Branching-Time Model Checking Gap-Order Constraint Systems

Branching-Time Model Checking Gap-Order Constraint Systems

Richard Mayr and Patrick Totzke

University of Edinburgh, 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 [Min67] 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 [Min67]. 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 [May84].

Gap-order constraint systems [Boz12,BP12] 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 [AD06].

Previous work. Beyond reachability, several model checking problems have been studied for GCS and related formalisms. The paper [Cer94] studies Integral Relational Automata (IRA), a model that is subsumed by GCS. It is shown that CTL model checking of IRA is undecidable in general, but the existential and universal fragments of CTL^* remain decidable for IRA. Termination and safety properties for GCS were studied in [Boz12,BP12]. In particular, LTL model checking

is PSPACE complete for GCS. Moreover, model checking GCS is decidable for the logic $ECTL^*$, but undecidable for $ACTL^*$, which are the existential and universal fragments of CTL^* respectively. These fragments do not allow arbitrary nesting of path quantifiers and negation and are therefore orthogonal to **EF** and **EG**, which allow nesting of negation and operators EF/EG but forbid the respective other operator.

Our contribution. We study the decidability of model checking problems for GCS with fragments of computation-tree logic (CTL), namely **EF** and **EG**. While general CTL model checking of GCS is undecidable (even for the weaker model of IRA [Cer94]), we show that it is decidable for the logic **EF**. On the other hand, model checking is undecidable for the logic **EG**. An immediate consequence of our decidability result for **EF** is that strong and weak bisimulation equivalence are 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 $Pre_a(S) = \{s' | s' \xrightarrow{a} s \in S\}$ and let $Pre^*(S) = \{s' | s' \longrightarrow^* s \in S\}$. A state in a LTS is often referred to as a *process*.

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 *evaluations* $\nu : Var \to \mathbb{Z}$. To simplify the notation, we will sometimes extend the domain of evaluations to constants, where they behave as the identity, i.e., $\nu(c) = c$ for all $c \in \mathbb{Z}$.

Definition 1 (Gap-Constraints). *A* gap clause *over $(Var, Const)$ is an inequation of the form*

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

where $x, y \in Var \cup Const$ 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.*

An evaluation $\nu : Var \to \mathbb{Z}$ 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 $Sat(\varphi) = \{\nu \in Val \mid \nu \models \varphi\}$ for the set of evaluations that satisfy the formula φ. In particular, an evaluation satisfies a gap constraint iff it satisfies all its clauses. A set $S \subseteq Val$ of evaluations is called gap-definable *if there is a gap formula φ with $S = Sat(\varphi)$.*

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

Let $Var' = \{x' | x \in Var\}$ 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 evaluation to another: x' is interpreted as the next value of variable x. A *transitional* gap-order clause (-constraint, -formula) is a gap-order clause (-constraint, -formula) with variables in $Var \cup Var'$.

For evaluations $\nu : Var \to \mathbb{Z}$ and $\nu' : Var \to \mathbb{Z}$ we define the combined evaluation $\nu \oplus \nu' : Var \cup Var' \to \mathbb{Z}$ of variables in $Var \cup Var'$ by

$$\nu \oplus \nu'(x) = \begin{cases} \nu(x), & \text{if } x \in Var \\ \nu'(x), & \text{if } x \in Var'. \end{cases} \tag{3}$$

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

$$\nu \xrightarrow{a} \nu' \iff \nu \oplus \nu' \models \mathcal{C} \tag{4}$$

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

Observe that a constraint $(x - 0 \geq 0) \ \wedge \ (0 - x \geq 0)$ is satisfied only by evaluations assigning value 0 to variable x. Similarly, one can test if an evaluation 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).

Example 1. 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 $\lambda(\mathcal{C}Y) = b$.

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

$$\mathcal{C}Y = ((y - y' \geq 1) \ \wedge \ (x' - x \geq 0) \ \wedge \ (y' - 0 \geq 0)). \tag{6}$$

[1] In fact, [BP12,Boz12] consider an equivalent notion of GCS that explicitly includes a finite control.

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 evaluations satisfying this 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.

$$\psi ::= \mathcal{C} \mid true \mid \neg\psi \mid \psi \wedge \psi \mid \langle a \rangle \psi \mid EF\psi \mid EG\psi \mid E(\psi U \psi) \tag{7}$$

For the semantics, let $Paths^{\omega}(\nu_0)$ be the set of infinite derivations

$$\pi = \nu_0 \xrightarrow{a_0} \nu_1 \xrightarrow{a_1} \nu_2 \ldots \tag{8}$$

of $\mathcal{G}$ starting with evaluation $\nu_0 \in Val$ and $\pi(i) = \nu_i$ the ith evaluation ν_i on π. Similarly, we write $Paths^{*}(\nu_0)$ for the set of finite paths from ν_0 and let $Paths(\nu_0) = Paths^{\omega}(\nu_0) \cup Paths^{*}(\nu_0)$. The denotation of composite formulae is defined in the standard way.

$$[\![\mathcal{C}]\!] = Sat(\mathcal{C}) \tag{9}$$
$$[\![true]\!] = Val \tag{10}$$
$$[\![\neg\psi]\!] = Val \setminus [\![\psi]\!] \tag{11}$$
$$[\![\psi_1 \wedge \psi_2]\!] = [\![\psi_1]\!] \cap [\![\psi_2]\!] \tag{12}$$
$$[\![\langle a \rangle \psi]\!] = Pre_a([\![\psi]\!]) \tag{13}$$
$$[\![EF\psi]\!] = \{\nu \mid \exists \pi \in Paths^{*}(\nu).\ \exists i \in \mathbb{N}.\ \pi(i) \in [\![\psi]\!]\} \tag{14}$$
$$[\![EG\psi]\!] = \{\nu \mid \exists \pi \in Paths^{\omega}(\nu).\ \forall i \in \mathbb{N}.\ \pi(i) \in [\![\psi]\!]\} \tag{15}$$
$$[\![E(\psi_1 U \psi_2)]\!] = \{\nu \mid \exists \pi \in Paths(\nu).\ \exists i \in \mathbb{N}.\ \pi(i) \in [\![\psi_2]\!] \wedge \forall j < i.\pi(j) \in [\![\psi_1]\!]\} \tag{16}$$

We use the usual syntactic abbreviations $false = \neg true$, $\psi_1 \vee \phi_2 = \neg(\neg\psi_1 \wedge \neg\phi_2)$.

The sublogics **EF** and **EG** are defined by restricting the grammar (7) defining well-formed formulae: **EG** disallows subformulae of the form $E(\psi_1 U \psi_2)$ and $EF\psi$ and in **EF**, no subformulae of the form $E(\psi_1 U \psi_2)$ or $EG\psi$ are allowed. The *Model Checking Problem* is the following decision problem.

INPUT: A GCS $G = (Var, Const, Act, \Delta, \lambda)$, an evaluation $\nu : Var \to \mathbb{Z}$ and a formula ψ.
QUESTION: $\nu \models \psi$?

Cerans [Cer94] showed that general CTL model checking is undecidable for gap-order 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 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 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 [Min67].

Given a 2CM M, we will construct a GCS together with an initial evaluation 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 implement 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 $p \overset{x_1 = x_1 + 1}{\longrightarrow} q$ 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 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

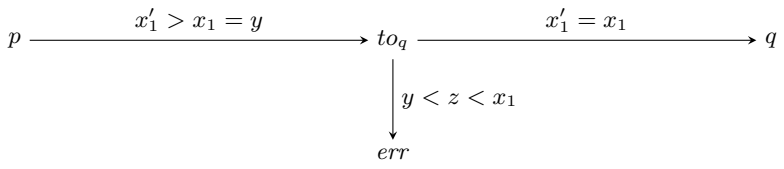


Fig. 1. Forcing faithful simulation of x_1-increment. All steps contain the additional constraint $x_2' = x_2$, which is not shown, to preserve the value of the other counter x_2.

thus translated to the following three constraints.

$$(p \longrightarrow to_q) \wedge (x_1' > x_1 = y) \wedge (x_2' = x_2) \tag{17}$$

$$(to_q \longrightarrow q) \wedge (x_1' = x_1) \wedge (x_2 = x_2) \tag{18}$$

$$(to_q \longrightarrow err) \wedge (y < z < x_1). \tag{19}$$

If we translate all operations of the 2CM into the GCS formalism as indicated above, we end up with an overapproximation 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

$$\phi = EG(\neg halt \ \wedge \ \neg \langle a \rangle err) \tag{20}$$

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 φ is satisfied by evaluation $\nu_0 = \{state = [init], x_1 = x_2 = y = z = 0\}$ 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. □

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 called *monotonicity graphs* (MG) which are finite graphs with nodes $Var \cup Const$.[2]

Monotonicity graphs can be used to represent sets of evaluations. 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.

[2] These were called *Graphose Inequality Systems* in [Cer94].

Definition 3 (Monotonicity Graphs). *A* monotonicity graph *(MG) over* $(Var, Const)$ *is a finite, directed graph* $M = (V, E)$ *with nodes* $V = Var \cup Const$ *and edges labeled by elements of* $\mathbb{Z} \cup \{-\infty, \infty\}$.

An evaluation $\nu : Var \to \mathbb{Z}$ *satisfies* M *if for every edge* $(x \xrightarrow{k} y)$ *it holds that* $\nu(x) - \nu(y) \geq k$. *We write* $\nu \models M$ *in this case and let* $Sat(M)$ *denote the set of evaluations satisfying* M.

Let $M(x, y) \in \{-\infty, \infty\} \cup \mathbb{Z}$ *denote the least upper bound of the sums of the labels on any path from node* x *to node* y. *The* closure $|M|$ *is the unique complete monotonicity graph with edges* $x \xrightarrow{M(x,y)} y$ *for all* $x, y \in Var \cup Const$.

The degree *of* M *is the smallest* $K \in \mathbb{N}$ *such that* $k = -\infty$ *or* $k \geq -K$ *for all edge labels* k *in* M. *It is the negation of the smallest finite negative label or* 0 *if no such label exists.*

The following lemma states some basic properties of monotonicity graphs that can easily be verified.

Lemma 1.

1. $Sat(M) = \emptyset$ *for any monotonicity graph* M *that contains an edge labeled by* ∞ *or some 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 MG* $M_{\mathcal{C}}$ *containing an edge* $x \xrightarrow{k} y$ *iff there is a clause* $x - y \geq k$ *in* $\mathcal{C}$, *for which* $Sat(M_{\mathcal{C}}) = Sat(\mathcal{C})$.

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

Definition 4. *Let* M, N *be monotonicity graphs over* $Var, Const$ *and* $V \subseteq Var$.

- *The* restriction $M|_V$ *of* M *to* V *is the maximal subgraph of* M *with nodes* $V \cup Const$.
- *The* projection $Proj(M, V) = |M|_V$ *is the restriction of* M*'s closure to* V.
- *The* intersection $M \otimes N$ *is the MG that contains and edge* $x \xrightarrow{k} y$ *if* k *is the maximal label 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* $Var \cup Const$. $G \circ M := Proj(M_{[Var \mapsto Var']} \otimes G, Var)$.

These operations are surely computable in polynomial time. The next lemma states important properties of these operations; see also [Cer94,BP12].

Lemma 2.

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 $\leq n$.*

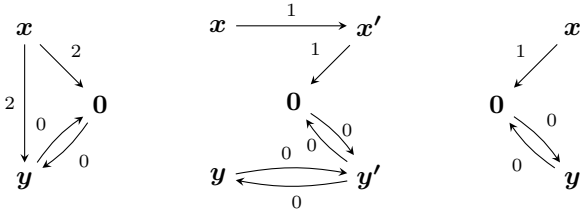
We will use monotonicity graphs to finitely represent sets of evaluations. To that end, let us call a set $S \subseteq Val$ *MG-definable* if there is a finite set $\{M_0, M_1, \ldots, M_k\}$ of MG such that

$$S = \bigcup_{0 \leq i \leq k} Sat(M_i). \tag{21}$$

Call S MG^n-definable if there is such a set of MG with degree $\leq n$.

Example 2. The monotonicity graph on the left below corresponds to the constraint $\mathcal{C}X$ in Example 1. On the right we see its closure (where edges labeled by $-\infty$ 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 MG-definable set S, gap-constraints $\mathcal{C}$ and actions $a \in Act$. The next lemma is a consequence of Lemma 1, point 3 and asserts that we can do the same for complements.

Lemma 3. *The class of MG-definable sets is effectively closed under complements.*

Proof. By Lemma 1 we can interpret a set $\mathcal{M} = \{M_0, M_1, \ldots, M_k\}$ as gap-formula in DNF. One can then use De Morgan's laws to propagate negations to atomic propositions, which are gap-clauses $x - y \geq k$ for which the negation is expressible as gap-clause $y - x \geq -(k+1)$. It remains to bring the formula into DNF again, which can be interpreted as set of MG. □

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

Example 3. The set $S = \{\nu \mid \nu(x) > \nu(y) = 0\}$ from the previous example corresponds to the gap-formula $\varphi_S = (x - 0 \geq 1) \wedge\ (0 - y \geq 0)\ \wedge\ (y - 0 \geq 0)$. Its complement is characterized by the set $S^c = \{(0 \xrightarrow{-2} x), (y \xrightarrow{-1} 0), (0 \xrightarrow{-1} y)\}$, which contains a MG of degree 2.

It remains to show that we can compute $Pre^*(S)$ for MG-definable sets S. We recall [Cer94] the following partial ordering on monotonicity graphs and its properties.

Definition 5. *Let M, N be MG over $(Var, Const)$. We say 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 4.

1. *If $N \sqsubseteq M$ then $Sat(N) \supseteq Sat(M)$.*
2. *$\sqsubseteq$ is a WQO on $\mathbf{MG}^n$ for every fixed $n \in \mathbb{N}$.*

Note that point 1 states that a $\sqsubseteq$-bigger MG is more restrictive and hence has a smaller denotation. Also note that $\sqsubseteq$ is **not** a well ordering on the set of all MG, because edges may be labeled with arbitrary integers (and hence ever smaller negative ones).

Lemma 5. *Let S be a MG^n-definable set of evaluations. 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 characterized by a single monotonicity graph M because $Pre^*(S \cup S') = Pre^*(S) \cup Pre^*(S')$. Assume that M has degree n.

We proceed by exhaustively building a tree of MG, starting in M. 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 the Lemma 2 guarantees that all intermediate representations have degree $\leq n$. By Lemma 4, point 2, any branch eventually ends in a node that covers a previous one and Lemma 4, 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 2. ***EF*** *Model checking is decidable for Gap-order constraint systems. Moreover, the set* $[\![\psi]\!]$ *of evaluations 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 gap-definable sets are effectively closed under intersection (Lemma 2) and negation (Lemma 3), and that we can compute representations of $Pre_a(S)$ and $Pre^*(S)$ for MG-definable sets S by Lemmas 2 and 5.

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. □

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

$$\stackrel{\tau}{\Longrightarrow} = \stackrel{\tau}{\longrightarrow}^*, \text{ and for } a \neq \tau, \qquad \stackrel{a}{\Longrightarrow} = \stackrel{\tau}{\longrightarrow}^* \cdot \stackrel{a}{\longrightarrow} \cdot \stackrel{\tau}{\longrightarrow}^*.$$

Bisimulation and weak bisimulation are semantic equivalences in van Glabbeeks linear time – branching time spectrum [Gla01], which are used to compare the behavior of processes. Their standard co-inductive definition is as follows.

Definition 6. *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* $s \stackrel{a}{\longrightarrow} s'$ *there is a* t' *such that* $t \stackrel{a}{\longrightarrow} t'$ *and* $s'Rt'$*, and*
2. *for all* $t \stackrel{a}{\longrightarrow} t'$ *there is a* s' *such that* $s \stackrel{a}{\longrightarrow} s'$ *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*.*

Let the maximal (weak) bisimulation between two LTS with state sets S *and* T *be 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 processes of GCS and finite systems. Note that the decidability of weak

bisimulation implies the decidability of the corresponding strong bisimulation as $\sim$ and $\approx$ coincide for LTS without τ labels.

We recall (see e.g. [KJ06,JKM98]) that finite systems admit characteristic formulae up to weak bisimulation in **EF**.

Theorem 3. *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 3 and 2.

Theorem 4. *For every GCS $\mathcal{G} = (Var, Const, Act, \Delta, \lambda)$ and every LTS $T = (V, Act, \longrightarrow)$ with finite state set V, the maximal bisimulation $\approx$ between $T_{\mathcal{G}}$ and T is effectively gap-definable.*

Proof. By Theorems 3 we can compute, for every state s of T, a characteristic formula ψ_s that characterizes the set of evaluations $\{\nu \mid \nu \approx s\} = [\![\psi_s]\!]$. By Theorem 2 these sets are MG- and thus gap-definable. Since the class of gap-definable sets is effectively closed under finite unions 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 4 thus implies the decidability of equivalence checking between GCS processes and finite systems w.r.t. strong and weak bisimulation.

7 Conclusion

We have shown that model checking gap-order 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 one can compute finite representations of $Pre(S)$ and $Pre^*(S)$ for gap-definable sets S.

Our decidability result relies on a well-quasi-ordering argument to ensure termination of the fixpoint computation for $Pre^*(S)$, and therefore does not yield any meaningful upper complexity 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

[AD06] P A Abdulla and G Delzanno. Constrained multiset rewriting. In *Proc. AVIS06, 5th int. workshop on on Automated Verification of InfiniteState Systems*, 2006.

[Boz12] Laura Bozzelli. Strong termination for gap-order constraint abstractions of counter systems. In *LATA*, pages 155–168, 2012.

[BP12] Laura Bozzelli and Sophie Pinchinat. Verification of gap-order constraint abstractions of counter systems. In *VMCAI*, pages 88–103, 2012.

[Cer94] Karlis Cerans. Deciding properties of integral relational automata. In *ICALP*, pages 35–46, 1994.

[Gla01] 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.

[JKM98] Petr Jančar, Antonín Kučera, and Richard Mayr. Deciding bisimulation-like equivalences with finite-state processes. In *ICALP*, pages 200–211, 1998.

[KJ06] Antonín Kučera and Petr Jančar. Equivalence-checking on infinite-state systems: Techniques and results. *TPLP*, 6(3):227–264, 2006.

[May84] Ernst W. Mayr. An algorithm for the general petri net reachability problem. *SIAM J. Comput.*, 13(3):441–460, 1984.

[Min67] Marvin L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.