following lemma.

Lemma 5.13. Let S be a set of valuations represented by m monotonicity graphs and let $t \geq (c + d + 2)^d + \delta$ be an upper bound on their norms. Then $\mathit{Pre}^*(S)$ can be effectively represented by no more than $m \cdot F_{d+3}(t)$ graphs with norm at most $F_{d+2}(t)$.

The claim of the next lemma directly follows from Definitions 5.1, 5.3, Lemma 5.6 as well as the definition of the vector v_M representing the MG M. Notice that complementing a single MG M results in at most d graphs of degree $< \|v_M\|_\infty$. Each of them therefore corresponds to a vector with norm bounded by $2\|v_M\|_\infty + 1 = F_1(\|v_M\|_\infty)$.

Lemma 5.14. Let S, S' be sets of valuations, each represented by a set of m monotonicity graphs of norm at most t. Then,

1. $S \cup S'$ can be effectively represented by $m + m$ graphs of norm at most t,
2. $S \cap S'$ can be effectively represented by $m \cdot m$ MG of norm at most t,
3. $\mathit{Pre}_a(S)$ can be effectively represented by m graphs of norm at most $t + c$, where c is the maximal absolute value of any constant in the system,
4. $\mathit{Val} \setminus S$ can be effectively represented by d^m graphs of norm at most $F_1(t)$.

Proposition 5.15. Let $(\mathit{Var}, \mathit{Const}, \mathit{Act}, \Delta, \lambda)$ be a gap-order constraint system and let $d = (|\mathit{Var}| + |\mathit{Const}|)^2$, $\delta = |\Delta|$ and $c = \max\{|x| : x \in \mathit{Const}\}$. For every EF-formula φ of nesting depth k, one can effectively compute a representation of the set $[\![\varphi]\!]$, in space $F_{d+4}((c + d + 2)^d + \delta + k)$.

Proof:
Let $t = (c + d + 2)^d + \delta)$. We show by induction on the nesting depth k of subformulae that $[\![\varphi]\!]$ can be represented by at most $F^k_{d+3}(t)$ monotonicity graphs with norms bounded by $F^k_{d+2}(t)$.

For the base case, observe that atomic propositions are either $\varphi = true$ or stated as single gap constraint $\varphi = \mathcal{C}$. Either way, $[\![\varphi]\!]$ can be expressed as single monotonicity graph M_φ with norm $\|M_\varphi\|_\infty \leq c \leq F^0_{d+2}((c + d + 2)^d + \delta)$.

For the induction step, we assume that the claim is true for all formulae of height i and consider a formula φ of height $i + 1$. If the principal connector of φ is $\wedge, \vee, \neg$ or $\langle a \rangle$ for some action a, then the claim follows by Lemma 5.14. To see this, just notice that for all $m \in \mathbb{N}$, $F_{d+2}(m) \geq F_1(m) = 2m + 1$ and $F_{d+3}(m) \geq F_3(m) \geq m^m$. If φ is of the form $EF\phi$, then, by induction hypothesis, $[\![\phi]\!]$ can be effectively represented by no more than $F^i_{d+3}(t)$ graphs with norm $\leq F^i_2(t)$. Lemma 5.13 thus implies that $[\![\varphi]\!]$ is representable by $F^{i+1}_{d+3}(t)$ graphs, each with norm at most $F^{i+1}_{d+2}(t)$ as required.

The claim now follows from the observation that the total space required for the above representation is $d \cdot \log F^k_{d+2}(t) \cdot F^k_{d+3}(t) \leq F_{d+4}(t + k)$. □

6. Applications

We consider labeled transition systems induced by GCS. In a weak semantics, one abstracts from non-observable actions modeled by a dedicated action $\tau \in Act$. The *weak step* relation $\Longrightarrow$ is defined by

$$\begin{aligned} \overset{\tau}{\Longrightarrow} \quad &= \quad \xrightarrow{\tau}^* \\ \overset{a}{\Longrightarrow} \quad &= \quad \xrightarrow{\tau}^* \cdot \xrightarrow{a} \cdot \xrightarrow{\tau}^*, \quad \text{for } a \neq \tau \end{aligned}$$

Bisimulation and weak bisimulation [18, 16] are semantic equivalences in van Glabbeek's linear time – branching time spectrum [12], which are used to compare the behavior of processes. Their standard co-inductive definition relative to a given LTS is as follows.

Definition 6.1. A binary relation $R \subseteq V^2$ on the states of a labeled transition system is a *bisimulation* if sRt implies that

1. for all $s \xrightarrow{a} s'$ there is a t' such that $t \xrightarrow{a} t'$ and $s'Rt'$, and
2. for all $t \xrightarrow{a} t'$ there is a s' such that $s \xrightarrow{a} s'$ and $s'Rt'$.

Similarly, R is a *weak bisimulation* if in both conditions above, $\longrightarrow$ is replaced by $\Longrightarrow$. (Weak) bisimulations are closed under union, so there exist unique maximal bisimulation $\sim$ and weak bisimulation $\approx$ relations, which are equivalences on V.

By the maximal (weak) bisimulation between two LTS with state sets S and T we mean the maximal (weak) bisimulation in their union projected into $(S \times T) \cup (T \times S)$.

The *Equivalence Checking Problem* is the following decision problem.

INPUT: Given LTS $T_1 = (V_1, Act, \longrightarrow)$ and $T_2 = (V_2, Act, \longrightarrow)$,
states $s \in V_1$ and $t \in V_2$ and an equivalence R.

QUESTION: sRt?

In particular, we are interested in checking strong and weak bisimulation between LTS induced by GCS and finite systems. Note that the decidability of weak bisimulation implies the decidability of the corresponding strong bisimulation because $\sim$ and $\approx$ coincide for LTS without τ labels.

Finite systems admit *characteristic formulae* up to weak bisimulation in EF (see e.g. [14, 13]).

Theorem 6.2. Let $T_1 = (V_1, Act, \longrightarrow)$ be an LTS with finite state set V_1 and $T_2 = (V_2, Act, \longrightarrow)$ be an arbitrary LTS. For every state $s \in V_1$ one can construct an EF-formula ψ_s such that $t \approx s \iff t \models \psi_s$ for all states $t \in V_2$.

The following is a direct consequence of Theorems 6.2 and 5.11.

Theorem 6.3. For every GCS $\mathcal{G} = (Var, Const, Act, \Delta, \lambda)$ and every LTS $T = (V, Act, \longrightarrow)$ with finite state set V, the maximal weak bisimulation $\approx$ between $T_\mathcal{G}$ and T is effectively gap definable.

Proof:
By Theorem 6.2 we can compute, for every state s of T, a characteristic formula ψ_s that characterizes the set of valuations $\{\nu \mid \nu \approx s\} = [\![\psi_s]\!]$. By Theorem 5.11, these sets are MG- and thus gap definable. Since the class of gap definable sets is effectively closed under finite union and $\approx\, = \bigcup_{s \in V} [\![\psi_s]\!]$, the result follows. □

Considering that gap formulae are particular formulae of Presburger arithmetic, we know that gap definable sets have a decidable membership problem. Theorem 6.3 thus implies the decidability of equivalence checking between GCS processes and finite systems w.r.t. strong and weak bisimulation.

7. Conclusion and Open Questions

We have shown that model checking gap-order constraint systems with the logic EG is undecidable while the problem remains decidable for the logic EF. An immediate consequence of the latter result is the decidability of strong and weak bisimulation checking between GCS and finite systems.

The decidability of EF model checking is shown by using finite sets of monotonicity graphs or equivalently, gap formulae to represent intermediate results in a bottom-up evaluation. This works because the class of arbitrary gap definable sets is effectively closed under union and complements and for a gap definable set S and a GCS $\mathcal{G}$, $Pre(S)$ and $Pre^*(S)$ are effectively gap definable.

Our decidability result relies on a well-quasi-ordering argument to ensure termination of the fixpoint computation for $Pre^*(S)$, which does not yield any strong upper complexity bound. So far, there is only an Ackermannian upper bound and a PSPACE lower bound.

Interesting open questions include determining the exact complexity of model checking GCS with respect to EF. We also plan to investigate the decidability and complexity of checking behavioral equivalences like strong and weak bisimulation between two GCS processes, as well as checking (weak) simulation preorder and trace inclusion.

References

[1] P A Abdulla and G Delzanno. Constrained multiset rewriting. In *Proc. AVIS06, 5th int. workshop on Automated Verification of InfiniteState Systems*. 2006.

[2] Thomas Ball and Sriram K. Rajamani. Boolean programs: A model and process for software analysis. Technical Report MSR-TR-2000-14, Microsoft Research, February 2000.

[3] Amir M. Ben-Amram. Size-change termination, monotonicity constraints and ranking functions. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification*, volume 5643 of *LNCS*, pages 109–123. Springer, 2009.

[4] Laura Bozzelli. Strong termination for gap-order constraint abstractions of counter systems. In *LATA*, volume 7183 of *LNCS*, pages 155–168. 2012.

[5] Laura Bozzelli and Sophie Pinchinat. Verification of gap-order constraint abstractions of counter systems. In *VMCAI*, volume 7148 of *LNCS*, pages 88–103. Springer, 2012.

[6] Karlis Cerans. Deciding properties of integral relational automata. In *ICALP*, volume 820 of *LNCS*, pages 35–46. 1994.

[7] Stephen Cook and Michael Soltys. Boolean programs and quantified propositional proof systems. *Bulletin of the Section of Logic*, 1999.

[8] Stéphane Demri and Deepak D'Souza. An automata-theoretic approach to constraint ltl. *Inf. Comput.*, 205(3):380–415, 2007.

[9] Javier Esparza. Decidability of model checking for infinite-state concurrent systems. *Acta Informatica*, 34:85–107, 1997.

[10] Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. Ackermannian and primitive-recursive bounds with dickson's lemma. In *LICS*, pages 269–278, 2011.

[11] Laurent Fribourg and Julian Richardson. Symbolic verification with gap-order constraints. In *LOPSTR*, volume 1207 of *LNCS*, pages 20–37. 1996.

[12] R.J. van Glabbeek. The linear time – branching time spectrum I; the semantics of concrete, sequential processes. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, chapter 1, pages 3–99. Elsevier, 2001.

[13] Petr Jančar, Antonín Kučera, and Richard Mayr. Deciding bisimulation-like equivalences with finite-state processes. In *ICALP*, volume 1443 of *LNCS*, pages 200–211. 1998.

[14] Antonín Kučera and Petr Jančar. Equivalence-checking on infinite-state systems: Techniques and results. *TPLP*, 6(3):227–264, 2006.

[15] Ernst W. Mayr. An algorithm for the general petri net reachability problem. *SIAM J. Comput.*, 13(3):441–460, 1984.

[16] Robin Milner. *Communication and concurrency*. PHI Series in computer science. Prentice Hall, 1989.

[17] Marvin L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.

[18] David Park. Concurrency and automata on infinite sequences. In *TCS*, volume 104 of *LNCS*, pages 167–183. 1981.

[19] Peter Z. Revesz. A closed-form evaluation for datalog queries with integer (gap)-order constraints. *TCS*, 116(1&2):117–149, 1993.