Logical Methods in Computer Science
Vol. 9(2:01)2013, pp. 1–39
www.lmcs-online.org

Submitted Sep. 28, 2012
Published Apr. 2, 2013

DECIDABLE CLASSES OF TREE AUTOMATA MIXING LOCAL AND GLOBAL CONSTRAINTS MODULO FLAT THEORIES [*]

LUIS BARGUÑÓ [a], CARLES CREUS [b], GUILLEM GODOY [c], FLORENT JACQUEMARD [d], AND CAMILLE VACHER [e]

[a,b,c] Universitat Politècnica de Catalunya, Jordi Girona 1, Barcelona, Spain
e-mail address: {luisbargu,ccreuslopez}@gmail.com, ggodoy@lsi.upc.edu

[d] INRIA Saclay, LSV-CNRS/ENS Cachan
e-mail address: florent.jacquemard@lsv.ens-cachan.fr

[e] LIFL, Univ. Lille I, INRIA Lille, 40 avenue Halley, 59650 Villeneuve d'Ascq, France
e-mail address: vacher@lsv.ens-cachan.fr

Abstract. We define a class of ranked tree automata TABG generalizing both the tree automata with local tests between brothers of Bogaert and Tison (1992) and with global equality and disequality constraints (TAGED) of Filiot et al. (2007). TABG can test for equality and disequality modulo a given flat equational theory between brother subterms and between subterms whose positions are defined by the states reached during a computation. In particular, TABG can check that all the subterms reaching a given state are distinct. This constraint is related to monadic key constraints for XML documents, meaning that every two distinct positions of a given type have different values.

We prove decidability of the emptiness problem for TABG. This solves, in particular, the open question of the decidability of emptiness for TAGED. We further extend our result by allowing global arithmetic constraints for counting the number of occurrences of some state or the number of different equivalence classes of subterms (modulo a given flat equational theory) reaching some state during a computation. We also adapt the model to unranked ordered terms. As a consequence of our results for TABG, we prove the decidability of a fragment of the monadic second order logic on trees extended with predicates for equality and disequality between subtrees, and cardinality.

2012 ACM CCS: [**Theory of computation**]: Formal languages and automata theory—Tree languages; Logic—Higher order logic.

Key words and phrases: Logic, symbolic constraints, tree automata, XML processing.

[*] A preliminary version [BCG$^+$10] appeared in the Proceedings of the 25th Annual IEEE Symposium on Logic In Computer Science (LICS 2010). Here, we generalize the results by allowing the constraints to be interpreted modulo a flat equational theory, and make the results stronger and easier to follow by presenting completely new proofs for the part on arithmetic constraints.

[a,b,c] The first three authors were supported by the FORMALISM project (TIN2007-66523), by the SweetLogics project (TIN2010-21062-C02-01), and by an FI-DGR grant.

[d,e] The last two authors were supported by the Future and Emerging Technologies (FET) program under the FET-Open grant agreement FOX number FP7-ICT-23359, and by the INRIA ARC 2010 project ACCESS.

1. Introduction

Tree automata techniques are widely used in several domains like automated deduction (see e.g. [CDG+07]), static analysis of programs [BT05] or protocols [VGL07, FGVTT04], and XML processing [Sch07]. However, a severe limitation of standard tree automata (TA) is that they are not able to test for equality (isomorphism) or disequality between subterms in an input term. For instance, the language of terms matching a non-linear pattern such as $f(x, x)$ is not regular (i.e. there exists no TA recognizing this language). Let us illustrate how this limitation can be problematic in the context of XML documents processing. XML documents are commonly represented as labeled trees, and they can be constrained by XML schemas, which define both typing restrictions and integrity constraints. All the typing formalisms currently used for XML are based on finite tree automata. The key constraints for databases are common integrity constraints expressing that every two distinct positions of a given type have different values. This is typically the kind of constraints that can not be characterized by TA.

One first approach to overcome this limitation of TA consists in adding the possibility to make equality or disequality tests at each step of the computation of the automaton. The tests are performed *locally*, between subterms at a bounded distance from the current computation position in the input term. The emptiness problem, i.e. whether the language recognized by a given automaton is empty, is undecidable with such tests [Mon81]. A decidable subclass is obtained by restricting the tests to sibling subterms [BT92] (see [CDG+07] for a survey).

Another approach was proposed more recently in [FTT07, FTT08] with the definition of tree automata with *global* equality and disequality tests (TAGED). The TAGED do not perform the tests during the computation steps but globally on the term, at the end of the computation, at positions which are defined by the states reached during the computation. For instance, they can express that all the subterms that reached a given state q are equal, or that every two subterms that reached respectively the states q and q' are different. Nevertheless, arbitrary disequalities are not allowed in TAGED, since such q and q' must be different. The emptiness has been shown decidable for several subclasses of TAGED [FTT07, FTT08], but the decidability of emptiness for the whole class remained a challenging open question.

In this paper, we answer this question positively, for a class of tree recognizers more general than TAGED. We propose (in Section 3) a class of tree automata with local constraints between siblings and global constraints (TABG) which significantly extends TAGED in several directions: (*i*) TABG combine global constraints a la TAGED with *local* equality and disequality constraints between brother subterms a la [BT92], (*ii*) the equality and disequality constraints are treated modulo a given flat equational theory (here flat means that both sides of the equation have the same variables and height, and that this height is bounded by 1), allowing to consider relations more general than syntactic equalities and disequalities, like e.g. structural equalities and disequalities, (*iii*) testing global disequality constraints between subterms that reached the same state is allowed (such test specify key constraints, which are not expressible with TAGED), (*iv*) the global constraints are arbitrary Boolean combinations (including negation) of atomic equality and disequality (in TAGED, only conjunction of atoms are allowed, without negation).

In Section 4, we consider the addition to TABG of global counting constraints on the number $|q|$ of occurrences of a given state q in a computation, or the number $\|q\|$ of distinct equivalence classes (modulo the flat theory) of subterms reaching a given state q in a computation. These counting constraints are only allowed to compare states to constants, like in $|q| \leq 5$ or $\|q\| + 2\|q'\| \geq 9$ (with counting constraints being able to compare state cardinalities, like in $|q| = |q'|$, the emptiness problem becomes undecidable). Using this formalism as an intermediate step, we show that negative literals and disjunctions can be eliminated without loss of generality in the global constraints of TABG, i.e. that TABG whose global constraints are restricted to be conjunctions of positive literals (namely positive conjunctive TABG) have already the same expressiveness of the full TABG class. In particular, the counting constraints do not improve the expressiveness of TABG.

Our main result, presented in Section 5, is that emptiness is decidable for positive conjunctive TABG (and hence for TABG). The decision algorithm uses an involved pumping argument: every sufficiently large term recognized by the given TABG can be reduced by an operation of parallel pumping into a smaller term which is still recognized. The existence of the bound for the minimum accepted term is based on a particular well quasi-ordering.

We show that the emptiness decision algorithm of Section 5 can also be applied to a generalization of the subclass TAG of TABG without the local constraints computing on unranked ordered labeled trees (Section 6). This demonstrates the robustness of the method.

As an application of our results, in Section 7 we present a (strict) extension of the monadic second order logic on trees whose existential fragment corresponds exactly to TAG. In particular, we conclude its decidability.

Related Work. TABG is a strict (decidable) extension of TAG and TA with local equality and disequality constraints, since the expressiveness of both subclasses is incomparable (see e.g. [JKV09]).

The tree automata model of [BT92] has been generalized from ranked trees to unranked ordered trees into a decidable class called UTASC [WL07, LW09]. In unranked trees, the number of brothers (under a position) is unbounded, and UTASC transitions use MSO formulae (on words) with 2 free variables in order to select the sibling positions to be tested for equality and disequality. The decidable generalization of TAG to unranked ordered trees proposed in Section 6 and the automata of [WL07, LW09] are incomparable. The combination of both formalisms could be the object of a further study.

Another way to handle subterm equalities is to use automata computing on DAG representation of terms [Cha99, ANR05]. This model is incomparable to TAG whose constraints are conjunctions of equalities [JKV09]. The decidable extension of TA with one tree shaped memory [CC05] can simulate TAG with equality constraints only, providing that at most one state per run can be used to test equalities [FTT07].

We show in Section 3 that the TABG strictly generalize the TAGED of [FTT07, FTT08]. The latter have been introduced as a tool to decide a fragment of the spatial logic TQL [FTT07]. Decidable subclasses of TAGED were also shown decidable in correspondence with fragments of monadic second order logic on the tree extended with predicates for subtree (dis)equality tests. In Section 7, we generalize this correspondence to TAG and a more natural extension of MSO.

There have been several approaches to extend TA with arithmetic constraints on cardinalities $|q|$ described above: the constraints can be added to transitions in order to count between siblings [SSM03, DL06] (in this case we could call them *local* by analogy with equality tests) or they can be *global* [KR02]. We compare in Section 4 the latter approach (closer to our settings) with our extension of TABG, with respect to emptiness decision. To our knowledge, this is the first time that arithmetic constraints on cardinalities of the form $\|q\|$ are studied.

2. Preliminaries

2.1. Terms, Positions, Replacements.

2.1. Terms, Positions, Replacements. We use the standard notations for terms and positions, see [BN98]. A *signature* Σ is a finite set of function symbols with arity. We sometimes denote Σ explicitly as $\{f_1 : a_1, \dots, f_n : a_n\}$ where $f_1, \dots, f_n$ are the function symbols, and $a_1, \dots, a_n$ are the corresponding arities, or as $\{f_1, \dots, f_n\}$ when the arities are omitted. We denote the subset of function symbols of Σ of arity m as Σ_m. The set of (ranked) *terms* over the signature Σ is defined recursively as $\mathcal{T}(\Sigma) := \{f(t_1, \dots, t_m) \mid f : m \in \Sigma, t_1, \dots, t_m \in \mathcal{T}(\Sigma)\}$. Note that the base case of this definition is $\{f \mid f : 0 \in \Sigma\}$, which coincides with Σ_0 by omitting the arity. Elements of this subset are called constants.

Positions in terms are denoted by sequences of natural numbers. With λ we denote the empty sequence (root position), and $p.p'$ denotes the concatenation of positions p and p'. The set of positions of a term is defined recursively as $Pos\big(f(t_1, \dots, t_m)\big) = \{\lambda\} \cup \{i.p \mid i \in \{1, \dots, m\} \wedge p \in Pos(t_i)\}$. A term $t \in \mathcal{T}(\Sigma)$ can be seen as a function from its set of positions $Pos(t)$ into Σ. For this reason, the symbol labeling the position p in t shall be denoted by $t(p)$. By $p < p'$ and $p \leq p'$ we denote that p is a proper prefix of p', and that p is a prefix of p', respectively. In these cases, p' is necessarily of the form $p.p''$, and we define $p' - p$ as p''. Two positions p_1, p_2 incomparable with respect to the prefix ordering are called *parallel*, and it is denoted by $p_1 \parallel p_2$. The *subterm* of t at position p, denoted $t|_p$, is defined recursively as $t|_\lambda = t$ and $f(t_1, \dots, t_m)|_{i.p} = t_i|_p$. The replacement in t of the subterm at position p by s, denoted $t[s]_p$, is defined recursively as $t[s]_\lambda = s$ and $f(t_1, \dots, t_{i-1}, t_i, t_{i+1}, \dots, t_m)[s]_{i.p} = f(t_1, \dots, t_{i-1}, t_i[s]_p, t_{i+1}, \dots, t_m)$. The *height* of a term t, denoted $h(t)$, is the maximal length of a position of $Pos(t)$. In particular, the length of λ is 0.

2.2. Tree automata. A *tree automaton* (TA, see e.g. [CDG+07]) is a tuple $\mathcal{A} = \langle Q, \Sigma, F, \Delta \rangle$ where Q is a finite set of *states*, Σ is a signature, $F \subseteq Q$ is a subset of final (or accepting) states and Δ is a set of *transition* rules of the form $f(q_1, \dots, q_m) \to q$ where $f : m \in \Sigma$, $q_1, \dots, q_m, q \in Q$. Sometimes, we shall refer to $\mathcal{A}$ as a subscript of its components, like in $Q_\mathcal{A}$ to indicate that this is the set of states of $\mathcal{A}$.

A *run* of $\mathcal{A}$ is a pair $r = \langle t, M \rangle$ where t is a term in $\mathcal{T}(\Sigma)$ and $M : Pos(t) \to \Delta_\mathcal{A}$ is a mapping satisfying the following statement for each $p \in Pos(t)$: if $t|_p$ is written of the form $f(t_1, \dots, t_m)$, and $M(p.1), \dots, M(p.m)$ are rules with right-hand side states $q_1, \dots, q_m \in Q_\mathcal{A}$, respectively, then $M(p)$ is a rule of the form $f(q_1, \dots, q_m) \to q$ for some $q \in Q_\mathcal{A}$. We write $r(p)$ for the right-hand side state of $M(p)$, and say that r is a run of $\mathcal{A}$ on t. Moreover, by $\mathtt{term}(r)$ we refer to t, and by $\mathtt{symbol}(r)$ we refer to $t(\lambda)$. The run r is called *successful* (or *accepting*) if $r(\lambda)$ is in $F_\mathcal{A}$. The language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of terms t for which there exists a successful run of $\mathcal{A}$. A language L is called *regular* if there exists a TA $\mathcal{A}$ satisfying $L = \mathcal{L}(\mathcal{A})$. For facility of explanations, we shall use term-like notations for runs defined as follows in the natural way. For a run $r = \langle t, M \rangle$, by $Pos(r)$ we denote $Pos(t)$, and by $h(r)$ we denote $h(t)$. Similarly, by $r|_p$ we denote the run $\langle t|_p, M|_p \rangle$, where $M|_p$ is defined as

$M|_p(p') = M(p.p')$ for each p' in $Pos(t|_p)$, and say that $r|_p$ is a subrun of r. Moreover, for a run $r' = \langle t', M'\rangle$ such that the states $r'(\lambda)$ and $r(p)$ coincide, by $r[r']_p$ we denote the run $\langle t[t']_p, M[M']_p\rangle$, where $M[M']_p$ is defined as $M[M']_p(p.p') = M'(p')$ for each p' in $Pos(t')$, and as $M[M']_p(p') = M(p')$ for each p' with $p \not\leq p'$.

2.3. **Tree automata with local constraints between brothers.** A *tree automaton with constraints between brothers* (defined in [BT92] and called `TACBB` in [CDG$^+$07]) is a tuple $\mathcal{A} = \langle Q, \Sigma, F, \Delta\rangle$ where Q, Σ and F are defined as for `TA`, but with the difference that Δ is a set of constrained rules of the form $f(q_1, \ldots, q_m) \xrightarrow{C} q$, where C is a set of equalities and disequalities of the form $i \approx j$ or $i \not\approx j$ for $i, j \in \{1, \ldots, m\}$. We call C a local *constraint between brothers*. By $ta(\mathcal{A})$ we define the `TA` obtained from $\mathcal{A}$ by removing all constraints from Δ.

A *run* of a `TACBB` $\mathcal{A}$ is a pair $r = \langle t, M\rangle$ defined similarly to the case of `TA`; t is a term in $\mathcal{T}(\Sigma)$ and the mapping $M : Pos(t) \to \Delta_{\mathcal{A}}$ satisfies the following statement for each $p \in Pos(t)$: if $t|_p$ is written of the form $f(t_1, \ldots, t_m)$, and $M(p.1), \ldots, M(p.m)$ are rules with right-hand side states $q_1, \ldots, q_m \in Q_{\mathcal{A}}$, respectively, then $M(p)$ is a rule of the form $f(q_1, \ldots, q_m) \xrightarrow{C} q$ for some $q \in Q_{\mathcal{A}}$ and constraint between brothers C. Moreover, for each equality $i \approx j$ in C, $t_i = t_j$ holds, and for each disequality $i \not\approx j$ in C, $t_i \neq t_j$ holds. The notions of successful run and recognized language are defined for `TACBB` analogously to the case of `TA`.

2.4. **Term equations.** Given a set of variables $\mathcal{X}$, the set of (ranked) *terms* over Σ and $\mathcal{X}$ is defined as $\mathcal{T}(\Sigma \cup \mathcal{X})$ by considering arity 0 for the elements of $\mathcal{X}$. A substitution σ is a mapping from variables to terms $\sigma : \mathcal{X} \to \mathcal{T}(\Sigma \cup \mathcal{X})$. It is also considered as a function from arbitrary terms to terms $\sigma : \mathcal{T}(\Sigma \cup \mathcal{X}) \to \mathcal{T}(\Sigma \cup \mathcal{X})$ by the recursive definition $\sigma(f(t_1, \ldots, t_m)) = f(\sigma(t_1), \ldots, \sigma(t_m))$ for every function symbol f and subterms $t_1, \ldots, t_m$.

An equation between terms is an unordered pair of terms denoted $l \approx r$. Given a set of equations E and two terms s, t, we say that s and t are equivalent modulo E, denoted $s =_E t$, if there exist terms $s_1, s_2, \ldots, s_n, n \geq 1$ satisfying the following statement: $s = s_1$, $s_n = t$, and for each $i \in \{1, \ldots, n-1\}$, there exists an equation $l \approx r$ in E, a substitution σ, and a position p, such that $s_i|_p = \sigma(l)$ and $s_{i+1} = s_i[\sigma(r)]_p$. A *flat equation* is an equation $l \approx r$ where l and r are terms satisfying $h(l) = h(r) \leq 1$, and any variable x occurs in l if and only if x occurs in r. A *flat theory* is a set of flat equations.

The following technical lemma shows that equivalence modulo a flat theory is preserved by certain replacements of subterms. It will be useful in Section 5.

Lemma 2.1. *Let E be a flat theory. Let $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_m)$, $s' = f(s'_1, \ldots, s'_n)$ and $t' = g(t'_1, \ldots, t'_m)$ be terms satisfying the following conditions:*

- *For each $i \in \{1, \ldots, n\}$, $(s_i \in \Sigma_0 \Leftrightarrow s'_i \in \Sigma_0)$ and $(s_i, s'_i \in \Sigma_0 \Rightarrow s_i =_E s'_i)$ hold.*
- *For each $j \in \{1, \ldots, m\}$, $(t_j \in \Sigma_0 \Leftrightarrow t'_j \in \Sigma_0)$ and $(t_j, t'_j \in \Sigma_0 \Rightarrow t_j =_E t'_j)$ hold.*
- *For each $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$, $(s'_i =_E t'_j \Leftrightarrow s_i =_E t_j)$ holds.*

Then, $s =_E t \Leftrightarrow s' =_E t'$ holds.

Proof. We prove the left-to-right direction only. The other one is analogous by swapping the roles of s and t by the roles of s' and t', respectively.

Since $s =_E t$ holds, there exist terms $u_1, u_2, \dots, u_k, k \geq 1$ satisfying the following statement: $s = u_1$, $u_k = t$, and for each $i \in \{1, \dots, k-1\}$, there exists an equation $l \approx r$ in E, a substitution σ, and a position p, such that $u_i|_p = \sigma(l)$ and $u_{i+1} = u_i[\sigma(r)]_p$.

We prove the statement by induction on k. For $k = 1$, $s = t$ holds. Thus, g is f, m is n, and for each $i \in \{1, \dots, n\}$, $s_i = t_i$ holds. In particular, each $s_i =_E t_i$ holds. Therefore, each $s'_i =_E t'_i$ also holds, and hence $s' = f(s'_1, \dots, s'_n) =_E f(t'_1, \dots, t'_n) = t'$ holds.

Now, assume $k > 1$. Let $l \approx r$, p and σ be the rule, position and substitution satisfying $u_1|_p = \sigma(l)$ and $u_2 = u_1[\sigma(r)]_p$. Recall that u_1 is s. First, suppose that p is not λ. Then, p is of the form $j.p'$ for some $j \in \{1, \dots, n\}$ and position p'. Note that $u_2|_j =_E u_1|_j$ holds, and for each $i \in \{1, \dots, n\} \setminus \{j\}$, $u_2|_i = u_1|_i$ holds. Thus, u_2 is of the form $f(v_1, \dots, v_n)$ and for each $i \in \{1, \dots, n\}$, $v_i =_E s_i$ holds. Moreover, since E is a flat theory, the step at p preserves the height, and hence, for each $i \in \{1, \dots, n\}$, $v_i \in \Sigma_0 \Leftrightarrow s_i \in \Sigma_0$ and $v_i, s_i \in \Sigma_0 \Rightarrow v_i =_E s_i$ hold. From the statement of the lemma, the following conditions follow:

- For each $i \in \{1, \dots, n\}$, $(v_i \in \Sigma_0 \Leftrightarrow s'_i \in \Sigma_0)$ and $(v_i, s'_i \in \Sigma_0 \Rightarrow v_i =_E s'_i)$ hold.
- For each $j \in \{1, \dots, m\}$, $(t_j \in \Sigma_0 \Leftrightarrow t'_j \in \Sigma_0)$ and $(t_j, t'_j \in \Sigma_0 \Rightarrow t_j =_E t'_j)$ hold.
- For each $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$, $(s'_i =_E t'_j \Leftrightarrow v_i =_E t_j)$ holds.

By induction hypothesis, $f(s'_1, \dots, s'_n) =_E g(t'_1, \dots, t'_m)$ holds, and we are done.

Now, consider the case where p is λ. In this case $s = u_1 = \sigma(l)$, and $u_2 = \sigma(r)$. Since E is a flat theory, l and r are of the form $f(\alpha_1, \dots, \alpha_n)$ and $h(\beta_1, \dots, \beta_\mu)$, where either $n, \mu > 0$ or $n = \mu = 0$, and $\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_\mu$ are either constants or variables. Moreover, a variable occurs in l if and only if it occurs in r. Note that $\sigma(\alpha_1) = s_1, \dots, \sigma(\alpha_n) = s_n$ holds. We call $v_1 = \sigma(\beta_1), \dots, v_\mu = \sigma(\beta_\mu)$. Note that $u_2 = h(v_1, \dots, v_\mu)$. We define terms $v'_1, \dots, v'_\mu$ as follows for each i in $\{1, \dots, \mu\}$. If v_i is a constant, then we define v'_i as v_i. Otherwise, if v_i is not a constant, then β_i is a variable x. Since E is a flat theory, some α_j (we choose any) must be x. In this case we define v'_i as s'_j. With these definitions, the following conditions follow:

- For each $i \in \{1, \dots, \mu\}$, $(v_i \in \Sigma_0 \Leftrightarrow v'_i \in \Sigma_0)$ and $(v_i, v'_i \in \Sigma_0 \Rightarrow v_i =_E v'_i)$ hold.
- For each $j \in \{1, \dots, m\}$, $(t_j \in \Sigma_0 \Leftrightarrow t'_j \in \Sigma_0)$ and $(t_j, t'_j \in \Sigma_0 \Rightarrow t_j =_E t'_j)$ hold.
- For each $i \in \{1, \dots, \mu\}$ and $j \in \{1, \dots, m\}$, $(v'_i =_E t'_j \Leftrightarrow v_i =_E t_j)$ holds.

By induction hypothesis, $h(v'_1, \dots, v'_\mu) =_E g(t'_1, \dots, t'_m)$ holds.

Now, let $s''_1, \dots, s''_n$ be defined as follows for each i in $\{1, \dots, n\}$. If s'_i is not a constant then define s''_i as s'_i. Otherwise, if s'_i is a constant, then define s''_i as s_i. By the condition $(s_i, s'_i \in \Sigma_0 \Rightarrow s_i =_E s'_i)$ we have that $f(s'_1, \dots, s'_n) =_E f(s''_1, \dots, s''_n)$ holds. Moreover, the same rule $l \approx r$ can be used to prove $f(s''_1, \dots, s''_n) =_E h(v'_1, \dots, v'_\mu)$. Hence, $f(s'_1, \dots, s'_n) =_E f(s''_1, \dots, s''_n) =_E h(v'_1, \dots, v'_\mu) =_E g(t'_1, \dots, t'_m)$ holds, and we are done. $\square$

2.5. **Well quasi-orderings.** A *well quasi-ordering* [Gal91] $\leq$ on a set S is a reflexive and transitive relation such that any infinite sequence of elements $e_1, e_2, \dots$ of S contains an increasing pair $e_i \leq e_j$ with $i < j$.

3. Tree Automata with Global Constraints

In this subsection, we define a class of tree automata with global constraints strictly generalizing both the `TACBB` of [BT92] and the `TAGED` of [FTT08]. The generalization consists in considering more general global constraints, and interpreting all the constraints modulo a flat equational theory.

As an intermediate step, we define an extension of the `TACBB` of [BT92] where the local constraints between brothers are considered modulo a flat equational theory.

Definition 3.1. A *tree automaton with constraints between brothers modulo a flat theory* (`TAB`) is a tuple $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E\rangle$ where $\langle Q, \Sigma, F, \Delta\rangle$ is a `TACBB` and E is a flat equational theory.

By $ta(\mathcal{A})$ we denote $ta(\langle Q, \Sigma, F, \Delta\rangle)$.

A *run* of a `TAB` $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E\rangle$ is a pair $r = \langle t, M\rangle$ defined analogously to a run of a `TACBB`, except that the constraints between brothers are interpreted modulo E. More specifically, for each position p in $Pos(t)$, if $t|_p$ is written of the form $f(t_1, \ldots, t_m)$, and $M(p.1), \ldots, M(p.m)$ are rules with right-hand side states $q_1, \ldots, q_m \in Q$, respectively, then $M(p)$ is a transition rule of $\Delta_{\mathcal{A}}$ of the form $f(q_1, \ldots, q_m) \xrightarrow{C} q$ for some $q \in Q$ and constraint between brothers C. Moreover, for each equality $i \approx j$ in C, $t_i =_E t_j$ holds, and for each disequality $i \not\approx j$ in C, $t_i \neq_E t_j$ holds. The notions of successful run and recognized language are defined for `TAB` analogously to the case of `TA`.

We further extend this class `TAB` with global equality and disequality constraints generalizing those of `TAGED` [FTT08].

Definition 3.2. A *tree automaton with global and brother constraints modulo a flat theory* (`TABG`) is a tuple $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ where $\langle Q, \Sigma, F, \Delta, E\rangle$ is a `TAB`, denoted $tab(\mathcal{A})$, and C is a Boolean combination of atomic constraints of the form $q \approx q'$ or $q \not\approx q'$, where $q, q' \in Q$.

By $ta(\mathcal{A})$ we denote $ta(tab(\mathcal{A}))$.

A *run* of a `TABG` $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ is a run $r = \langle t, M\rangle$ of $tab(\mathcal{A})$ such that r satisfies C, denoted $r \models C$, where the satisfiability of constraints is defined as follows. For atomic constraints, $r \models q \approx q'$ (respectively $r \models q \not\approx q'$) holds if and only if for all different positions $p, p' \in Pos(t)$ such that $M(p) = q$ and $M(p') = q'$, $t|_p =_E t|_{p'}$ (respectively $t|_p \neq_E t|_{p'}$) holds. This notion of satisfiability is extended to Boolean combinations as usual. As for `TA`, we say that r is a run of $\mathcal{A}$ on t. A run r of $\mathcal{A}$ on $t \in \mathcal{T}(\Sigma)$ is *successful* (or *accepting*) if $r(\lambda) \in F$. The language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of terms t for which there exists a successful run of $\mathcal{A}$.

It is important to note that the semantics of $\neg(q \approx q')$ and $q \not\approx q'$ differ, as well as the semantics of $\neg(q \not\approx q')$ and $q \approx q'$. This is because we have a "for all" quantifier in both definitions of semantics of $q \approx q'$ and $q \not\approx q'$.

Let us introduce some notations, summarized in Figure 1 that we use below to characterize some classes of tree automata related to `TABG` (Figure 1 also refers to a class defined in Section 4). A `TABG` $\mathcal{A}$ is called *positive* if $C_{\mathcal{A}}$ is a disjunction of conjunctions of atomic constraints and it is called *positive conjunctive* if $C_{\mathcal{A}}$ is a conjunction of atomic constraints. The subclass of positive conjunctive `TABG` is denoted by `TABG`$^{\wedge}$.

We recall that a `TAB` where all the constraints are empty is just a `TA`. For a `TABG` $\mathcal{A}$, when the theory $E_{\mathcal{A}}$ is empty and $tab(\mathcal{A})$ is just a `TA`, we say that $\mathcal{A}$ is just a tree automaton with global constraints (`TAG`). Its subclass with positive conjunctive constraints is denoted `TAG`$^{\wedge}$.

With the notation `TABG`$[\tau_1, \ldots, \tau_m]$, we characterize the class of tree automata with global and brother constraints modulo a flat theory whose global constraints are Boolean

$$\begin{tikzcd}
 & \mathtt{TABG}[\approx,\not\approx,\mathbb{N}] \arrow[rr, equal] & & \mathtt{TABG}[\approx,\not\approx] \arrow[r, equal] & \text{positive } \mathtt{TABG}[\approx,\not\approx] \arrow[r, equal] & \mathtt{TABG}^{\wedge}[\approx,\not\approx] & \\
 & & & \mathtt{TAG}[\approx,\not\approx] \arrow[u] & & & \\
\mathtt{TAG}[\approx] \arrow[r] \arrow[urrr] & \mathtt{TABG}[\approx] \arrow[uurr] & & & & \mathtt{TABG}[\not\approx] \arrow[uull] & \mathtt{TAG}[\not\approx] \arrow[l] \arrow[ulll] \\
 & & & \mathtt{TAB} \arrow[ull] \arrow[urr] & & & \\
 & & & \mathtt{TACBB} \arrow[u] & & & \\
 & & & \mathtt{TA} \arrow[u] \arrow[uuulll] \arrow[uuurrr] & & &
\end{tikzcd}$$

$\rightarrow$: effective strict inclusion $=$: effective equivalence

Figure 1: Decidable classes of TA with local and global constraints

combination of atomic constraints of types $\tau_1, \dots, \tau_m$. The types $\approx$ and $\not\approx$ denote respectively the atomic constraints of the form $q \approx q'$ and $q \not\approx q'$, where q, q' are states. For instance, the abbreviation TABG used in Definition 3.2 stands for $\mathtt{TABG}[\approx, \not\approx]$. This notation is extended to the positive conjunctive fragment by $\mathtt{TABG}^{\wedge}[\tau_1, \dots, \tau_k]$ and to the fragment without local constraints between brother, by $\mathtt{TAG}[\tau_1, \dots, \tau_k]$.

3.1. **Expressiveness.**

The class of regular languages is strictly included in the class of TABG languages due to the constraints.

Example 3.3. Let $\Sigma = \{a : 0, f : 2\}$. The set $\{f(t, t) \mid t \in \mathcal{T}(\Sigma)\}$ is not a regular tree language (this can be shown using a classical *pumping* argument).

However, it is recognized by the following TAB:

$$\big\langle \{q_0, q_\mathsf{f}\}, \Sigma, \{q_\mathsf{f}\}, \{a \to q_0, f(q_0, q_0) \to q_0, f(q_0, q_0) \xrightarrow{1\approx 2} q_\mathsf{f}\}, \emptyset \big\rangle,$$

and it is also recognized by the following $\mathtt{TAG}[\approx]$:

$$\mathcal{A} = \big\langle \{q_0, q_1, q_\mathsf{f}\}, \Sigma, \{q_\mathsf{f}\}, \{a \to q_0 \mid q_1, f(q_0, q_0) \to q_0 \mid q_1, f(q_1, q_1) \to q_\mathsf{f}\}, \emptyset, q_1 \approx q_1 \big\rangle,$$

where $t \to q \mid q_r$ is an abbreviation for $t \to q$ and $t \to q_r$. An example of successful run of $\mathcal{A}$ on $t = f(f(a, a), f(a, a))$ is $q_\mathsf{f}\big(q_1(q_0, q_0), q_1(q_0, q_0)\big)$, where we use term-like notation for marking the reached state at each position.

Moreover, the TAGED of [FTT08] are also a particular case of $\mathtt{TAG}[\approx, \not\approx]$, since they can be redefined in our setting as restricted $\mathtt{TAG}^{\wedge}[\approx, \not\approx]$, where the equational theory is empty, and where q and q' are required to be distinct in any atomic constraint of the form $q \not\approx q'$.

Reflexive disequality constraints such as $q \not\approx q$ correspond to monadic *key constraints* for XML documents, meaning that every two distinct positions of type q have different values. A state q of a $\mathtt{TAG}[\approx, \not\approx]$ can be used for instance to characterize unique identifiers as in the following example, which presents a $\mathtt{TAG}[\approx, \not\approx]$ whose language cannot be recognized by a TAGED. This example will be referred several times in Section 5, in order to illustrate the definitions used in the decision procedure of the emptiness problem for $\mathtt{TAG}[\approx, \not\approx]$.

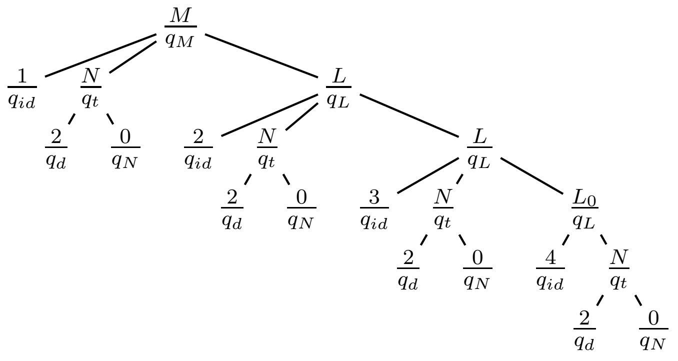

Figure 2: Term and successful run (Example 3.4).

Example 3.4. The TAG$[\approx, \not\approx]$ of our running example accepts (in state q_M) lists of dishes called menus, where every dish is associated with one identifier (state q_{id}) and the time needed to cook it (state q_t). We have other states accepting digits (q_d), numbers (q_N) and lists of dishes (q_L).

The TAG$[\approx, \not\approx]$ $\mathcal{A} = \langle Q, \Sigma, F, \Delta, \emptyset, C\rangle$ is defined as follows: $\Sigma = \{0, \dots, 9 : 0, N, L_0 : 2, L, M : 3\}$, $Q = \{q_d, q_N, q_{id}, q_t, q_L, q_M\}$, $F = \{q_M\}$, and $\Delta = \{i \to q_d \,|\, q_N \,|\, q_{id} \,|\, q_t : 0 \le i \le 9\}$ $\cup$ $\{N(q_d, q_N) \to q_N \mid q_{id} \mid q_t, L_0(q_{id}, q_t) \to q_L, L(q_{id}, q_t, q_L) \to q_L, M(q_{id}, q_t, q_L) \to q_M\}$.

The constraint C ensures that all the identifiers of the dishes in a menu are pairwise distinct (i.e. that q_{id} is a key) and that the time to cook is the same for all dishes: $C = q_{id} \not\approx q_{id} \wedge q_t \approx q_t$. A term in $\mathcal{L}(\mathcal{A})$ together with an associated successful run are depicted in Figure 2.

Althought this is a simple exercise, let us establish formally that TAG$[\approx, \not\approx]$ are strictly more expressive than TAGED.

Lemma 3.5. *The class of languages recognized by TAG$^\wedge[\approx, \not\approx]$ strictly includes the class of languages recognized by TAGED.*

Proof. Since a TAGED is just a TAG$^\wedge[\approx, \not\approx]$ where no constraint of the form $q \not\approx q$ occurs, the inclusion holds. In order to see that it is strict, it suffices to show a language L which can be recognized by a TAG$^\wedge[\approx, \not\approx]$ but not by a TAGED.

Let $\Sigma = \{a : 0, s : 1, f : 2\}$. The set L of terms of $\mathcal{T}(\Sigma)$ of the form $f(s^{n_1}(a), f(s^{n_2}(a), \dots, f(s^{n_k}(a), a)\dots))$, such that $k \ge 0$ and the natural numbers n_i, for $i \le k$, are pairwise distinct, is recognized by the following TAG$^\wedge[\approx, \not\approx]$:

$$\left\langle \{q_a, q, q_\mathsf{f}\}, \Sigma, \{q_\mathsf{f}\}, \left\{ \begin{array}{c} a \to q_a \mid q \mid q_\mathsf{f}, \\ s(q_a) \to q_a \mid q, \\ f(q, q_\mathsf{f}) \to q_\mathsf{f} \end{array} \right\}, \emptyset, q \not\approx q \right\rangle.$$

Assume that there exists a TAG$^\wedge[\approx, \not\approx]$ $\mathcal{A}$ without reflexive disequality constraints of the form $q \not\approx q$ (i.e. a TAGED), recognizing this language L. Then, there exists an accepting run r of $\mathcal{A}$ on the term $t = f(s(a), f(s^2(a), \dots f(s^{|Q_\mathcal{A}|+1}(a), a)\dots)) \in L$. Therefore, $r \models C_\mathcal{A}$ (the global constraint of $\mathcal{A}$, which is positive by hypothesis).

There are two different positions $p_i = \overbrace{2.2.\dots.2}^{i}.1$ and $p_j = \overbrace{2.2.\dots.2}^{j}.1$, $0 \le i < j \le |Q_\mathcal{A}|$ such that $r(p_i) = r(p_j)$. Let us show that $r' = r[r|_{p_i}]_{p_j}$ is an accepting run of $\mathcal{A}$ on

$t' = t[t|_{p_i}]_{p_j}$. Since $r(p_i) = r(p_j)$ and r is a run of $\mathcal{A}$ on t, r' is a run of $ta(\mathcal{A})$ on t'. Hence, it suffices to prove that the constraint $C_{\mathcal{A}}$ is satisfied by r'. Consider a position p of the form $2.2.\ldots.2$ with $|p| < j$. We start by proving that any atomic constraint involving $r'(p)$ is satisfied. Note that $r'(p) = r(p)$ holds, and that the subterm $t|_p$ has only this occurrence in t. Thus, any atomic constraint involving $r(p)$ and a state q occurring in r is necessarily of the form $r(p) \not\approx q$. Since any state occurring in r' occurs also in r, any atomic constraint involving $r'(p)$ and a state q occurring in r' is of the form $r'(p) \not\approx q$. Moreover, the subterm $t'|_p$ has only this occurrence in t'. Thus, such a constraint is satisfied. Now consider two different positions p_1, p_2 which are not of the form described above. It remains to see that any atomic constraint involving $r'(p_1)$ and $r'(p_2)$ is satisfied. In the case where $r'|_{p_1}$ and $r'|_{p_2}$ are different, this is a direct consequence of the fact that both subruns $r'|_{p_1}$ and $r'|_{p_2}$ are also subruns of r at different positions. Otherwise, in the case where $r'|_{p_1}$ and $r'|_{p_2}$ are the same subrun, then, $r'(p_1) = r'(p_2)$ holds, and any atomic constraint involving $r'(p_1)$ and $r'(p_2)$ must be of the form $r'(p_1) \approx r'(p_2)$ because $\mathcal{A}$ has no reflexive disequalities. Thus, the atomic constraint is also satisfied in this case. ☐

The following example shows a `TABG` recognizing a language that cannot be recognized by a `TAG`$[\approx, \not\approx]$. The proof is a simple exercise and it is left to the reader.

Example 3.6. Assume that the terms of Example 3.4 are now used to record the activity of a restaurant. To this end, we transform the `TAG` of example 3.4 into a `TABG` as follows. First, in order to simplify the example we omit the restriction that all cooking times coincide, i.e. $C = q_{id} \not\approx q_{id}$. Second, we add a new argument of type q_t to L_0, L and M, so that the old argument q_t characterizes the theoretical time to cook, and the new q_t characterizes the real time that was needed to cook the dish. Let us replace the transitions with L_0, L and M in input by $L_0(q_{id}, q_t, q_t) \xrightarrow{2\approx3} q_L$, $L_0(q_{id}, q_t, q_t) \xrightarrow{2\not\approx3} q'_L$, $L(q_{id}, q_t, q_t, q_L) \xrightarrow{2\approx3} q_L$, $L(q_{id}, q_t, q_t, q_L) \xrightarrow{2\not\approx3} q'_L$, $M(q_{id}, q_t, q_t, q_L) \xrightarrow{2\approx3} q_M$, $M(q_{id}, q_t, q_t, q_L) \xrightarrow{2\not\approx3} q'_M$, where q'_L is a new state meaning that there was an anomaly. We also add a transition $L(q_{id}, q_t, q_t, q'_L) \to q'_L$ to propagate q'_L and $M(q_{id}, q_t, q_t, q'_L) \to q'_M$.

By keeping the set of final states as $\{q_M\}$, the recognized language of the `TABG` obtained is the set of records well cooked, i.e. such that for all dishes, the real time to cook is equal to the theoretical time. By redefining the set of final states as $\{q'_M\}$, the recognized language is the set of records with an anomaly.

3.2. Decision Problems.

The *membership* is the problem to decide, given a term $t \in \mathcal{T}(\Sigma)$ and a `TABG` $\mathcal{A}$ over Σ whether $t \in \mathcal{L}(\mathcal{A})$.

Proposition 3.7. *Membership is NP-complete for `TABG`, by assuming that the maximum arity of the signature Σ is a constant for the problem.*

Proof. In order to prove that this problem is in NP, given a `TABG` $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C \rangle$ and a term $t \in \mathcal{T}(\Sigma)$, we can non-deterministically guess a function M from $Pos(t)$ into Δ, and check that $\langle t, M \rangle$ is a successful run of $\mathcal{A}$ on t. The checking can be performed in polynomial time. In particular, testing equivalence modulo E can be performed in polynomial time using a dynamic programming scheme, by assuming that the maximum arity of Σ is a constant of the problem, which is a usual assumption. More general results are given in [Nie96, CHJ94]. For NP-hardness, [FTT08, JKV09] present PTIME reductions of the satisfiability of Boolean expressions into membership for `TAG`$^\wedge[\approx]$ whose constraints are conjunctions of equalities of the form $q \approx q$. ☐

Recall that for plain TA, membership is in PTIME.

The *universality* is the problem to decide, given a TABG $\mathcal{A}$ over Σ, whether $\mathcal{L}(\mathcal{A}) = \mathcal{T}(\Sigma)$. It is known to be undecidable already for a small subclass of TAG.

Proposition 3.8. [FTT08, JKV09] *Universality is undecidable for* $\mathtt{TAG}^{\wedge}[\approx]$.

The following consequence is a new result for TAGED.

Proposition 3.9. *It is undecidable whether the language of a given* $\mathtt{TAG}^{\wedge}[\approx]$ *is regular.*

Proof. We show that universality is reducible to regularity using a new function symbol f with arity 2, and any non-regular language L which is recognizable by a $\mathtt{TAG}^{\wedge}[\approx]$ (such a language exists).

Let $\mathcal{A}$ be an input of universality for $\mathtt{TAG}^{\wedge}[\approx]$ and let

$$L' = \big\{f(t_1, t_2) \mid t_1 \in \mathcal{T}(\Sigma) \wedge t_2 \in L\big\} \cup \big\{f(t_1, t_2) \mid t_1 \in \mathcal{L}(\mathcal{A}) \wedge t_2 \in \mathcal{T}(\Sigma)\big\}.$$

It is possible to compute a new $\mathtt{TAG}^{\wedge}[\approx]$ $\mathcal{A}'$ recognizing the language L' (see Lemma 4.19). Thus, in order to conclude, it suffices to show that $\mathcal{L}(\mathcal{A}) = \mathcal{T}(\Sigma)$ if and only if $\mathcal{L}(\mathcal{A}')$ is regular. For this purpose let us first define the quotient of a term language R by a term s with respect to a function symbol f: $R/s := \{t \mid f(s,t) \in R\}$. This operation preserves regular languages: for all s and f, if R is regular then R/s is regular.

If $\mathcal{L}(\mathcal{A}) = \mathcal{T}(\Sigma)$, then $\mathcal{L}(\mathcal{A}')$ is $\big\{f(t_1, t_2) \mid t_1, t_2 \in \mathcal{T}(\Sigma)\big\}$, which is regular. Assume that $\mathcal{L}(\mathcal{A}) \neq \mathcal{T}(\Sigma)$ and let $s \in \mathcal{T}(\Sigma) \setminus \mathcal{L}(\mathcal{A})$. By construction, $\mathcal{L}(\mathcal{A}')/s = L$ which is not regular. Hence $\mathcal{L}(\mathcal{A}')$ is not regular. □

The *emptiness* is the problem to decide, given a TABG $\mathcal{A}$, whether $\mathcal{L}(\mathcal{A}) = \emptyset$. The proof that it is decidable for TABG is rather involved and is presented in Section 5.

4. Arithmetic Constraints and Reduction to $\mathtt{TABG}^{\wedge}$

This section has two goals. The first goal is to present an extension of TABG by allowing certain global arithmetic constraints. They are interesting by themselves since they allow the representation of several natural properties in a simple way. The second goal is to show that the class of TABG languages coincides (in expressiveness) with the class of $\mathtt{TABG}^{\wedge}$ languages. In other words, for each TABG there exists a $\mathtt{TABG}^{\wedge}$ recognizing the same language. This reduction will be very useful in Section 5 in order to prove decidability of emptiness of TABG.

The reason for presenting both results in the same section is that arithmetic constraints simplify the task of transforming a TABG into a $\mathtt{TABG}^{\wedge}$ representing the same language. This is because negations can be replaced by arithmetic constraints with an equivalent meaning in a first intermediate step, and such constraints are easier to deal with.

All this work is developed in Subsection 4.2. Before that, in Subsection 4.1 we present a more general form of arithmetic constraints for which emptiness is undecidable. The motivation of this first subsection is to show the limits of positive results in this setting, and to justify the limited form of the constraints in Subsection 4.2.

4.1. **Global Integer Linear Constraints.** Let Q be a set of states. A *linear inequality* over Q is an expression of the form $\sum_{q \in Q} a_q \cdot |q| \geq a$ or $\sum_{q \in Q} a_q \cdot \|q\| \geq a$ where every a_q and a belong to $\mathbb{Z}$. We consider the above linear inequalities as atomic constraints of tree automata with global constraints, and denote by $|.|_{\mathbb{Z}}$ and $\|.\|_{\mathbb{Z}}$ their respective types. The type $\mathbb{Z}$ denotes $|.|_{\mathbb{Z}}$ and $\|.\|_{\mathbb{Z}}$ together.

Using the notation introduced in Section 3, $\mathtt{TABG}[\approx, \not\approx, |.|_{\mathbb{Z}}, \|.\|_{\mathbb{Z}}]$ (or $\mathtt{TABG}[\approx, \not\approx, \mathbb{Z}]$) denotes the class of tree automata with global and brother constraints modulo a flat theory of the form $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C \rangle$ such that $\langle Q, \Sigma, F, \Delta, E \rangle$ is a $\mathtt{TAB}$ (denoted $tab(\mathcal{A})$) and C is a Boolean combination of atomic constraints which can be linear inequalities as above or equality or disequality constraints of the form $q \approx q'$ or $q \not\approx q'$, with $q, q' \in Q$.

Let $\mathcal{A}$ be a $\mathtt{TABG}[\approx, \not\approx, |.|_{\mathbb{Z}}, \|.\|_{\mathbb{Z}}]$ over Σ and with state set Q and flat equational theory E, let r be a run of $tab(\mathcal{A})$ on a term $t \in \mathcal{T}(\Sigma)$ and let $q \in Q$. Intuitively, the interpretation of $|q|$ with respect to r is the number of occurrences of q in r, i.e. the number of positions p holding $r(p) = q$. The interpretation of $\|q\|$ with respect to r is the number of different subterms (modulo E) in t reaching state q with r, i.e. the maximum number of positions $p_1, p_2, \ldots, p_n$ holding $r(p_1) = r(p_2) = \ldots = r(p_m) = q$ and such that the terms $t|_{p_1}, t|_{p_2}, \ldots, t|_{p_n}$ are pairwise different (modulo E). More formally, the interpretations of $|q|$ and $\|q\|$ with respect to r (and t) are defined, respectively, by the following cardinalities:

$$[\![\, |q| \,]\!]_r = \big|\{p \mid p \in Pos(t) \ \wedge \ r(p) = q\}\big|$$

$$[\![\, \|q\| \,]\!]_r = \big|\{[t|_p]_E \mid p \in Pos(t) \ \wedge \ r(p) = q\}\big|.$$

This permits to define the satisfiability of linear inequalities with respect to r and t: $r \models \sum_{q \in Q} a_q \cdot |q| \geq a$ holds if and only if $\sum_{q \in Q} a_q \cdot [\![\, |q| \,]\!]_r \geq a$ holds, and $r \models \sum_{q \in Q} a_q \cdot \|q\| \geq a$ holds if and only if $\sum_{q \in Q} a_q \cdot [\![\, \|q\| \,]\!]_r \geq a$ holds. The satisfiability of the global constraint $C_{\mathcal{A}}$ of $\mathcal{A}$ by r, denoted $r \models C_{\mathcal{A}}$ is defined accordingly, and if $r \models C_{\mathcal{A}}$ then r is called a run of $\mathcal{A}$. A run of $\mathcal{A}$ on $t \in \mathcal{T}(\Sigma)$ is *successful* (or *accepting*) if $r(\lambda) \in F_{\mathcal{A}}$. The language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of terms t for which there exists a successful run of $\mathcal{A}$.

Example 4.1. Let us add a new argument to the dishes of the menu of Example 3.4 which represents the price coded on two digits by a term $N(d_1, d_0)$. We add a new state q_p for the type of prices, and other states q_{cheap}, $q_{moderate}$, $q_{expensive}$, q_{chic} describing price level ranges, and transitions $0|1 \to q_{cheap}$, $2|3 \to q_{moderate}$, $4|5|6 \to q_{expensive}$, $7|8|9 \to q_{chic}$ and $N(q_{cheap}, q_d) \to q_p$, ... The price is a new argument of L_0, L and M, hence we replace the transitions with these symbols in input by $L_0(q_{id}, q_t, q_p) \to q_L$, $L(q_{id}, q_t, q_p, q_L) \to q_L$, $M(q_{id}, q_t, q_p, q_L) \to q_M$. We can use a linear inequality $|q_{cheap}| + |q_{moderate}| - |q_{expensive}| - |q_{chic}| \geq 0$ to characterize the moderate menus, and $|q_{expensive}| + |q_{chic}| \geq 6$ to characterize the menus with too many expensive dishes. A linear inequality $\|q_p\| \leq 1$ expresses that all the dishes have the same price.

The class $\mathtt{TAG}[\,|.|_{\mathbb{Z}}]$ has been studied under different names (e.g. Parikh automata in [KR02], linear constraint tree automata in [BMSL09]) and it has a decidable emptiness test. Indeed, the set of successful runs of a given $\mathtt{TA}$ with state set Q is a context-free language (seeing runs as words of Q^*), and the Parikh projection (the set of tuples over $\mathbb{N}^{|Q|}$ whose components are the $[\![\, |q| \,]\!]_r$ for every run r) of such a language is a semi-linear set. The idea for deciding emptiness for a $\mathtt{TAG}[\,|.|_{\mathbb{Z}}]$ $\mathcal{A}$ is to compute this semi-linear set and to test the emptiness of its intersection with the set of solutions in $\mathbb{N}^{|Q|}$ of $C_{\mathcal{A}}$, the arithmetic constraint of $\mathcal{A}$ (a Boolean combination of linear inequalities of type $|.|_{\mathbb{Z}}$) which is also semi-linear. This can be done in NPTIME, see [BMSL09].

To our knowledge, the class $\texttt{TAG}[\,\|.\|_{\mathbb{Z}}]$ with global constraints counting the number of distinct subterms in each state, has not been studied, even modulo an empty theory.

Combining constraints of type $\approx$ and counting constraints of type $|.|_{\mathbb{Z}}$ however leads to undecidability.

Theorem 4.2. *Emptiness is undecidable for* $\texttt{TAG}^{\wedge}[\approx, |.|_{\mathbb{Z}}]$.

Proof. We consider the Hilbert's tenth problem, that is, solvability of an input equation $P = 0$ where P is a polynomial with integer coefficients and variables ranging over the natural numbers. This problem is known undecidable, and with the addition of new variables it is easily reducible to a question of the form $\exists x_1 \ldots \exists x_n : e_1 \wedge \ldots \wedge e_m$, where $x_1, \ldots, x_n$ are variables ranging over the natural numbers, and $e_1, \ldots, e_m$ are equations that are either of the form $x_j + x_k = x_t$ or $x_j * x_k = x_t$ or $x_j = 1$ or $x_j = 0$. We reduce this last problem to emptiness of $\texttt{TAG}^{\wedge}[\approx, |.|_{\mathbb{Z}}]$.

We consider an instance $\varphi \equiv \exists x_1 \ldots \exists x_n : e_1 \wedge \ldots \wedge e_m$. Without loss of generality, we assume that $e_1, \ldots, e_{m'}$ for $m' \leq m$ are all the equations of the form $x_j * x_k = x_t$, and that for each of such equations, the indexes j, k, t are different. We will construct a $\texttt{TAG}^{\wedge}[\approx, |.|_{\mathbb{Z}}]$ $\mathcal{A}$ such that φ is true if and only if $\mathcal{L}(\mathcal{A})$ is not empty.

Since the construction of $\mathcal{A}$ is technical, let us give first some intuitions (see Figure 3). Consider a possible assignment $x_1 := v_1, \ldots, x_n := v_n$. A concrete run of $\mathcal{A}$ will be able to check whether this assignment proves that φ is true, and only accept the corresponding term if the answer is positive. In this run, there will be v_1 occurrences of state $q_{|x_1|}$, v_2 occurrences of state $q_{|x_2|}$, and so on. Equations of the form $x_j + x_k = x_t$, $x_j = 1$ and $x_j = 0$ can directly be checked by constraints of the form $|q_{|x_j|}| + |q_{|x_k|}| = |q_{|x_t|}|$, $|q_{|x_j|}| = 1$ and $|q_{|x_j|}| = 0$.

For each equation e_i of the form $x_j * x_k = x_t$ there will be v_k occurrences of a state called $q_{e_i,|x_k|}$. This is ensured by the constraint $|q_{e_i,|x_k|}| = |q_{|x_k|}|$. Under each of these occurrences, there will be the same term, reaching a state q_{e_i,x_j}, and containing v_j occurrences of a state $q_{e_i,|x_t|}$. The uniqueness of this term, as well as the number of occurrences of $q_{e_i,|x_t|}$, are both ensured by an equality constraint $q_{x_j} \approx q_{e_i,x_j}$. In summary, there will be $v_j * v_k$ occurrences of state $q_{e_i,|x_t|}$. The satisfiability of the equation $x_j * x_k = x_t$ will be checked by the constraint $|q_{|x_t|}| = |q_{e_i,|x_t|}|$.

The components of the $\texttt{TAG}^{\wedge}[\approx, |.|_{\mathbb{Z}}]$ $\mathcal{A} = \langle Q, \Sigma, F, \Delta, C \rangle$ are defined as follows:

$$\begin{array}{rcl}
Q & = & \{q_{\mathtt{accept}},\ q_a\} \cup \{q_{|x_j|},\ q_{x_j} \mid j \in \{1, \ldots, n\}\} \cup \{q_{e_i} \mid i \in \{1, \ldots, m'\}\} \cup \\
& & \{q_{e_i,x_j},\ q_{e_i,|x_t|},\ q_{e_i,|x_k|} \mid i \in \{1, \ldots, m'\}, e_i \equiv x_j * x_k = x_t\} \\
\Sigma & = & \{a:0,\ g:1,\ h:2,\ f:n+m'\} \\
F & = & \{q_{\mathtt{accept}}\} \\
\Delta & = & \{a \to q_a,\ f(q_{x_1}, \ldots, q_{x_n}, q_{e_1}, \ldots, q_{e_{m'}}) \to q_{\mathtt{accept}}\} \cup \\
& & \{g(q_a) \to q_{|x_j|},\ g(q_a) \to q_{x_j},\ g(q_{|x_j|}) \to q_{|x_j|},\ g(q_{|x_j|}) \to q_{x_j} \mid j \in \{1, \ldots, n\}\} \cup \\
& & \{g(q_a) \to q_{e_i,|x_t|},\ g(q_a) \to q_{e_i,x_j},\ g(q_{e_i,|x_t|}) \to q_{e_i,|x_t|},\ g(q_{e_i,|x_t|}) \to q_{e_i,x_j}, \\
& & \ \ h(q_{e_i,x_j}, q_a) \to q_{e_i,|x_k|},\ h(q_{e_i,x_j}, q_{e_i,|x_k|}) \to q_{e_i,|x_k|},\ h(q_a, q_{e_i,|x_k|}) \to q_{e_i}, \\
& & \ \ h(q_a, q_a) \to q_{e_i} \mid i \in \{1, \ldots, m'\}, e_i \equiv x_j * x_k = x_t\}
\end{array}$$

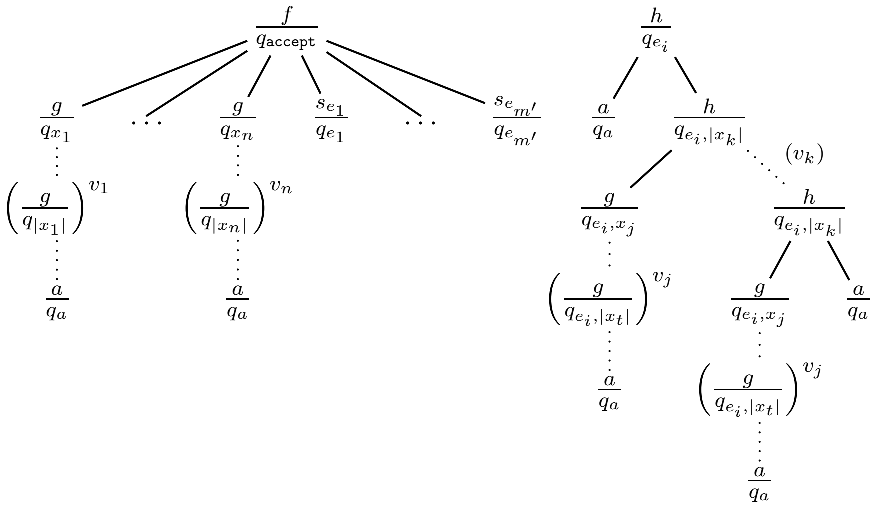
$\frac{f}{q_{\mathsf{accept}}}$
$\frac{g}{q_{x_1}}$
$\frac{g}{q_{x_n}}$
$\frac{s_{e_1}}{q_{e_1}}$
$\frac{s_{e_{m'}}}{q_{e_{m'}}}$
$\left(\frac{g}{q_{|x_1|}}\right)^{v_1}$
$\left(\frac{g}{q_{|x_n|}}\right)^{v_n}$
$\frac{a}{q_a}$
$\frac{h}{q_{e_i}}$
$\frac{h}{q_{e_i,|x_k|}}$
(v_k)
$\frac{g}{q_{e_i,x_j}}$
$\left(\frac{g}{q_{e_i,|x_t|}}\right)^{v_j}$

Figure 3: Accepting run of $s = f(g^{v_1+1}(a), \dots, g^{v_n+1}(a), s_{e_1}, \dots, s_{e_{m'}})$ and the subrun of s_{e_i}, where e_i is of the form $x_j * x_k = x_t$.

$$\begin{array}{rcl} C & = & \bigwedge_{m'<i\leq m, e_i\equiv x_j+x_k=x_t} |q_{|x_j|}| + |q_{|x_k|}| = |q_{|x_t|}| \wedge \\ & & \bigwedge_{m'<i\leq m, e_i\equiv x_j=1} |q_{|x_j|}| = 1 \wedge \\ & & \bigwedge_{m'<i\leq m, e_i\equiv x_j=0} |q_{|x_j|}| = 0 \wedge \\ & & \bigwedge_{1\leq i\leq m', e_i\equiv x_j*x_k=x_t} \left(|q_{e_i,|x_t|}| = |q_{|x_t|}| \wedge |q_{e_i,|x_k|}| = |q_{|x_k|}| \wedge q_{x_j} \approx q_{e_i,x_j}\right) \end{array}$$

It remains to prove that φ is true if and only if $\mathcal{L}(\mathcal{A})$ is not empty. To this end, let us first assume that $x_1 := v_1, \dots, x_n := v_n$ is a solution of φ. In order to simplify the presentation, we denote the term $h(a, h(s, h(s, \dots, h(s, a) \dots)))$, with k occurrences of s, by $h[a, s, \dots (k) \dots, s, a]$, and given an equation $e_i \equiv x_j * x_k = x_t$, we denote the term $h[a, g^{v_j+1}(a), \dots (v_k) \dots, g^{v_j+1}(a), a]$ by s_{e_i}. Let us consider the term $s = f(g^{v_1+1}(a), \dots, g^{v_n+1}(a), s_{e_1}, \dots, s_{e_{m'}})$. It is not difficult to see that the run of Figure 3 is an accepting run of s. Note that for each equation $e_i \equiv x_j * x_k = x_t$, the constraints $|q_{e_i,|x_t|}| = |q_{|x_t|}|$, $|q_{e_i,|x_k|}| = |q_{|x_k|}|$, $q_{x_j} \approx q_{e_i,x_j}$ are satisfied, since $x_j := v_j$, $x_k := v_k$, $x_t := v_t$ satisfies the equation.

Now, assume that there is an accepting run r of $\mathcal{A}$ on a term s. Since r is accepting, the transition rule $f(q_{x_1}, \dots, q_{x_n}, q_{e_1}, \dots, q_{e_{m'}}) \to q_{\mathsf{accept}}$ is applied at the root of s. According to the form of the rules involving $q_{x_1}, \dots, q_{x_n}$, it holds that s is of the form $s = f(g^{v_1+1}(a), \dots, g^{v_n+1}(a), s_{e_1}, \dots, s_{e_{m'}})$, for some natural numbers $v_1, \dots, v_n$ and some terms $s_{e_1}, \dots, s_{e_{m'}}$. Moreover, the states $q_{|x_1|}, \dots, q_{|x_n|}$ have $v_1, \dots, v_n$ occurrences, respectively. It remains to see that the assignment $x_1 := v_1, \dots, x_n := v_n$ makes φ true. The satisfiability of a constraint of the form $|q_{|x_j|}| + |q_{|x_k|}| = |q_{|x_t|}|$ (or $|q_{|x_j|}| = 1$ or $|q_{|x_j|}| = 0$) implies that $v_j + v_k = v_t$ (or $v_j = 1$ or $v_j = 0$), thus an equation of the form $x_j + x_k = x_t$ (or $x_j = 1$ or $x_j = 0$) holds with this assignment. It remains to see that every equation e_i of the form $x_j * x_k = x_t$ also holds with this assignment. According to the form of the rules of $\mathcal{A}$ and the satisfiability of the constraints $|q_{e_i,|x_k|}| = |q_{|x_k|}|, q_{x_j} \approx q_{e_i,x_j}$, the term s_{e_i} is of the form $h[a, g^{v_j+1}(a), \dots (v_k) \dots, g^{v_j+1}(a), a]$. Moreover, $|q_{e_i,|x_t|}|$ has $v_j * v_k$ occurrences.

Therefore, by the satisfiability of the constraint $|q_{e_i,|x_t|}| = |q_{|x_t|}|$, it follows $v_j * v_k = v_t$, and hence the equation $x_j * x_k = x_t$ holds with this assignment, and we are done. $\square$

4.2. **Global Natural Linear Constraints.** We present now a restriction on linear inequalities which enables a decidable emptiness test when combined with $\approx$ and $\not\approx$ as global constraints. A *natural linear inequality* over Q is a linear inequality as above whose coefficients a_q and a all have the same sign. We call them natural since it is equivalent to consider inequalities in both directions whose coefficients are all non-negative, like $\sum a_q \cdot |q| \leq a$, with $a_q, a \in \mathbb{N}$, to refer to $\sum -a_q \cdot |q| \geq -a$. We also consider linear equalities $\sum a_q \cdot |q| = a$, with $a_q, a \in \mathbb{N}$, to refer to a conjunction of two natural linear inequalities.

The types of the natural linear inequalities are denoted by $|.|_{\mathbb{N}}$ and $\|.\|_{\mathbb{N}}$. Below, we shall abbreviate these two types by $\mathbb{N}$.

The main difference between the linear inequalities of type $|.|_{\mathbb{Z}}$ and $|.|_{\mathbb{N}}$ (and respectively $\|.\|_{\mathbb{Z}}$ and $\|.\|_{\mathbb{N}}$) is that the former permits to compare the respective number of occurrences of two states, like e.g. in $|q| \leq |q'|$, whereas the latter only permits to compare the number of occurrences of one state (or a sum of the number occurrences of several states with coefficients) to a constant as e.g. in $|q| \leq 4$ or $|q| + 2|q'| \leq 9$.

In the rest of the subsection we show that $\mathtt{TABG}[\approx, \not\approx, \mathbb{N}]$ has the same expressiveness as $\mathtt{TABG}^\wedge[\approx, \not\approx]$. The proof works in several steps:

- First, we define the notion of normalized $\mathtt{TABG}[\approx, \not\approx, \mathbb{N}]$, that is a $\mathtt{TABG}[\approx, \not\approx, \mathbb{N}]$ with a constraint being a disjunction of conjunctions of literals in a simple form.
- Second, we remove negative literals of the form $\neg(q \approx q')$ or $\neg(q \not\approx q')$, obtaining a list of $\mathtt{TABG}^\wedge[\approx, \not\approx, \mathbb{N}]$ such that the union of their languages coincides with the language of the original $\mathtt{TABG}[\approx, \not\approx, \mathbb{N}]$. In this step we use arithmetic constraints for simulating the removed negative literals.
- Third, we remove arithmetic literals of type $\|.\|_{\mathbb{N}}$, obtaining a new list of $\mathtt{TABG}^\wedge[\approx, \not\approx, |.|_{\mathbb{N}}]$ such that the union of their languages coincides with the language of the original $\mathtt{TABG}[\approx, \not\approx, \mathbb{N}]$. In this step we use positive literals of types $\approx$, $\not\approx$, and $|.|_{\mathbb{N}}$ in order to simulate the removed literals of type $\|.\|_{\mathbb{N}}$.
- Fourth, we remove arithmetic literals of type $|.|_{\mathbb{N}}$, obtaining a new list of $\mathtt{TABG}^\wedge[\approx, \not\approx]$ such that the union of their languages coincides with the language of the original $\mathtt{TABG}[\approx, \not\approx, \mathbb{N}]$. In this step, new states are used for counting the amount of occurrences of original states.
- Finally, we show that $\mathtt{TABG}^\wedge[\approx, \not\approx]$ are closed under union. Hence, we obtain a single $\mathtt{TABG}^\wedge[\approx, \not\approx]$ whose language coincides with the one of the original $\mathtt{TABG}[\approx, \not\approx, \mathbb{N}]$.

Definition 4.3. Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C \rangle$ be a $\mathtt{TABG}[\approx, \not\approx, \mathbb{N}]$. The constraint C is *normalized* if it is either *true* or *false* or a disjunction of conjunctions of literals, where all arithmetic literals are positive.

Remember that the form of the positive arithmetic literals can be either $a_1\|q_1\| + \ldots + a_n\|q_n\| \otimes k$ or $a_1|q_1| + \ldots + a_n|q_n| \otimes k$, with $\otimes$ in $\{\geq, \leq, =\}$, $n > 0$, $k \geq 0$ and strictly positive $a_1, \ldots, a_n$.

Lemma 4.4. *Any $\mathtt{TABG}[\approx, \not\approx, \mathbb{N}]$ can be effectively transformed into a normalized $\mathtt{TABG}[\approx, \not\approx, \mathbb{N}]$ with the same equational theory and preserving the language.*

Proof. First, by applying de Morgan laws, negations are moved inwards so that each negation is applied to just an atom. Second, negative arithmetic literals are made positive by simple transformations: inequalities are inverted and equalities become disjunctions of inequalities. Third, strict inequalities are converted into non-strict by adding or subtracting 1 to a side. Fourth, by applying simple arithmetic operations all such literals are made of the required form $a_1\|q_1\|+\ldots+a_n\|q_n\|\otimes k$ or $a_1|q_1|+\ldots+a_n|q_n|\otimes k$, for $\otimes$ in $\{\geq,\leq,=\}$, $n>0$ and strictly positive $a_1,\ldots,a_n$. In this step, a trivially false literal is replaced by *false*, and a trivially true literal is replaced by *true*. Finally, by applying the standard transformation into disjunctive conjunctive normal form we get the desired result. □

In order to remove negative equality and disequality literals and positive arithmetic constraints, we use the idea of inserting new states which are *synonyms* of existing states. Intuitively, a synonym is a new state $\hat{q}$ that behaves analogous to an existing state $\bar{q}$, i.e. the rules and constraints are modified such that the relation of $\hat{q}$ with the other states is the same as for $\bar{q}$. Nevertheless, the constraints are further modified to ensure that, whenever $\bar{q}$ occurs in an execution, $\hat{q}$ also occurs. Moreover, all subterms reaching $\hat{q}$ are the same (or equivalent modulo the relation induced by the flat theory), but are different from (non-equivalent to) the ones reaching $\bar{q}$. This way, an execution of the original automaton with occurrences of $\bar{q}$ can be transformed into an execution of the new automaton, where the occurrences of a concrete subterm (up to the equivalence relation) reaching $\bar{q}$ in the original execution now reach $\hat{q}$ instead.

Definition 4.5. Let $\mathcal{A}=\langle Q,\Sigma,F,\Delta,E,C\rangle$ be a $\texttt{TABG}[\approx,\not\approx,\mathbb{N}]$. Let $\bar{q}$ be a state in Q. Let $\hat{q}$ be a state not in Q.

We define $F_{\bar{q}\rightsquigarrow\hat{q}}$ as F if $\bar{q}$ is not in F, and as $F\cup\{\hat{q}\}$ if $\bar{q}$ is in F.

We define $\Delta_{\bar{q}\rightsquigarrow\hat{q}}$ as the set of rules obtained from the rules of Δ with all possible replacements of occurrences of $\bar{q}$ by $\hat{q}$. More formally, $\Delta_{\bar{q}\rightsquigarrow\hat{q}}$ is $\{f(q'_1,\ldots,q'_n)\to q'_{n+1} \mid \exists f(q_1,\ldots,q_n)\to q_{n+1}\in\Delta : \forall i\in\{1,\ldots,n+1\} : (q_i=q'_i\vee(q_i=\bar{q}\wedge q'_i=\hat{q}))\}$.

We define $C_{\bar{q}\rightsquigarrow\hat{q}}$ as the constraint $\big((\|\bar{q}\|=0\wedge\|\hat{q}\|=0)\vee(\|\hat{q}\|=1\wedge\bar{q}\not\approx\hat{q})\big)\wedge C'$, where C' is obtained from the normalization of C by replacing each literal by a new formula according to the following description.

- Each literal $(q_1\approx q_2)$ is replaced by the conjunction of the literals of the set $\big\{q'_1\approx q'_2 \mid ((q'_1=q_1\vee(q_1=\bar{q}\wedge q'_1=\hat{q}))\wedge(q'_2=q_2\vee(q_2=\bar{q}\wedge q'_2=\hat{q})))\big\}$.
- Each literal $(q_1\not\approx q_2)$ is replaced by the conjunction of the literals of the set $\big\{q'_1\not\approx q'_2 \mid ((q'_1=q_1\vee(q_1=\bar{q}\wedge q'_1=\hat{q}))\wedge(q'_2=q_2\vee(q_2=\bar{q}\wedge q'_2=\hat{q})))\big\}$.
- Each literal $\neg(q_1\approx q_2)$ is replaced by the disjunction of the literals of the set $\big\{\neg(q'_1\approx q'_2) \mid ((q'_1=q_1\vee(q_1=\bar{q}\wedge q'_1=\hat{q}))\wedge(q'_2=q_2\vee(q_2=\bar{q}\wedge q'_2=\hat{q})))\big\}$.
- Each literal $\neg(q_1\not\approx q_2)$ is replaced by the disjunction of the literals of the set $\big\{\neg(q'_1\not\approx q'_2) \mid ((q'_1=q_1\vee(q_1=\bar{q}\wedge q'_1=\hat{q}))\wedge(q'_2=q_2\vee(q_2=\bar{q}\wedge q'_2=\hat{q})))\big\}$.
- Each occurrence of $|\bar{q}|$ is replaced by $|\bar{q}|+|\hat{q}|$, and each occurrence of $\|\bar{q}\|$ is replaced by $\|\bar{q}\|+\|\hat{q}\|$.

We define $\mathcal{A}_{\bar{q}\rightsquigarrow\hat{q}}$ as $\langle Q\cup\{\hat{q}\},\Sigma,F_{\bar{q}\rightsquigarrow\hat{q}},\Delta_{\bar{q}\rightsquigarrow\hat{q}},E,C_{\bar{q}\rightsquigarrow\hat{q}}\rangle$.

We write $(F_{\bar{q}\rightsquigarrow\hat{q}})_{\bar{q}'\rightsquigarrow\hat{q}'}$ for $\hat{q}\neq\bar{q}'$ and $\hat{q}\neq\hat{q}'$ more succinctly as $F_{\bar{q},\bar{q}'\rightsquigarrow\hat{q},\hat{q}'}$, and similarly for $\Delta_{\bar{q},\bar{q}'\rightsquigarrow\hat{q},\hat{q}'}$, $C_{\bar{q},\bar{q}'\rightsquigarrow\hat{q},\hat{q}'}$ and $\mathcal{A}_{\bar{q},\bar{q}'\rightsquigarrow\hat{q},\hat{q}'}$.

The condition $(\|\bar{q}\|=0\wedge\|\hat{q}\|=0)$ added to $C_{\bar{q}\rightsquigarrow\hat{q}}$ is necessary to satisfy $\mathcal{L}(\mathcal{A}_{\bar{q}\rightsquigarrow\hat{q}})=\mathcal{L}(\mathcal{A})$, as it is proved in Lemma 4.6. This lemma is not used in the rest of the article, since the introduction of synonyms is combined with other constraints in further transformations. Nevertheless, we preserve Lemma 4.6 since its proof gives intuition about the definition of synonyms, and the arguments are similar to other ones appearing later.

Lemma 4.6. *Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ be a* TABG$[\approx, \not\approx, \mathbb{N}]$. *Let $\bar{q}$ be a state in Q. Let $\hat{q}$ be a state not in Q.*

Then, $\mathcal{L}(\mathcal{A}_{\bar{q}\rightsquigarrow\hat{q}}) = \mathcal{L}(\mathcal{A})$.

Proof. Accepting runs of $\mathcal{A}$ having no occurrence of $\bar{q}$ are also accepting runs of $\mathcal{A}_{\bar{q}\rightsquigarrow\hat{q}}$. An accepting run of $\mathcal{A}$ having occurrences of $\bar{q}$ can be converted into an accepting run of $\mathcal{A}_{\bar{q}\rightsquigarrow\hat{q}}$ by choosing one subterm t reaching $\bar{q}$ and replacing $\bar{q}$ by $\hat{q}$ at all positions with subterms equivalent to t by the relation induced by E.

Accepting runs of $\mathcal{A}_{\bar{q}\rightsquigarrow\hat{q}}$ can be converted into accepting runs of $\mathcal{A}$ by replacing each occurrence of $\hat{q}$ by $\bar{q}$. □

The following lemma makes use of synonyms in order to remove a negative literal of the form $\neg(\bar{q} \approx \bar{q}')$ preserving the language. The next one, Lemma 4.8, analogously permits to remove a negative literal of the form $\neg(\bar{q} \not\approx \bar{q}')$.

Lemma 4.7. *Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ be a* TABG$[\approx, \not\approx, \mathbb{N}]$. *Let $\bar{q}$, $\bar{q}'$ be states in Q. Let $\hat{q}$, $\hat{q}'$ be distinct states not in Q. Let C be of the form $\neg(\bar{q} \approx \bar{q}') \wedge C'$. Let $\mathcal{A}'$ be $\langle Q \cup \{\hat{q}, \hat{q}'\}, \Sigma, F_{\bar{q},\bar{q}'\rightsquigarrow\hat{q},\hat{q}'}, \Delta_{\bar{q},\bar{q}'\rightsquigarrow\hat{q},\hat{q}'}, E, (\|\hat{q}\| = 1 \wedge \|\hat{q}'\| = 1 \wedge \hat{q} \not\approx \hat{q}') \wedge C'_{\bar{q},\bar{q}'\rightsquigarrow\hat{q},\hat{q}'}\rangle$.*

Then, $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$ holds.

Proof. Accepting runs of $\mathcal{A}$ can be converted into accepting runs of $\mathcal{A}'$ as follows. First, we choose two subterms $\bar{t}$ and $\bar{t}'$ different modulo the equivalence relation induced by E and reaching $\bar{q}$ and $\bar{q}'$, respectively. Note that these terms must exist in order to satisfy the literal $\neg(\bar{q} \approx \bar{q}')$ of C. Second, we replace $\bar{q}$ by $\hat{q}$ at all the positions with subterms equivalent to $\bar{t}$ by the relation induced by E. Similarly, we replace $\bar{q}'$ by $\hat{q}'$ at all the positions with subterms equivalent to $\bar{t}'$ by the relation induced by E. This way, the subconstraint $\|\hat{q}\| = 1 \wedge \|\hat{q}'\| = 1 \wedge \hat{q} \not\approx \hat{q}'$ is satisfied, but also $C'_{\bar{q},\bar{q}'\rightsquigarrow\hat{q},\hat{q}'}$ is satisfied.

Accepting runs of $\mathcal{A}'$ can be converted into accepting runs of $\mathcal{A}$ by replacing each occurrence of $\hat{q}$ by $\bar{q}$, and each occurrence of $\hat{q}'$ by $\bar{q}'$. Note that the subconstraint $\|\hat{q}\| = 1 \wedge \|\hat{q}'\| = 1 \wedge \hat{q} \not\approx \hat{q}'$ ensures the existence of such occurrences, and with subterms which are different modulo the equivalence relation induced by E. Thus, the literal $\neg(\bar{q} \approx \bar{q}')$ of C is satisfied. The constraint C' is also satisfied. □

Lemma 4.8. *Consider the same assumptions as in Lemma 4.7, except that C is of the form $\neg(\bar{q} \not\approx \bar{q}') \wedge C'$ and the constraint of $\mathcal{A}'$ is $(\|\hat{q}\| = 1 \wedge \|\hat{q}'\| = 1 \wedge \hat{q} \approx \hat{q}') \wedge C'_{\bar{q},\bar{q}'\rightsquigarrow\hat{q},\hat{q}'}$*

Then, $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$ holds.

Proof. Analogous to the proof of Lemma 4.8. □

The following definition will be used to remove literals of type $\|.\|_{\mathbb{N}}$.

Definition 4.9. Let C be a constraint, and let k be a natural number. By $C_{\|\bar{q}\|\rightsquigarrow k}$ we define the constraint obtained from C by replacing all occurrences of $\|\bar{q}\|$ by k.

The following two lemmas show how to remove literals of the form $\|q\| = 1$ or $\|q\| = 0$ preserving the language.

Lemma 4.10. *Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ be a* TABG$[\approx, \not\approx, \mathbb{N}]$. *Let $\bar{q}$ be a state in Q. Let C be of the form $\|\bar{q}\| = 1 \wedge C'$. Let $\mathcal{A}'$ be $\langle Q, \Sigma, F, \Delta, E, |\bar{q}| \geq 1 \wedge \bar{q} \approx \bar{q} \wedge C'_{\|\bar{q}\|\rightsquigarrow 1}\rangle$.*

Then, $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$ holds.

Proof. Accepting runs of $\mathcal{A}'$ and $\mathcal{A}$ coincide because the constraints C and $C_{\mathcal{A}'}$ have the same semantics. □

Lemma 4.11. *Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ be a* TABG$[\approx, \not\approx, \mathbb{N}]$. *Let $\bar{q}$ be a state in Q. Let C be of the form $\|\bar{q}\| = 0 \wedge C'$. Let $\mathcal{A}'$ be $\langle Q, \Sigma, F, \Delta, E, |\bar{q}| = 0 \wedge C'_{\|\bar{q}\| \rightsquigarrow 0}\rangle$.*

Then, $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$ holds.

Proof. Accepting runs of $\mathcal{A}'$ and $\mathcal{A}$ coincide because the constraints C and $C_{\mathcal{A}'}$ have the same semantics. □

Now, we will use the above lemmas in order to iteratively remove all negative literals and the arithmetic literals of type $\|.\|_{\mathbb{N}}$. Each removal step is not defined for arbitrary normalized TABG$[\approx, \not\approx, \mathbb{N}]$, but just for normalized conjunctive TABG$[\approx, \not\approx, \mathbb{N}]$. For this reason, we first describe how to transform a given normalized TABG$[\approx, \not\approx, \mathbb{N}]$ into a list of normalized conjunctive TABG$[\approx, \not\approx, \mathbb{N}]$ such that, the union of their languages coincides with the language of the original TABG$[\approx, \not\approx, \mathbb{N}]$.

Definition 4.12. Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ be a normalized TABG$[\approx, \not\approx, \mathbb{N}]$, such that C is of the form $C_1 \vee C_2 \vee \ldots \vee C_n$ for conjunctive constraints $C_1, C_2, \ldots, C_n$. Let $\mathcal{A}_1 = \langle Q, \Sigma, F, \Delta, E, C_1\rangle, \mathcal{A}_2 = \langle Q, \Sigma, F, \Delta, E, C_2\rangle, \ldots, \mathcal{A}_n = \langle Q, \Sigma, F, \Delta, E, C_n\rangle$. These automata are conjunctive and normalized and, moreover, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2) \cup \ldots \cup \mathcal{L}(\mathcal{A}_n)$ holds. We say that $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ is the *subdivision* of $\mathcal{A}$.

Iteratively, we will transform a list of normalized conjunctive TABG$[\approx, \not\approx, \mathbb{N}]$ into a new list of automata of the same kind but with simplified constraints, preserving the language. In order to show that this process terminates, we define a measure on normalized conjunctive TABG$[\approx, \not\approx, \mathbb{N}]$ which will decrease at each step. Moreover, a case with minimal measure corresponds to a positive TABG$[\approx, \not\approx, |.|_{\mathbb{N}}]$. This measure is a pair of natural numbers which depends on the constraint C of the normalized conjunctive TABG$[\approx, \not\approx, \mathbb{N}]$. In the first component we have the amount of negative literals in C. In the second component we have the addition of the isolated constants in all arithmetic literal constraints of type $\|.\|_{\mathbb{N}}$ plus the number of uses of the function symbol $\|.\|_{\mathbb{N}}$.

Definition 4.13. We define the measure of a normalized conjunctive constraint C, denoted $\langle C\rangle$ as a pair of natural numbers. We describe it by distinguishing the following cases.

- If C is of the form $q_1 \approx q_2$ or $q_1 \not\approx q_2$, then its measure is $\langle 0, 0\rangle$.
- If C is of the form $\neg(q_1 \approx q_2)$ or $\neg(q_1 \not\approx q_2)$, then its measure is $\langle 1, 0\rangle$.
- If C is of the form $(a_1\|q_1\| + \ldots + a_n\|q_n\| \otimes k)$, where $\otimes$ is in $\{=, \geq, \leq\}$, then its measure is $\langle 0, n + k\rangle$.
- If C is of the form $(a_1|q_1| + \ldots + a_n|q_n| \otimes k)$, where $\otimes$ is in $\{=, \geq, \leq\}$, then its measure is $\langle 0, 0\rangle$.
- If C is either *true* or *false*, then its measure is $\langle 0, 0\rangle$.
- If C is a conjunction of two or more literals $l_1 \wedge l_2 \wedge \ldots \wedge l_n$ with measures $\langle a_1, b_1\rangle$, $\langle a_2, b_2\rangle, \ldots, \langle a_n, b_n\rangle$, then its measure is $\langle a_1 + a_2 + \ldots + a_n, b_1 + b_2 + \ldots + b_n\rangle$.

Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ be a normalized conjunctive TABG$[\approx, \not\approx, \mathbb{N}]$. The measure of $\mathcal{A}$, denoted $\langle\mathcal{A}\rangle$ is defined as $\langle C\rangle$.

We say that $\mathcal{A}_1$ is bigger than $\mathcal{A}_2$ (or, equivalently, that $\mathcal{A}_2$ is smaller than $\mathcal{A}_1$), denoted $\mathcal{A}_1 > \mathcal{A}_2$ (or $\mathcal{A}_2 < \mathcal{A}_1$), if the measure of $\mathcal{A}_1$ is bigger (or smaller) than the measure of $\mathcal{A}_2$, according to the lexicographic extension of the relation $>$ of natural numbers.

The following lemma shows that any normalized conjunctive TABG$[\approx, \not\approx, \mathbb{N}]$ with non-minimal measure can be transformed into a list of TABG$[\approx, \not\approx, \mathbb{N}]$ of the same kind with smaller measures and preserving the language.

Lemma 4.14. *Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ be a normalized conjunctive* TABG$[\approx, \not\approx, \mathbb{N}]$ *whose measure is not $\langle 0, 0\rangle$.*

Then one can construct normalized conjunctive TABG$[\approx, \not\approx, \mathbb{N}]$ *$\mathcal{A}_1, \dots, \mathcal{A}_n$ with the same equational theory E, each of them having a measure smaller than $\langle\mathcal{A}\rangle$ and such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cup \dots \cup \mathcal{L}(\mathcal{A}_n)$ holds.*

Proof. In the case where C has some negative literal $\neg(q \approx q')$ or $\neg(q \not\approx q')$, the transformations described in Lemmas 4.7 and 4.8 give a new TABG$[\approx, \not\approx, \mathbb{N}]$ $\mathcal{A}'$, and the subdivision $\mathcal{A}_1, \dots, \mathcal{A}_n$ of the normalization of $\mathcal{A}'$ (as defined in Definition 4.12) is such that the constraints $C_{\mathcal{A}_1}, \dots, C_{\mathcal{A}_n}$ have one less negative literal than C. Thus, the measure of each of these automata is smaller than the measure of $\mathcal{A}$.

In the case where C has no negative literals of the form $\neg(q \approx q')$ or $\neg(q \not\approx q')$, its measure is of the form $\langle 0, m\rangle$ for $m > 0$. It follows that there is at least one literal of the form $(a\|\bar{q}\| + \sum a_i \cdot \|q_i\| \otimes k)$, where $\otimes$ is in $\{=, \geq, \leq\}$. We consider a new state $\hat{q}$ and the automaton $\mathcal{A}_{\bar{q} \leadsto \hat{q}}$. Its constraint $C_{\bar{q} \leadsto \hat{q}}$ is of the form $\big((\|\bar{q}\| = 0 \wedge \|\hat{q}\| = 0) \vee (\|\hat{q}\| = 1 \wedge \bar{q} \not\approx \hat{q})\big) \wedge C'$. Note that, according to Definition 4.5, C' is a conjunction because there are no negative literals of the form $\neg(q \approx q')$ or $\neg(q \not\approx q')$ in C. Thus, $C_{\bar{q} \leadsto \hat{q}}$ can be rewritten as the disjunction of two conjunctions C_1 and C_2, where C_1 is $\|\bar{q}\| = 0 \wedge \|\hat{q}\| = 0 \wedge C'$ and C_2 is $\|\hat{q}\| = 1 \wedge \bar{q} \not\approx \hat{q} \wedge C'$. Hence, the subdivision of the normalization of $\mathcal{A}_{\bar{q} \leadsto \hat{q}}$ are the automata $\mathcal{A}_1, \mathcal{A}_2$ obtained from $\mathcal{A}_{\bar{q} \leadsto \hat{q}}$ by replacing its constraint by C_1 and C_2, respectively. The measures of C_1 and C_2 may be bigger than the one of C. In order to conclude, for each case we show that additional transformations can be applied to $\mathcal{A}_1$ and $\mathcal{A}_2$, producing automata with smaller measures than the one of $\mathcal{A}$ and preserving the represented language.

- The literals of C_1 of type $\|.\|_{\mathbb{N}}$ are $\|\bar{q}\| = 0$ and $\|\hat{q}\| = 0$, and those obtained from the literals of C of type $\|.\|_{\mathbb{N}}$ by replacing $\|\bar{q}\|$ by $\|\bar{q}\| + \|\hat{q}\|$. Note that original literals of the form $(a\|\bar{q}\| + \sum a_i \cdot \|q_i\| \otimes k)$ have been converted into $(a\|\bar{q}\| + a\|\hat{q}\| + \sum a_i \cdot \|q_i\| \otimes k)$, and recall that there is at least one literal of this form in C. Applying to $\mathcal{A}_1$ the transformation described in Lemma 4.11 for $\bar{q}$ and $\hat{q}$, each one of the above literals is transformed into $(a \cdot 0 + a \cdot 0 + \sum a_i \cdot \|q_i\| \otimes k)$, which has a smaller measure than the original literal $(a\|\bar{q}\| + \sum a_i \cdot \|q_i\| \otimes k)$. Moreover, the literals $\|\bar{q}\| = 0$ and $\|\hat{q}\| = 0$ are converted into $|\bar{q}| = 0$ and $|\hat{q}| = 0$, respectively. In summary, the measure of $((C_1)_{\bar{q} \leadsto 0})_{\hat{q} \leadsto 0}$ is smaller than the one of C.
- Similarly, the literals of C_2 of type $\|.\|_{\mathbb{N}}$ are $\|\hat{q}\| = 1$ and those obtained from the literals of C of type $\|.\|_{\mathbb{N}}$ by replacing $\|\bar{q}\|$ by $\|\bar{q}\| + \|\hat{q}\|$. As above, note that original literals of the form $(a\|\bar{q}\| + \sum a_i \cdot \|q_i\| \otimes k)$ have been transformed into $(a\|\bar{q}\| + a\|\hat{q}\| + \sum a_i \cdot \|q_i\| \otimes k)$, and recall that there is at least one literal of this form in C. Applying to C_2 the transformation described in Lemma 4.10 for $\hat{q}$, each one of the above literals is converted into $(a \cdot \|\bar{q}\| + a \cdot 1 + \sum a_i \cdot \|q_i\| \otimes k)$. The normalization of such a literal is the normalization of $(a \cdot \|\bar{q}\| + \sum a_i \cdot \|q_i\| \otimes k - a)$, which might be already normalized or must be replaced by *true* or *false* in order to normalize it, depending on $k - a$ and $\otimes$. In every case, the resulting literal has a smaller measure than the original literal $(a\|\bar{q}\| + \sum a_i \cdot \|q_i\| \otimes k)$. Moreover, the literal $\|\hat{q}\| = 1$ is replaced by $|\hat{q}| \geq 1 \;\wedge\; \hat{q} \approx \hat{q}$ as a consequence of the transformation of Lemma 4.10. To summarize, the measure of $(C_2)_{\hat{q} \leadsto 1}$ is smaller than the one of C. □

Corollary 4.15. *Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ be a* TABG$[\approx, \not\approx, \mathbb{N}]$.

Then, one can construct some TABG$^{\wedge}[\approx, \not\approx, |.|_{\mathbb{N}}]$ *$\mathcal{A}_1, \dots, \mathcal{A}_n$ with the same equational theory E such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cup \dots \cup \mathcal{L}(\mathcal{A}_n)$.*

Proof. Without loss of generality, the constraint C can be assumed to be normalized. The subdivision of $\mathcal{A}$ is a collection of normalized conjunctive $\mathtt{TABG}[\approx, \not\approx, \mathbb{N}]$ such that the union of their languages coincides with $\mathcal{L}(\mathcal{A})$.

By iterated application of the Lemma 4.14 to each automaton of the subdivision, combined with the fact that the ordering on measures is well founded, we conclude to the effective existence of normalized conjunctive $\mathtt{TABG}[\approx, \not\approx, \mathbb{N}]$ $\mathcal{A}_1, \dots, \mathcal{A}_n$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cup \dots \cup \mathcal{L}(\mathcal{A}_n)$ and each of them has measure $\langle 0, 0\rangle$. This kind of automata are, in fact, $\mathtt{TABG}^\wedge[\approx, \not\approx, |.|_\mathbb{N}]$, since measure $\langle 0, 0\rangle$ implies that negative literals and literals of type $\|.\|_\mathbb{N}$ do not occur. □

Now, in order to remove all arithmetic constraints, it remains to remove the ones of type $|.|_\mathbb{N}$. This is a rather easy task. For a given $\mathtt{TABG}^\wedge[\approx, \not\approx, |.|_\mathbb{N}]$ $\mathcal{A}$ we create a new $\mathtt{TABG}^\wedge[\approx, \not\approx]$ $\mathcal{A}_{\not\mathbb{N}}$ whose purpose is to simulate the computations of $\mathcal{A}$. To this end, the states of $\mathcal{A}_{\not\mathbb{N}}$ count the number of occurrences of the states of $\mathcal{A}$ in the simulated computation, up to a certain maximum value. This allows $\mathcal{A}_{\not\mathbb{N}}$ to check the constraints of type $|.|_\mathbb{N}$ of $\mathcal{A}$ directly through states. Thus, each state of $\mathcal{A}_{\not\mathbb{N}}$ is of the form q_M for a state q of $\mathcal{A}$ and a mapping $M : Q_\mathcal{A} \to \mathbb{N}$, that is, a mapping counting the number of occurrences of each state.

Definition 4.16. Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ be a normalized $\mathtt{TABG}^\wedge[\approx, \not\approx, |.|_\mathbb{N}]$.

We define $\mathtt{max}_\mathcal{A}$ as one plus the maximum isolated constant occurring in the literals of C of type $|.|_\mathbb{N}$, i.e. one plus the maximum constant k occurring in a literal of C of the form $(a_1|q_1| + \dots + a_n|q_n| \otimes k)$, for $\otimes$ in $\{\geq, \leq, =\}$.

Given two mappings $M_1 : Q \to \{0, \dots, \mathtt{max}_\mathcal{A}\}$ and $M_2 : Q \to \{0, \dots, \mathtt{max}_\mathcal{A}\}$, the *sum* of M_1 and M_2 is defined as the mapping $M_1 + M_2 : Q \to \{0, \dots, \mathtt{max}_\mathcal{A}\}$ satisfying $(M_1 + M_2)(q) = \mathtt{min}(M_1(q) + M_2(q), \mathtt{max}_\mathcal{A})$. Given a state q in Q we define $M_q : Q \to \{0, \dots, \mathtt{max}_\mathcal{A}\}$ as the mapping satisfying $M_q(q) = 1$ and $M_q(q') = 0$ for all $q' \in Q \setminus \{q\}$.

We define $\mathcal{A}_{\not\mathbb{N}}$ as the $\mathtt{TABG}^\wedge[\approx, \not\approx]$ $\langle Q_{\not\mathbb{N}}, \Sigma, F_{\not\mathbb{N}}, \Delta_{\not\mathbb{N}}, E, C_{\not\mathbb{N}}\rangle$, where:

- $Q_{\not\mathbb{N}}$ is $\{q_M \mid q \in Q \wedge M : Q \to \{0, \dots, \mathtt{max}_\mathcal{A}\}\}$.
- $F_{\not\mathbb{N}}$ is $\{q_M \in Q_{\not\mathbb{N}} \mid q \in F \wedge \forall (a_1|q_1| + \dots + a_n|q_n| \otimes k) \in C, \otimes \in \{\geq, \leq, =\} : (a_1 M(q_1) + \dots + a_n M(q_n) \otimes k)\}$.
- $\Delta_{\not\mathbb{N}}$ is $\{f((q_1)_{M_1}, \dots, (q_m)_{M_m}) \xrightarrow{D} q_{M_1 + \dots + M_m + M_q} \mid (f(q_1, \dots, q_m) \xrightarrow{D} q) \in \Delta\}$.
- $C_{\not\mathbb{N}}$ is $\{\bar{q}_{\bar{M}} \approx \tilde{q}_{\tilde{M}} \mid (\bar{q} \approx \tilde{q}) \in C\} \cup \{\bar{q}_{\bar{M}} \not\approx \tilde{q}_{\tilde{M}} \mid (\bar{q} \not\approx \tilde{q}) \in C\}$.

Lemma 4.17. *Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ be a $\mathtt{TABG}^\wedge[\approx, \not\approx, |.|_\mathbb{N}]$.*
Then, $\mathcal{L}(\mathcal{A}_{\not\mathbb{N}}) = \mathcal{L}(\mathcal{A})$.

Proof. The accepting runs of $\mathcal{A}$ can be converted into accepting runs of $\mathcal{A}_{\not\mathbb{N}}$ and vice-versa, following the transformations described below.

- A run $r_{\not\mathbb{N}}$ of $\mathcal{A}_{\not\mathbb{N}}$ can be converted into a run r of $\mathcal{A}$ by replacing each occurrence of a state q_M by the corresponding state q.
- A run r of $\mathcal{A}$ can be converted into a run $r_{\not\mathbb{N}}$ of $\mathcal{A}_{\not\mathbb{N}}$. The transformation can be defined recursively as follows. Let r be a run of the form $(f(q_1, \dots, q_m) \xrightarrow{D} q)(r_1, \dots, r_m)$. Let $(r_1)_{\not\mathbb{N}}, \dots, (r_m)_{\not\mathbb{N}}$ be the transformations of $r_1, \dots, r_m$, and let $(q_1)_{M_1}, \dots, (q_m)_{M_m}$ be the states reached by $(r_1)_{\not\mathbb{N}}, \dots, (r_m)_{\not\mathbb{N}}$, respectively. Then, $r_{\not\mathbb{N}}$ is $(f((q_1)_{M_1}, \dots, (q_m)_{M_m}) \xrightarrow{D} q_{M_1 + \dots + M_m + M_q})((r_1)_{\not\mathbb{N}}, \dots, (r_m)_{\not\mathbb{N}})$.

Each one of the two above transformations is the inverse of the other. Thus, they describe a bijection between runs of $\mathcal{A}$ and runs of $\mathcal{A}_{\not\mathbb{N}}$. Moreover, for each run r of $\mathcal{A}$, the state q_M reached by $r_{\not\mathbb{N}}$ holds that each $q' \in Q$ satisfies $M(q') = \mathtt{min}(|r^{-1}(q')|, \mathtt{max}_\mathcal{A})$ (note that

$r^{-1}(q')$ is the set of positions reaching state q'). Hence, by the definition of $F_{\mathbb{N}}$, it follows that q is in F and r satisfies the arithmetic constraints of C if and only if q_M is in $F_{\mathbb{N}}$. As a consequence, r is accepting if and only if $r_{\mathbb{N}}$ is accepting. Thus, $\mathcal{L}(\mathcal{A}_{\mathbb{N}}) = \mathcal{L}(\mathcal{A})$ holds. □

The following corollary is a consequence of Corollary 4.15 combined with Lemma 4.17.

Corollary 4.18. *Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ be a* TABG$[\approx, \not\approx, \mathbb{N}]$.

Then, one can construct some TABG$^\wedge[\approx, \not\approx]$ $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *with the same equational theory E such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cup \ldots \cup \mathcal{L}(\mathcal{A}_n)$.*

As a final step, we show that TABG$^\wedge[\approx, \not\approx]$ are closed under union for a fixed E.

Lemma 4.19. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be* TABG$^\wedge[\approx, \not\approx]$ *with the same equational theory E. Then, a* TABG$^\wedge[\approx, \not\approx]$ *$\mathcal{A}$ with the same equational theory E can be effectively constructed satisfying $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$.*

Proof. Let $\mathcal{A}_1$ be $\langle Q_1, \Sigma, F_1, \Delta_1, E, C_1\rangle$ and $\mathcal{A}_2$ be $\langle Q_2, \Sigma, F_2, \Delta_2, E, C_2\rangle$. Without loss of generality we can assume that the sets of states Q_1 and Q_2 are disjoint.

In the case where C_1 is just *false* the result follows by defining $\mathcal{A} := \mathcal{A}_2$. Similarly, in the case where C_2 is just *false* the result follows by defining $\mathcal{A} := \mathcal{A}_1$. From now on we assume that these cases do not take place.

We define $\mathcal{A}$ as $\langle Q_1 \uplus Q_2, \Sigma, F_1 \uplus F_2, \Delta_1 \uplus \Delta_2, E, C_1 \wedge C_2\rangle$. Note that $\mathcal{A}$ is a TABG$^\wedge[\approx, \not\approx]$. It is clear that any accepting run of $\mathcal{A}$ is also an accepting run of either $\mathcal{A}_1$ or $\mathcal{A}_2$. Moreover, it can be proved that any accepting run of either $\mathcal{A}_1$ or $\mathcal{A}_2$ is also an accepting run of $\mathcal{A}$. We show this fact only for $\mathcal{A}_1$, since the case for $\mathcal{A}_2$ is analogous.

Let r be an accepting run of $\mathcal{A}_1$. Then, $r \models C_1$ holds. In order to see that it is, in fact, an accepting run of $\mathcal{A}$, it remains to prove $r \models C_2$. Since $\mathcal{A}_2$ is a TABG$^\wedge[\approx, \not\approx]$, C_2 is a conjunction of positive literals of type $\approx$, $\not\approx$ applied to states of Q_2. Therefore, $r \models C_2$ holds, since C_2 is not *false* and any positive literal holds because r uses only states from Q_1. □

Corollary 4.20. *Let $\mathcal{A} = \langle Q, \Sigma, F, \Delta, E, C\rangle$ be a* TABG$[\approx, \not\approx, \mathbb{N}]$.

Then, one can construct a TABG$^\wedge[\approx, \not\approx]$ *$\mathcal{A}'$ with the same equational theory E such that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$.*

Corollary 4.21. *The class of* TABG *languages (modulo the same equational theory) is closed under union.*

In order to complete the closure results for TABG languages under basic set operations, we show that they are also closed under intersection, but not under complementation.

Lemma 4.22. *The class of* TABG *languages (modulo the same equational theory) is closed under intersection.*

Proof. We use a classical Cartesian product of sets of states, with a careful redefinition of constraints on this product.

More precisely, let $\mathcal{A}_1 = \langle Q_1, \Sigma, F_1, \Delta_1, E, C_1\rangle$ and $\mathcal{A}_2 = \langle Q_2, \Sigma, F_2, \Delta_2, E, C_2\rangle$ be two TABG. We construct the TABG $\mathcal{A} = \langle Q_1 \times Q_2, \Sigma, F_1 \times F_2, \Delta, E, C\rangle$ where $\Delta = \big\{ f\big(\langle q_{1,1}, q_{2,1}\rangle, \ldots, \langle q_{1,n}, q_{2,n}\rangle\big) \to \langle q_1, q_2\rangle \bigm| f(q_{i,1}, \ldots, q_{i,n}) \to q_i \in \Delta_i \text{ for } i \in \{1,2\}\big\}$ and the constraint C is obtained from $C_1 \wedge C_2$ by replacing every atom $q_1 \approx q_1'$ with $q_1, q_1' \in Q_1$ (respectively $q_2 \approx q_2'$ with $q_2, q_2' \in Q_2$) by $\bigwedge_{q_2, q_2' \in Q_2} \langle q_1, q_2\rangle \approx \langle q_1', q_2'\rangle$ (respectively $\bigwedge_{q_1, q_1' \in Q_1} \langle q_1, q_2\rangle \approx \langle q_1', q_2'\rangle$), and similarly for the atoms $q_1 \not\approx q_1'$, $q_2 \not\approx q_2'$. With this construction, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$

holds: the left (respectively right) projection of a successful run of $\mathcal{A}$ on a term $t \in \mathcal{T}(\Sigma)$ is a successful run of $\mathcal{A}_1$ (respectively $\mathcal{A}_2$) on t, and the product of two successful runs r_1 of $\mathcal{A}_1$ and r_2 of $\mathcal{A}_2$, both on the same term $t \in \mathcal{T}(\Sigma)$, is a a successful run of $\mathcal{A}$ on t. ☐

Lemma 4.23. *The class of* `TABG` *languages is not closed under complementation.*

Proof. To prove the statement it suffices to define a language L such that L is not recognizable by `TABG` but its complement $\overline{L}$ is. In order to simplify the presentation, we denote terms of the form $f(g^{n_1}(a), f(g^{n_2}(a), \ldots f(g^{n_{k-1}}(a), g^{n_k}(a)) \ldots))$ simply with $[n_1, n_2, \ldots, n_{k-1}, n_k]$. Let L be the language defined as:

$$L = \{[n_1, \ldots, n_k] \mid k, n_1, \ldots, n_k \in \mathbb{N} \wedge \\ \forall i \in \{1, \ldots, k\}\ \exists! j \in \{1, \ldots, k\} \setminus \{i\} : n_i = n_j\}$$

In order to prove that L is not recognizable by `TABG`, by Corollary 4.20, it suffices to prove it for $\texttt{TABG}^\wedge[\approx, \not\approx]$. We proceed by contradiction assuming that there exists a $\texttt{TABG}^\wedge[\approx, \not\approx]$ $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = L$. Let $t \in L$ be the term $[1, \ldots, n, n, \ldots, 1]$, where $n > |Q_\mathcal{A}|$, and let r be an accepting run of $\mathcal{A}$ on t. By the pigeonhole principle, there exist $i, j \in \{1, \ldots, n\}$, with $i < j$, such that the positions $p_i = \overbrace{2.\ldots.2}^{i-1}$ and $p_j = \overbrace{2.\ldots.2}^{j-1}$ satisfy $r(p_i) = r(p_j)$. Let r' be the replacement $r[r|_{p_j}]_{p_i}$. Note that r' is an accepting run of $ta(\mathcal{A})$ on the term $[1, \ldots, i-1, j, \ldots, n, n, \ldots, 1]$, which is not in L. To conclude, it remains to prove that the constraints of $\mathcal{A}$ are satisfied in r'. First, note that this replacement only introduces new subterms at the positions $\hat{P} = \{\hat{p} \in Pos(r) \mid \hat{p} < p_i\}$. Moreover, the rules applied by r' at positions in $\hat{P}$ are the same as in r, and any constraint affecting a position in $\hat{P}$ in r is necessarily a disequality, since $\texttt{term}(r|_{\hat{p}}) \neq_{E_\mathcal{A}} \texttt{term}(r|_{p'})$ holds for $\hat{p} \in \hat{P}$ and $p' \in Pos(r) \setminus \{\hat{p}\}$. By the definition of r', necessarily $\texttt{term}(r'|_{\hat{p}}) \neq_{E_\mathcal{A}} \texttt{term}(r'|_{p'})$ holds also for $\hat{p} \in \hat{P}$ and $p' \in Pos(r') \setminus \{\hat{p}\}$. Therefore, r' satisfies all the constraints, and hence, r' is an accepting run of $\mathcal{A}$, a contradiction.

It remains to prove that $\overline{L}$ can be recognized by a `TABG`. We start by decomposing $\overline{L}$ into simpler languages. First, let L_1 be the language of the malformed terms, i.e. the terms over $\{f : 2, g : 1, a : 0\}$ that are not of the form $[n_1, \ldots, n_k]$. Second, let L_2 be the language of the well-formed terms $[n_1, \ldots, n_k]$ such that for some $i \in \{1, \ldots, k\}$ there exists no $j \in \{1, \ldots, k\} \setminus \{i\}$ satisfying $n_i = n_j$. Third, let L_3 be the language of the well-formed terms $[n_1, \ldots, n_k]$ such that there exist different $i_1, i_2, i_3 \in \{1, \ldots, k\}$ satisfying $n_{i_1} = n_{i_2} = n_{i_3}$. It is easy to see that $\overline{L} = L_1 \cup L_2 \cup L_3$. Moreover, note that L_1 can be recognized by a `TA`, L_2 can be recognized by a $\texttt{TABG}^\wedge[\not\approx, |.|_\mathbb{N}]$ and L_3 can be recognized by a $\texttt{TABG}^\wedge[\approx, |.|_\mathbb{N}]$. By Corollaries 4.20 and 4.21, this concludes the proof. ☐

5. Emptiness Decision Algorithm

In this section we prove the decidability of the *emptiness* problem for $\texttt{TABG}^\wedge$. As a consequence of this result and the results of Section 4, it follows the decidability of emptiness for `TABG`, and even more, of $\texttt{TABG}[\approx, \not\approx, \mathbb{N}]$.

The decidability of emptiness for $\texttt{TABG}^\wedge$ is proved in three steps. In Subsection 5.1, we present a new notion of pumping which allows to transform a run into a smaller run under certain conditions. In Subsection 5.2, we define a well quasi-ordering $\leq$ on a certain set S. In Subsection 5.3, we connect the two previous subsections by describing how to compute,

i	H_i	$\check{H}_i$	$\mathring{\check{H}}_i$
5	$\{\lambda\}$	$\emptyset$	$\emptyset$
4	$\{3\}$	$\{2\}$	$\{1\}$
3	$\{3.3\}$	$\{2, 3.2\}$	$\{1, 3.1\}$
2	$\{3.3.3\}$	$\{2, 3.2, 3.3.2\}$	$\{1, 3.1, 3.3.1\}$
1	$\{2, 3.2, 3.3.2, 3.3.3.2\}$	$\emptyset$	$\{1, 3.1, 3.3.1, 3.3.3.1\}$

Figure 4: H_i, $\check{H}_i$ and $\mathring{\check{H}}_i$ of Example 5.2.

for each run r with height $h = h(r)$, a certain sequence $e_h, \dots, e_1$ of elements of S satisfying the following fact: there exists a pumping on r if and only if $e_i \leq e_j$ for some $h \geq i > j \geq 1$. Moreover, each e_i of the computed sequence is chosen among a finite number of possibilities. Finally, all of these constructions are used as follows. Suppose the existence of an accepting run r. If r is "too high", the fact that $\leq$ is a well quasi-ordering and the properties of the sequence imply the existence of such i, j. Thus, it follows the existence of a pumping providing a smaller accepting run r'. We conclude the existence of a computational bound for the height of a minimum accepting run, and hence, decidability of emptiness.

5.1. **Global Pumpings.**

Pumping is a traditional concept in automata theory, and in particular, it is very useful in order to reason about tree automata. The basic idea is to convert a given run r into another run by replacing a subrun at a certain position p in r by a run r', thus obtaining a run $r[r']_p$. Pumpings are useful for deciding emptiness: if a "big" run can always be reduced by a pumping, then decision of emptiness is obtained by a search of an accepting "small" run.

For plain tree automata, a necessary and sufficient condition to ensure that $r[r']_p$ is a run is that the resulting states of $r|_p$ and r' coincide, since the correct application of a rule at a certain position depends only on the resulting states of the subruns of the direct children. In this case, an accepting run with height bounded by the number of states exists, whenever the accepted language is not empty.

When the tree automaton has equality and disequality constraints, the constraints may be falsified when replacing a subrun by a new run. For $\texttt{TABG}^{\wedge}$, we will define a notion of pumping ensuring that the constraints are satisfied. This notion of pumping requires to perform several replacements in parallel. We first define the sets of positions involved in such kind of pumping.

Definition 5.1. Let $\mathcal{A}$ be a $\texttt{TABG}^{\wedge}$. Let r be a run of $\mathcal{A}$. Let i be an integer between 1 and $h(r)$. We define

H_i as $\{p \in Pos(r) \mid 0 < h(r|_p) = i\}$,
$\check{H}_i$ as $\{p.j \in Pos(r) \mid 0 < h(r|_{p.j}) < i \wedge h(r|_p) > i\}$,
$\mathring{\check{H}}_i$ as $\{p.j \in Pos(r) \mid 0 = h(r|_{p.j}) < i \wedge h(r|_p) > i\}$.

Example 5.2. According to Definition 5.1, for our running example (Example 3.4), we have the H_i, $\check{H}_i$ and $\mathring{\check{H}}_i$ presented in Figure 4.

The following lemma is rather straightforward from the previous definition.

Lemma 5.3. *Let $\mathcal{A}$ be a $\texttt{TABG}^{\wedge}$. Let r be a run of $\mathcal{A}$. Let i be an integer between* 1 *and $h(r)$. Then, any two different positions in $H_i \cup \check{H}_i \cup \mathring{\check{H}}_i$ are parallel, and for any arbitrary*

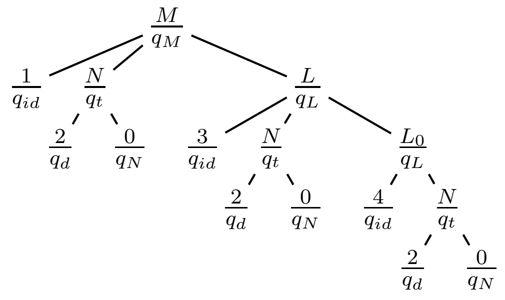

Figure 5: Global pumping of Example 5.5.

position p in $Pos(r)$ there is a position $\bar{p}$ in $H_i \cup \check{H}_i \cup \mathring{H}_i$ such that, either p is a prefix of $\bar{p}$, or $\bar{p}$ is a prefix of p.

Proof. For the first fact, note that any proper prefix p of a position $\bar{p}$ in $H_i \cup \check{H}_i \cup \mathring{H}_i$ satisfies $h(r|_p) > i$. Thus, such a p is not in $H_i \cup \check{H}_i \cup \mathring{H}_i$. For the second fact, consider any p in $Pos(r)$. If $h(r|_p) \leq i$ holds, then the smallest position $\bar{p}$ satisfying $\bar{p} \leq p$ and $h(r|_{\bar{p}}) \leq i$ is in $H_i \cup \check{H}_i \cup \mathring{H}_i$, and we are done. Otherwise, if $h(r|_p) > i$ holds, then the smallest position $\bar{p}$ of the form $p.1.\ldots.1$ and satisfying $h(r|_{\bar{p}}) \leq i$ is in $H_i \cup \check{H}_i \cup \mathring{H}_i$, and we are done. □

Definition 5.4. Let $\mathcal{A}$ be a $\texttt{TABG}^\wedge$. Let E be $E_{\mathcal{A}}$. Let r be a run of $\mathcal{A}$. Let i, j be integers satisfying $1 \leq j < i \leq h(r)$. A *pump-injection* $I : (H_i \cup \check{H}_i \cup \mathring{H}_i) \to (H_j \cup \check{H}_j \cup \mathring{H}_j)$ is an injective function such that the following conditions hold:

(C_1) $I(H_i) \subseteq H_j$, $I(\check{H}_i) \subseteq \check{H}_j$ and $I(\mathring{H}_i) \subseteq \mathring{H}_j$. Moreover, I restricted to $\mathring{H}_i$ is the identity, i.e. $I(p) = p$ for each p in $\mathring{H}_i$.

(C_2) For each $\bar{p}$ in $H_i \cup \check{H}_i \cup \mathring{H}_i$, $r(\bar{p}) = r(I(\bar{p}))$.

(C_3) For each $\bar{p}_1, \bar{p}_2$ in $H_i \cup \check{H}_i \cup \mathring{H}_i$, $(\texttt{term}(r|_{\bar{p}_1}) =_E \texttt{term}(r|_{\bar{p}_2})) \Leftrightarrow (\texttt{term}(r|_{I(\bar{p}_1)}) =_E \texttt{term}(r|_{I(\bar{p}_2)}))$.

Let $\{\bar{p}_1, \ldots, \bar{p}_n\}$ be $H_i \cup \check{H}_i \cup \mathring{H}_i$ more explicitly written. The run $r[r|_{I(\bar{p}_1)}]_{\bar{p}_1} \ldots [r|_{I(\bar{p}_n)}]_{\bar{p}_n}$ is called a *global pumping* on r with indexes i, j, and injection I.

By Condition C_2, $r[r|_{I(\bar{p}_1)}]_{\bar{p}_1} \ldots [r|_{I(\bar{p}_n)}]_{\bar{p}_n}$ is a run of $ta(\mathcal{A})$, but it is still necessary to prove that it is a run of $\mathcal{A}$. By abuse of notation, when we write $r[r|_{I(\bar{p}_1)}]_{\bar{p}_1} \ldots [r|_{I(\bar{p}_n)}]_{\bar{p}_n}$, we sometimes consider that I and $\{\bar{p}_1, \ldots, \bar{p}_n\}$ are still explicit, and say that it is a global pumping with some indexes $1 \leq j < i \leq h(r)$.

Example 5.5. Following our running example, we define a pump-injection $I : (H_4 \cup \check{H}_4 \cup \mathring{H}_4) \to (H_3 \cup \check{H}_3 \cup \mathring{H}_3)$ as follows: $I(1) = 1$, $I(2) = 2$, $I(3) = 3.3$. We note that I is a correct pump-injection: $I(H_4) \subseteq H_3$, $I(\check{H}_4) \subseteq \check{H}_3$ and $I(\mathring{H}_4) \subseteq \mathring{H}_3$ hold, and I restricted to $\mathring{H}_4$ is, in fact, the identity, thus (C_1) holds. For (C_2), we have $r(1) = r(I(1)) = q_{id}$, $r(2) = r(I(2)) = q_t$, and $r(3) = r(I(3)) = q_L$. Regarding (C_3), for each different $\bar{p}_1, \bar{p}_2$ in $H_4 \cup \check{H}_4 \cup \mathring{H}_4$, $\texttt{term}(r|_{\bar{p}_1}) \neq \texttt{term}(r|_{\bar{p}_2})$ and $\texttt{term}(r|_{I(\bar{p}_1)}) \neq \texttt{term}(r|_{I(\bar{p}_2)})$ hold. After applying the pump-injection I, we obtain the term and run r' of Figure 5.

Our goal is to prove that any global pumping $r[r|_{I(\bar{p}_1)}]_{\bar{p}_1} \ldots [r|_{I(\bar{p}_n)}]_{\bar{p}_n}$ is a run, and in particular, that all equality and disequality constraints are satisfied. To this end we first state the following intermediate statement, which determines the height of the terms pending at some positions after the pumping. It can be easily proved by induction on the height of the involved term.

Lemma 5.6. *Let $\mathcal{A}$ be a* $\texttt{TABG}^{\wedge}$*. Let r be a run of $\mathcal{A}$. Let r' be the global pumping $r[r|_{I(\bar{p}_1)}]_{\bar{p}_1} \dots [r|_{I(\bar{p}_n)}]_{\bar{p}_n}$ on r with indexes $1 \leq j < i \leq h(r)$ and injection I. Let $k \geq 0$ be a natural number and let p be a position of r such that $h(r|_p)$ is $i+k$.*

Then, p is also a position of r' and $h(r'|_p)$ is $j+k$.

Proof. Position p is obviously a position of r' since no position in $H_i \cup \check{H}_i \cup \mathring{H}_i$ is a proper prefix of p. We prove the second part of the statement by induction on k. First, assume $k = 0$. Then, $h(r|_p)$ is i. Thus, p is in H_i, say p is $\bar{p}_1$. Therefore, $r'|_p$ is $r|_{I(\bar{p}_1)}$. By Condition (C_1) of the definition of pump-injection, $I(\bar{p}_1) \in H_j$ holds. Hence, $h(r'|_p) = h(r|_{I(\bar{p}_1)}) = j$.

Now, assume $k > 0$. Let m be the arity of $\texttt{symbol}(r|_p)$. Thus, $p.1, \dots, p.m$ are all the child positions of p in r. Since $h(r|_p)$ is $i+k$, all $h(r|_{p.1}), \dots, h(r|_{p.m})$ are smaller than or equal to $i+k-1$, and at least one of them is equal to $i+k-1$.

Consider any α in $\{1, \dots, m\}$. If $h(r|_{p.\alpha})$ is $i+k'$ for some $0 \leq k' \leq k-1$, then, by induction hypothesis, $h(r'|_{p.\alpha})$ is $j+k'$. Otherwise, if $h(r|_{p.\alpha})$ is strictly smaller than i, then $p.\alpha$ is one of the positions in $\check{H}_i \cup \mathring{H}_i$, say $\bar{p}_1$. In this case, $r'|_{\bar{p}_1}$ is $r|_{I(\bar{p}_1)}$, and by Condition (C_1) of the definition of I, $I(\bar{p}_1)$ belongs to $\check{H}_j \cup \mathring{H}_j$. Therefore, $h(r|_{I(\bar{p}_1)}) < j$ holds, and hence, $h(r'|_{p.\alpha}) = h(r'|_{\bar{p}_1}) = h(r|_{I(\bar{p}_1)}) < j \leq j+k-1$ holds.

From the above cases we conclude that, if $h(r|_{p.\alpha})$ is $i+k-1$, then $h(r'|_{p.\alpha})$ is $j+k-1$, and if $h(r|_{p.\alpha})$ is smaller than $i+k-1$, then $h(r'|_{p.\alpha})$ is smaller than $j+k-1$. It follows that all $h(r'|_{p.1}), \dots, h(r'|_{p.m})$ are smaller than or equal to $j+k-1$, and at least one of them is equal to $j+k-1$. As a consequence, $h(r'|_p)$ is $j+k$. □

Corollary 5.7. *Let $\mathcal{A}$ be a* $\texttt{TABG}^{\wedge}$*. Let r be a run of $\mathcal{A}$. Let r' be a global pumping on r. Then, $h(r') < h(r)$.*

The following lemma states that equality and disequality relations are preserved, not only for terms pending at the positions of the domain of I, but also for terms pending at prefixes of positions of such domain. Again, it is rather easy to prove by induction on the height of the involved terms.

Lemma 5.8. *Let $\mathcal{A}$ be a* $\texttt{TABG}^{\wedge}$*. Let r be a run of $\mathcal{A}$. Let r' be the global pumping $r[r|_{I(\bar{p}_1)}]_{\bar{p}_1} \dots [r|_{I(\bar{p}_n)}]_{\bar{p}_n}$ with indexes $1 \leq j < i \leq h(r)$ and injection I. Let p_1, p_2 be positions of r satisfying that each of them is a prefix of a position in $H_i \cup \check{H}_i \cup \mathring{H}_i$.*

Then, p_1, p_2 are positions of r' and $(\texttt{term}(r|_{p_1}) =_E \texttt{term}(r|_{p_2})) \Leftrightarrow (\texttt{term}(r'|_{p_1}) =_E \texttt{term}(r'|_{p_2}))$ holds.

Proof. The first statement follows by Lemma 5.6. We prove the second part of the statement by induction on $h(r|_{p_1}) + h(r|_{p_2})$. We distinguish the following cases:

- Assume that both p_1 and p_2 are positions in $H_i \cup \check{H}_i \cup \mathring{H}_i$, say $\bar{p}_1$ and $\bar{p}_2$, respectively. Therefore, $r'|_{p_1}$ is $r|_{I(\bar{p}_1)}$ and $r'|_{p_2}$ is $r|_{I(\bar{p}_2)}$. By Condition (C_3) of the definition of pump-injection, $(\texttt{term}(r|_{\bar{p}_1}) =_E \texttt{term}(r|_{\bar{p}_2})) \Leftrightarrow (\texttt{term}(r|_{I(\bar{p}_1)}) =_E \texttt{term}(r|_{I(\bar{p}_2)}))$ holds. Thus, $(\texttt{term}(r|_{p_1}) =_E \texttt{term}(r|_{p_2})) \Leftrightarrow (\texttt{term}(r'|_{p_1}) =_E \texttt{term}(r'|_{p_2}))$ holds, and we are done.
- Assume that one of p_1 or p_2, say p_1, is a proper prefix of a position in $H_i \cup \check{H}_i \cup \mathring{H}_i$, and p_2 is a position in $H_i \cup \check{H}_i \cup \mathring{H}_i$. Then, $h(r|_{p_1}) = i+k$ for some $k > 0$, and $h(r|_{p_2}) \leq i$ holds. Thus, $(\texttt{term}(r|_{p_1}) \neq_E \texttt{term}(r|_{p_2}))$ holds. By Lemma 5.6, $h(r'|_{p_1}) = j+k$. By the definition of pump-injection, $h(r'|_{p_2}) \leq j$. Thus, also $(\texttt{term}(r'|_{p_1}) \neq_E \texttt{term}(r'|_{p_2}))$ holds, and we are done.

- Assume that both p_1 and p_2 are proper prefixes of positions in $H_i \cup \check{H}_i \cup \mathring{H}_i$. Note that, in this case, $\mathtt{symbol}(r'|_{p_1}) = \mathtt{symbol}(r|_{p_1})$ and $\mathtt{symbol}(r'|_{p_2}) = \mathtt{symbol}(r|_{p_2})$ hold. Let $\mathtt{symbol}(r|_{p_1})$ and $\mathtt{symbol}(r|_{p_2})$ be f and g, with arities n and m, respectively. Recall that I is the identity for the positions in $\mathring{H}_i$, and hence, a position α in $\{1, \ldots, n\}$ satisfies $\mathtt{symbol}(r|_{p_1.\alpha}) \in \Sigma_0 \Leftrightarrow \mathtt{symbol}(r'|_{p_1.\alpha}) \in \Sigma_0$, and $\mathtt{symbol}(r|_{p_1.\alpha}), \mathtt{symbol}(r'|_{p_1.\alpha}) \in \Sigma_0 \Rightarrow \mathtt{symbol}(r|_{p_1.\alpha}) = \mathtt{symbol}(r'|_{p_1.\alpha})$. Similarly, a position β in $\{1, \ldots, m\}$ satisfies $\mathtt{symbol}(r|_{p_2.\beta}) \in \Sigma_0 \Leftrightarrow \mathtt{symbol}(r'|_{p_2.\beta}) \in \Sigma_0$, and $\mathtt{symbol}(r|_{p_2.\beta}), \mathtt{symbol}(r'|_{p_2.\beta}) \in \Sigma_0 \Rightarrow \mathtt{symbol}(r|_{p_2.\beta}) = \mathtt{symbol}(r'|_{p_2.\beta})$. Moreover, since such positions $p_1.\alpha$ and $p_2.\beta$ are prefixes of positions in $H_i \cup \check{H}_i \cup \mathring{H}_i$, by induction hypothesis, $(\mathtt{term}(r|_{p_1.\alpha}) =_E \mathtt{term}(r|_{p_2.\beta})) \Leftrightarrow (\mathtt{term}(r'|_{p_1.\alpha}) =_E \mathtt{term}(r'|_{p_2.\beta}))$ for all such α in $\{1, \ldots, n\}$ and β in $\{1, \ldots, m\}$. By Lemma 2.1, $(\mathtt{term}(r|_{p_1}) =_E \mathtt{term}(r|_{p_2})) \Leftrightarrow (\mathtt{term}(r'|_{p_1}) =_E \mathtt{term}(r'|_{p_2}))$ follows, and we are done. □

Now we prove that the result of a global pumping preserves the satisfaction of the global constraints.

Lemma 5.9. *Let $\mathcal{A}$ be a $\mathtt{TABG}^{\wedge}$. Let r be a run of $\mathcal{A}$. Let r' be the global pumping $r[r|_{I(\bar{p}_1)}]_{\bar{p}_1} \ldots [r|_{I(\bar{p}_n)}]_{\bar{p}_n}$ with indexes $1 \leq j < i \leq h(r)$ and injection I.*

Then, r' satisfies all global constraints of $\mathcal{A}$.

Proof. Let us consider two different positions p_1, p_2 of $Pos(r')$ involved in the constraint $C_{\mathcal{A}}$, i.e. either $r'(p_1) \approx r'(p_2)$ or $r'(p_1) \not\approx r'(p_2)$ occurs in $C_{\mathcal{A}}$. According to Lemma 5.3, we can distinguish the following cases:

- Suppose that a position in $H_i \cup \check{H}_i \cup \mathring{H}_i$, say $\bar{p}_1$, is a prefix of both p_1, p_2. Then, $r'|_{p_1} = r|_{I(\bar{p}_1).(p_1 - \bar{p}_1)}$ and $r'|_{p_2} = r|_{I(\bar{p}_1).(p_2 - \bar{p}_1)}$ hold. Hence, $r'|_{p_1}$ and $r'|_{p_2}$ are also subruns of r occurring at different positions. Thus, since r is a run, they satisfy the atom involving $r'(p_1)$ and $r'(p_2)$.
- Suppose that two different positions in $H_i \cup \check{H}_i \cup \mathring{H}_i$, say $\bar{p}_1$ and $\bar{p}_2$, are prefixes of p_1 and p_2, respectively. Then, $r'|_{p_1} = r|_{I(\bar{p}_1).(p_1 - \bar{p}_1)}$ and $r'|_{p_2} = r|_{I(\bar{p}_2).(p_2 - \bar{p}_2)}$ hold. By the injectivity of I, $I(\bar{p}_1) \neq I(\bar{p}_2)$ holds. Moreover, by Lemma 5.3, $I(\bar{p}_1) \parallel I(\bar{p}_2)$ holds. Hence, as before, $r'|_{p_1}$ and $r'|_{p_2}$ are subruns of r occurring at different (in fact, parallel) positions. Thus, they satisfy the atom involving $r'(p_1)$ and $r'(p_2)$.
- Suppose that one of p_1, p_2, say p_1, is a proper prefix of a position in $H_i \cup \check{H}_i \cup \mathring{H}_i$, and that p_2 satisfies that some position in $H_i \cup \check{H}_i \cup \mathring{H}_i$ is a prefix of p_2. It follows that $h(r'|_{p_2})$ is smaller than or equal to j, and $r'|_{p_2}$ is also a subrun of r. Moreover, p_1 is also a position of r, $r'(p_1) = r(p_1)$ holds, and $h(r|_{p_1}) = i + k$ holds for some $k > 0$. Hence, $\mathtt{term}(r|_{p_1}) \neq_E \mathtt{term}(r'|_{p_2})$ holds. Since r is a run and $r'|_{p_2}$ is a subrun of r, the atom involving $r(p_1)$ and $r'(p_2)$ is necessarily of the form $r(p_1) \not\approx r'(p_2)$. Thus, the atom involving $r'(p_1)$ and $r'(p_2)$ is necessarily of the form $r'(p_1) \not\approx r'(p_2)$. By Lemma 5.6, $h(r'|_{p_1})$ is $j + k$. Therefore, also $\mathtt{term}(r'|_{p_1}) \neq_E \mathtt{term}(r'|_{p_2})$ holds, and hence, such an atom is satisfied for such positions in r'.
- Suppose that both p_1, p_2 are proper prefixes of positions in $H_i \cup \check{H}_i \cup \mathring{H}_i$. Then, p_1, p_2 are positions of r satisfying $h(r|_{p_1}), h(r|_{p_2}) \geq i$. Moreover, $r(p_1) = r'(p_1)$ and $r(p_2) = r'(p_2)$ hold. Since r is a run, the atom involving $r(p_1)$ and $r(p_2)$ is satisfied in the run r for positions p_1 and p_2. By Lemma 5.8, $(\mathtt{term}(r|_{p_1}) =_E \mathtt{term}(r|_{p_2})) \Leftrightarrow (\mathtt{term}(r'|_{p_1}) =_E \mathtt{term}(r'|_{p_2}))$ holds. Thus, the atom involving $r'(p_1)$ and $r'(p_2)$ is satisfied in the run r' for positions p_1 and p_2. □

Finally, we prove that the result of a global pumping preserves the satisfaction of the constraints between brothers.

Lemma 5.10. *Let $\mathcal{A}$ be a* $\texttt{TABG}^{\wedge}$. *Let r be a run of $\mathcal{A}$. Let r' be the global pumping $r[r|_{I(\bar{p}_1)}]_{\bar{p}_1} \dots [r|_{I(\bar{p}_n)}]_{\bar{p}_n}$ with indexes $1 \leq j < i \leq h(r)$ and injection I.*

Then, r' satisfies all constraints between brothers of $\mathcal{A}$.

Proof. Let us consider a position p of $Pos(r')$ and two positions i_1, i_2 involved in a constraint of the rule used at position p in r', i.e. either $\gamma = (i_1 \approx i_2)$ or $\gamma = (i_1 \not\approx i_2)$ occur in this constraint. According to Lemma 5.3, we can distinguish the following cases:

- Suppose that a position in $H_i \cup \check{H}_i \cup \mathring{H}_i$, is a prefix of p. Then, $r'|_p$ is also a subrun of r. Thus, since r is a run, the constraint is satisfied.
- Suppose that p is a proper prefix of a position in $H_i \cup \check{H}_i \cup \mathring{H}_i$. Then, $p.i_1$ and $p.i_2$ are prefixes of positions in $H_i \cup \check{H}_i \cup \mathring{H}_i$. By Lemma 5.8, $(\texttt{term}(r|_{p.i_1}) =_E \texttt{term}(r|_{p.i_2})) \Leftrightarrow (\texttt{term}(r'|_{p.i_1}) =_E \texttt{term}(r'|_{p.i_2}))$ holds. Since r is a run, it follows that $(\texttt{term}(r|_{p.i_1}) =_E \texttt{term}(r|_{p.i_2})) \Leftrightarrow \gamma = (i_1 \approx i_2)$. Thus, $(\texttt{term}(r'|_{p.i_1}) =_E \texttt{term}(r'|_{p.i_2})) \Leftrightarrow \gamma = (i_1 \approx i_2)$ holds. Thus, the atom involving i_1 and i_2 is satisfied in the run r' for position p. $\square$

As a consequence of the previous lemmas, we have that the result of a global pumping satisfies all constraints.

Corollary 5.11. *Let $\mathcal{A}$ be a* $\texttt{TABG}^{\wedge}$. *Let r be a run of $\mathcal{A}$. Let r' be the global pumping $r[r|_{I(\bar{p}_1)}]_{\bar{p}_1} \dots [r|_{I(\bar{p}_n)}]_{\bar{p}_n}$ with indexes $1 \leq j < i \leq h(r)$ and injection I.*

Then, r' is a run of $\mathcal{A}$.

5.2. **A well quasi-ordering.**

In this subsection we define a well quasi-ordering. It assures the existence of a computational bound for certain sequences of elements of the corresponding well quasi-ordered set. It will be connected with global pumpings in the next subsection.

Definition 5.12. Let $\leq$ denote the usual quasi-ordering on natural numbers. Let n be a natural number.

We define the extension of $\leq$ to n-tuples of natural numbers as $\langle x_1, \dots, x_n \rangle \leq \langle y_1, \dots, y_n \rangle$ if $x_i \leq y_i$ for each i in $\{1, \dots, n\}$. We define $\texttt{sum}(\langle x_1, \dots, x_n \rangle) := x_1 + \cdots + x_n$.

We define the extension of $\leq$ to multisets of n-tuples of natural numbers as $[e_1, \dots, e_\alpha] \leq [e'_1, \dots, e'_\beta]$ if there is an injection $I : \{1, \dots, \alpha\} \to \{1, \dots, \beta\}$ satisfying $e_i \leq e'_{I(i)}$ for each i in $\{1, \dots, \alpha\}$. We define $\texttt{sum}([e_1, \dots, e_\alpha]) := \texttt{sum}(e_1) + \cdots + \texttt{sum}(e_\alpha)$.

We define the extension of $\leq$ to pairs of multisets of n-tuples of natural numbers as $\langle P_1, \check{P}_1 \rangle \leq \langle P_2, \check{P}_2 \rangle$ if $P_1 \leq P_2$ and $\check{P}_1 \leq \check{P}_2$.

As a direct consequence of Higman's Lemma [Gal91] we have the following:

Lemma 5.13. *Given n, $\leq$ is a well quasi-ordering for pairs of multisets of n-tuples of natural numbers.*

In any infinite sequence $e_1, e_2, \dots$ of elements from a well quasi-ordered set there always exist two indexes $i < j$ satisfying $e_i \leq e_j$. In general, this fact does not imply the existence of a bound for the length of sequences without such indexes. For example, the relation $\leq$ between natural numbers is a well quasi-ordering, but there may exist arbitrarily long sequences $x_1, \dots, x_k$ of natural numbers such that $x_i > x_j$ for all $1 \leq i < j \leq k$. In order to bound the length of such sequences, it is sufficient to force that the first element and each next element of the sequence are chosen among a finite number of possibilities. Indeed in this this case, by König's lemma, the prefix trees describing all such (finite) sequences is finite. As a particular case of this fact we have the following result (the proof is standard, but we include it for completeness).

Lemma 5.14. *There exists a computable function* $B : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *such that, given two natural numbers* a, n, $B(a,n)$ *is a bound for the length* ℓ *of any sequence* $\langle T_1, \check{T}_1\rangle, \ldots, \langle T_\ell, \check{T}_\ell\rangle$ *of pairs of multisets of* n*-tuples of natural numbers such that the following conditions hold:*

(1) *The tuple* $\langle 0, \ldots, 0\rangle$ *does not occur in any* T_i, $\check{T}_i$ *for* i *in* $\{1, \ldots, \ell\}$*.*
(2) $\mathtt{sum}(T_1) = 1$ *and* $\mathtt{sum}(\check{T}_1) = 0$*.*
(3) *For each* i *in* $\{1, \ldots, \ell - 1\}$, $a \cdot \mathtt{sum}(T_i) + \mathtt{sum}(\check{T}_i) \geq \mathtt{sum}(T_{i+1}) + \mathtt{sum}(\check{T}_{i+1})$*.*
(4) *There are no* i, j *satisfying* $1 \leq i < j \leq \ell$ *and* $\langle T_i, \check{T}_i\rangle \leq \langle T_j, \check{T}_j\rangle$*.*

Proof. For proving the statement, we first construct a rooted tree $S = (V, E)$ labelled by sequences of pairs of multisets of n-tuples, where the depth of each node is equal to the length of the sequence labeling it and such that the set of internal nodes of S corresponds exactly to the set of sequences satisfying conditions (1) to (4). Second, we show that S is finite. This concludes the proof, since finiteness of S and its constructive definition imply that S is computable, and $B(a,n)$ can be defined as the maximal depth of S.

We define V as the set of all the sequences $\langle T_1, \check{T}_1\rangle, \ldots, \langle T_\ell, \check{T}_\ell\rangle$ of pairs of multisets of n-tuples satisfying the conditions (1) to (3) and such that there are no i, j satisfying $1 \leq i < j < \ell$ and $\langle T_i, \check{T}_i\rangle \leq \langle T_j, \check{T}_j\rangle$. This last condition, that we will refer to as (5), is weaker than (4) since in (5) we have $j < \ell$ instead of $j \leq \ell$. Thus, all sequences satisfying conditions (1) to (4) belong to V. Note that V contains the empty sequence, which we denote as ε. We define $E \subseteq V^2$ as the set of edges containing $\langle T_1, \check{T}_1\rangle, \ldots, \langle T_i, \check{T}_i\rangle \longrightarrow \langle T_1, \check{T}_1\rangle, \ldots, \langle T_i, \check{T}_i\rangle, \langle T_{i+1}, \check{T}_{i+1}\rangle$ for every such couple of sequences in V.

It is quite obvious that $S = (V, E)$ is a tree rooted at ε, since ε does not have an input edge, each sequence of length 1 has a unique input edge coming from ε, and each sequence of length $i > 1$ has a unique input edge coming from its unique prefix sequence of length $i-1$. Also, the set of internal nodes of S is exactly the set of sequences satisfying conditions (1) to (4), and the set of leaves of S is exactly the set of sequences satisfying conditions (1) to (3), and (5), but not (4).

It remains to show that S is finite. To this end, it suffices to see that S is finitely branching and that there is no path with infinite length.

First, we prove that each node $v \in V$ has a finite branching: ε links to all the sequences of length 1, the number of which is bounded by conditions (1) and (2); and each sequence $\langle T_1, \check{T}_1\rangle, \ldots, \langle T_i, \check{T}_i\rangle$ can only link to sequences of the form $\langle T_1, \check{T}_1\rangle, \ldots, \langle T_i, \check{T}_i\rangle, \langle T_{i+1}, \check{T}_{i+1}\rangle$, the number of which is bounded by conditions (1) and (3).

Second, we prove that there is no path with infinite length in S in a standard way. We proceed by contradiction by assuming that we have an infinite path $v_0, v_1, v_2, v_3, \ldots$ By construction, we have $v_0 = \varepsilon$, and for all $i \geq 1$ and all $j \geq i$, the prefix of length i of the sequence v_j is equal to v_i. Consider the infinite sequence $\langle T_1, \check{T}_1\rangle, \langle T_2, \check{T}_2\rangle, \ldots$ where for all $i \geq 1$, $\langle T_i, \check{T}_i\rangle$ is the last element of the sequence v_i. Since $\leq$ on pairs of multisets of n-tuples is a well quasi-ordering, there exist two indexes i, j satisfying $i < j$ and $\langle T_i, \check{T}_i\rangle \leq \langle T_j, \check{T}_j\rangle$. Hence, all sequences v_k for $k > j$ do not satisfy condition (5), and hence they do not belong to V, contradicting the infiniteness of the path. □

i	r_{H_i}	$r_{\check{H}_i}$	$r_{\mathring{H}_i}$
5	$[\langle 0,0,0,0,0,1\rangle]$	$[\,]$	$[\,]$
4	$[\langle 0,0,0,0,1,0\rangle]$	$[\langle 0,0,0,1,0,0\rangle]$	$[\langle 0,0,1,0,0,0\rangle]$
3	$[\langle 0,0,0,0,1,0\rangle]$	$[\langle 0,0,0,2,0,0\rangle]$	$[\langle 0,0,1,0,0,0\rangle,\langle 0,0,1,0,0,0\rangle]$
2	$[\langle 0,0,0,0,1,0\rangle]$	$[\langle 0,0,0,3,0,0\rangle]$	$[\langle 0,0,1,0,0,0\rangle,\langle 0,0,1,0,0,0\rangle,$ $\langle 0,0,1,0,0,0\rangle]$
1	$[\langle 0,0,0,4,0,0\rangle]$	$[\,]$	$[\langle 0,0,1,0,0,0\rangle,\langle 0,0,1,0,0,0\rangle,$ $\langle 0,0,1,0,0,0\rangle,\langle 0,0,1,0,0,0\rangle]$

Figure 6: Multisets r_{H_i}, $r_{\check{H}_i}$ and $r_{\mathring{H}_i}$ of Example 5.17.

In order to bound the height of a term accepted by a given $\texttt{TABG}^{\wedge}$ $\mathcal{A}$ (and of minimum height), Lemma 5.14 will be used by making a to be the maximum arity of the signature of $\mathcal{A}$, and making n to be the number of states of $\mathcal{A}$.

5.3. **Mapping a run to a sequence of the well quasi-ordered set.** We will associate, to each number i in $\{1,\ldots,h(r)\}$, a pair of multisets of n-tuples of natural numbers, which can be compared with other pairs according to the definition of $\leq$ in the previous subsection. To this end, we first associate n-tuples to terms and multisets of n-tuples to sets of positions.

Definition 5.15. Let $\mathcal{A}$ be a $\texttt{TABG}^{\wedge}$. Let E be $E_{\mathcal{A}}$. Let $q_1,\ldots,q_n$ be the states of $\mathcal{A}$. Let r be a run of $\mathcal{A}$. Let P be a set of positions of r. Let t be a term. We define $r_{t,P}$ as the following tuple of natural numbers: $\Big\langle \big|\{p \in P \mid \texttt{term}(r|_p) =_E t \wedge r(p) = q_1\}\big|, \ldots, \big|\{p \in P \mid \texttt{term}(r|_p) =_E t \wedge r(p) = q_n\}\big|\Big\rangle$

Definition 5.16. Let $\mathcal{A}$ be a $\texttt{TABG}^{\wedge}$. Let E be $E_{\mathcal{A}}$. Let r be a run of $\mathcal{A}$. Let P be a set of positions of r. Let $\{[t_1],\ldots,[t_k]\}$ be the set of equivalence classes modulo E of the set of terms $\{\texttt{term}(r|_p) \mid p \in P\}$ with representatives $t_1,\ldots,t_k$. We define r_P as the multiset $[r_{t_1,P},\ldots,r_{t_k,P}]$.

Example 5.17. Following our running example, for the representation of the n-tuples of natural numbers we order the states as $\langle q_d, q_N, q_{id}, q_t, q_L, q_M\rangle$. The multisets r_{H_i}, $r_{\check{H}_i}$ and $r_{\mathring{H}_i}$ are presented in Figure 6.

The following lemma connects the existence of a pump-injection with the quasi-ordering relation.

Lemma 5.18. *Let $\mathcal{A}$ be a $\texttt{TABG}^{\wedge}$. Let r be a run of $\mathcal{A}$. Let i,j be integers satisfying $1 \leq j < i \leq h(r)$.*

Then, there exists a pump-injection $I : (H_i \cup \check{H}_i \cup \mathring{H}_i) \to (H_j \cup \check{H}_j \cup \mathring{H}_j)$ if and only if $\langle r_{H_i}, r_{\check{H}_i}\rangle \leq \langle r_{H_j}, r_{\check{H}_j}\rangle$.

Proof. Although we prove both directions of the double implication, the left-to-right one is technical but not conceptually difficult, and it is not necessary for the rest of the paper. In the following, we write E for $E_{\mathcal{A}}$.
$\Rightarrow$) Assume that there exists a pump-injection $I : (H_i \cup \check{H}_i \cup \mathring{H}_i) \to (H_j \cup \check{H}_j \cup \mathring{H}_j)$. We just prove $r_{H_i} \leq r_{H_j}$, since $r_{\check{H}_i} \leq r_{\check{H}_j}$ can be proved analogously. By Condition (C_1) of the definition of pump-injection, $I(H_i) \subseteq H_j$ holds. We write the equivalence classes of

$\{\mathtt{term}(r|_p) \mid p \in H_i\}$ and $\{\mathtt{term}(r|_p) \mid p \in H_j\}$ modulo E more explicitly as $\{[t_{i,1}], \ldots, [t_{i,\alpha}]\}$ and $\{[t_{j,1}], \ldots, [t_{j,\beta}]\}$, respectively. Hence, it remains to prove that $[r_{t_{i,1},H_i}, \ldots, r_{t_{i,\alpha},H_i}] \leq [r_{t_{j,1},H_j}, \ldots, r_{t_{j,\beta},H_j}]$. To this end we define the function $I' : \{1, \ldots, \alpha\} \to \{1, \ldots, \beta\}$ as follows. For each γ in $\{1, \ldots, \alpha\}$, we choose a position p in H_i satisfying $\mathtt{term}(r|_p) =_E t_{i,\gamma}$, determine the index δ of the term $t_{j,\delta}$ satisfying $t_{j,\delta} =_E \mathtt{term}(r|_{I(p)})$, and define $I'(\gamma) := \delta$. This function I' is injective due to Condition (C_3) of the definition of pump-injection. In order to conclude, it suffices to prove $r_{t_{i,\gamma},H_i} \leq r_{t_{j,I'(\gamma)},H_j}$ for each γ in $\{1, \ldots, \alpha\}$. We just prove it for $\gamma = 1$. For proving $r_{t_{i,1},H_i} \leq r_{t_{j,I'(1)},H_j}$ it suffices to prove the following statement for each state q of $\mathcal{A}$: $\big|\{p \in H_i \mid \mathtt{term}(r|_p) =_E t_{i,1} \wedge r(p) = q\}\big| \leq \big|\{p \in H_j \mid \mathtt{term}(r|_p) =_E t_{j,I'(1)} \wedge r(p) = q\}\big|$.

To this end, since I is injective, it suffices to prove that $I(\{p \in H_i \mid \mathtt{term}(r|_p) =_E t_{i,1} \wedge r(p) = q\})$ is included in $\{p \in H_j \mid \mathtt{term}(r|_p) =_E t_{j,I'(1)} \wedge r(p) = q\}$ for each state q of $\mathcal{A}$. Thus, consider any $\bar{p}$ of $\{p \in H_i \mid \mathtt{term}(r|_p) =_E t_{i,1} \wedge r(p) = q\}$. Let p' be the chosen position for defining $I'(1)$. In particular, $\mathtt{term}(r|_{p'}) =_E t_{i,1}$ and $\mathtt{term}(r|_{I(p')}) =_E t_{j,I'(1)}$ hold. Note that $\mathtt{term}(r|_{\bar{p}}) =_E \mathtt{term}(r|_{p'}) =_E t_{i,1}$ holds. Thus, by Condition (C_3) of the definition of pump-injection, $\mathtt{term}(r|_{I(\bar{p})}) =_E \mathtt{term}(r|_{I(p')})$ holds. Therefore, $\mathtt{term}(r|_{I(\bar{p})}) =_E t_{j,I'(1)}$ holds. In order to show the inclusion $I(\bar{p}) \in \{p \in H_j \mid \mathtt{term}(r|_p) =_E t_{j,I'(1)} \wedge r(p) = q\}$ it remains to see $r(I(\bar{p})) = q$. Note that, since $\bar{p}$ belongs to $\{p \in H_i \mid \mathtt{term}(r|_p) =_E t_{i,1} \wedge r(p) = q\}$, $r(\bar{p}) = q$ holds. By Condition (C_2) of the definition of pump-injection, $r(I(\bar{p})) = r(\bar{p}) = q$ holds, and we are done.

$\Leftarrow$) Assume that $\langle r_{H_i}, r_{\check{H}_i}\rangle \leq \langle r_{H_j}, r_{\check{H}_j}\rangle$ holds. We have to construct a pump-injection $I : (H_i \cup \check{H}_i \cup \mathring{H}_i) \to (H_j \cup \check{H}_j \cup \mathring{H}_j)$. By the definition of pump-injection, the restriction $I : \mathring{H}_i \to \mathring{H}_j$ must be defined as the identity, which is not a problem since $\mathring{H}_i$ is always included in $\mathring{H}_j$. Conditions (C_2) and (C_3) are satisfied for free for these positions. Moreover, for positions $\bar{p}'_1 \in H_i \cup \check{H}_i$ and $\bar{p}'_2 \in \mathring{H}_i$, Condition (C_3) holds whenever Condition (C_1) holds since in this case $\mathtt{term}(r|_{\bar{p}'_1}) \neq_E \mathtt{term}(r|_{\bar{p}'_2})$ and $\mathtt{term}(r|_{I(\bar{p}'_1)}) \neq_E \mathtt{term}(r|_{I(\bar{p}'_2)})$ hold.

Hence, it remains to define $I : (H_i \cup \check{H}_i) \to (H_j \cup \check{H}_j)$. We just define $I : H_i \to H_j$ and prove Conditions (C_2) and (C_3) for $\bar{p}, \bar{p}_1, \bar{p}_2$ in H_i. This is because $I : \check{H}_i \to \check{H}_j$ can be defined analogously, and Conditions (C_2) and (C_3) for the corresponding positions can be checked analogously. Moreover, for positions $\bar{p}'_1 \in H_i$ and $\bar{p}'_2 \in \check{H}_i$, Condition (C_3) holds whenever Condition (C_1) holds since in this case $\mathtt{term}(r|_{\bar{p}'_1}) \neq_E \mathtt{term}(r|_{\bar{p}'_2})$ and $\mathtt{term}(r|_{I(\bar{p}'_1)}) \neq_E \mathtt{term}(r|_{I(\bar{p}'_2)})$ hold. Hence, this simple case is enough to prove the whole statement.

We write the set of equivalence classes of $\{\mathtt{term}(r|_p) \mid p \in H_i\}$ and $\{\mathtt{term}(r|_p) \mid p \in H_j\}$ modulo E more explicitly as $\{[t_{i,1}], \ldots, [t_{i,\alpha}]\}$ and $\{[t_{j,1}], \ldots, [t_{j,\beta}]\}$, respectively. Since $\langle r_{H_i}, r_{\check{H}_i}\rangle \leq \langle r_{H_j}, r_{\check{H}_j}\rangle$ holds, $r_{H_i} \leq r_{H_j}$ also holds. Thus, there exists an injective function $I' : \{1, \ldots, \alpha\} \to \{1, \ldots, \beta\}$ satisfying the following statement for each δ in $\{1, \ldots, \alpha\}$ and each state q of $\mathcal{A}$: $\big|\{p \in H_i \mid \mathtt{term}(r|_p) =_E t_{i,\delta} \wedge r(p) = q\}\big| \leq \big|\{p \in H_j \mid \mathtt{term}(r|_p) =_E t_{j,I'(\delta)} \wedge r(p) = q\}\big| \quad (\dagger)$.

In order to define $I : H_i \to H_j$, we define I for each of such sets $\{p \in H_i \mid \mathtt{term}(r|_p) =_E t_{i,\delta} \wedge r(p) = q\}$ as any injective function $I : \{p \in H_i \mid \mathtt{term}(r|_p) =_E t_{i,\delta} \wedge r(p) = q\} \to \{p \in H_j \mid \mathtt{term}(r|_p) =_E t_{j,I'(\delta)} \wedge r(p) = q\}$, which is possible by the above inequality $(\dagger)$. The global I is then injective thanks to the injectivity of I'. Conditions (C_2) and (C_3) trivially follow from this definition. $\square$

Example 5.19. Following our running example, we first prove $\langle r_{H_4}, r_{\check{H}_4}\rangle \leq \langle r_{H_3}, r_{\check{H}_3}\rangle$. To this end just note that $[\langle 0,0,0,0,1,0\rangle] \leq [\langle 0,0,0,0,1,0\rangle]$ and that $[\langle 0,0,0,1,0,0\rangle] \leq [\langle 0,0,0,2,0,0\rangle]$ hold. We can define $I : (H_4 \cup \check{H}_4 \cup \mathring{H}_4) \to (H_3 \cup \check{H}_3 \cup \mathring{H}_3)$ from this relation according to Lemma 5.18. Doing the adequate guess we obtain the following definition: $I(1) = 1$, $I(2) = 2$, $I(3) = 3.3$ which is the pump-injection considered in Example 5.5 for our running example.

The following lemma follows directly from the definition of the sets H_i and $\check{H}_i$, and allows to connect such definitions with Lemma 5.14.

Lemma 5.20. *Let $\mathcal{A}$ be a* $\texttt{TABG}^{\wedge}$*. Let a be the maximum arity of the symbols in the signature of $\mathcal{A}$. Let r be a run of $\mathcal{A}$. Then, the following conditions hold:*
(1) $|H_{h(r)}| = 1$ *and* $|\check{H}_{h(r)}| = 0$.
(2) *For each i in* $\{2, \dots, h(r)\}$, $a \cdot |H_i| + |\check{H}_i| \geq |H_{i-1}| + |\check{H}_{i-1}|$.
(3) *For each i in* $\{1, \dots, h(r)\}$, $|H_i| = \texttt{sum}(r_{H_i})$ *and* $|\check{H}_i| = \texttt{sum}(r_{\check{H}_i})$.

Proof. Item (1) is trivial by definition of H_i and $\check{H}_i$ for $i = h(r)$. For Item (2), it suffices to observe that the positions in $H_{i-1} \cup \check{H}_{i-1}$ are all the positions in $\check{H}_i$ plus a subset of all child positions of positions in H_i, and that each position has at most a children. For Item (3) we just prove $|H_i| = \texttt{sum}(r_{H_i})$, since $|\check{H}_i| = \texttt{sum}(r_{\check{H}_i})$ can be proved analogously. We write the equivalence classes of the set $\{\texttt{term}(r|_p) \mid p \in H_i\}$ modulo $E = E_{\mathcal{A}}$ more explicitly as $\{[t_1], \dots, [t_\alpha]\}$.

Note that H_i is the disjoint union $\{p \in H_i \mid \texttt{term}(r|_p) =_E t_1\} \cup \dots \cup \{p \in H_i \mid \texttt{term}(r|_p) =_E t_\alpha\}$. Thus, $|H_i|$ equals $|\{p \in H_i \mid \texttt{term}(r|_p) =_E t_1\}| + \dots + |\{p \in H_i \mid \texttt{term}(r|_p) =_E t_\alpha\}|$. We conclude by observing that $|\{p \in H_i \mid \texttt{term}(r|_p) =_E t_1\}| = \texttt{sum}(r_{t_1,H_i})$, ..., $|\{p \in H_i \mid \texttt{term}(r|_p) =_E t_\alpha\}| = \texttt{sum}(r_{t_\alpha,H_i})$ hold. □

Lemma 5.21. *Let $B : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be the computable function of Lemma 5.14. Let $\mathcal{A}$ be a* $\texttt{TABG}^{\wedge}$*. Let a be the maximum arity of the symbols in the signature of $\mathcal{A}$. Let n be the number of states of $\mathcal{A}$. Let r be a run of $\mathcal{A}$ satisfying $h(r) > B(a, n)$. Then, there is a global pumping on r.*

Proof. Consider the sequence $\langle r_{H_{h(r)}}, r_{\check{H}_{h(r)}}\rangle, \dots, \langle r_{H_1}, r_{\check{H}_1}\rangle$. Note that the n-tuple $\langle 0, \dots, 0\rangle$ does not appear in the multisets of the pairs of this sequence. By Lemma 5.20, $|H_{h(r)}| = 1$ and $|\check{H}_{h(r)}| = 0$ hold, and for each i in $\{2, \dots, h(r)\}$, $a \cdot |H_i| + |\check{H}_i| \geq |H_{i-1}| + |\check{H}_{i-1}|$ holds. Moreover, for each i in $\{1, \dots, h(r)\}$, $|H_i| = \texttt{sum}(r_{H_i})$ and $|\check{H}_i| = \texttt{sum}(r_{\check{H}_i})$ hold. Thus, $\texttt{sum}(r_{H_{h(r)}}) = 1$, $\texttt{sum}(r_{\check{H}_{h(r)}}) = 0$, and for each i in $\{2, \dots, h(r)\}$, $a \cdot \texttt{sum}(r_{H_i}) + \texttt{sum}(r_{\check{H}_i}) \geq \texttt{sum}(r_{H_{i-1}}) + \texttt{sum}(r_{\check{H}_{i-1}})$ hold. Hence, since $h(r) \geq B(a, n)$ holds, by Lemma 5.14 there exist i, j satisfying $h(r) \geq i > j \geq 1$ and $\langle r_{H_i}, r_{\check{H}_i}\rangle \leq \langle r_{H_j}, r_{\check{H}_j}\rangle$. By Lemma 5.18, there exists a pump-injection $I : (H_i \cup \check{H}_i \cup \mathring{H}_i) \to (H_j \cup \check{H}_j \cup \mathring{H}_j)$. Therefore, there exists a global pumping on r. □

Theorem 5.22. *Emptiness is decidable for* $\texttt{TABG}^{\wedge}$.

Proof. Let a be the maximum arity of the symbols in the signature of $\mathcal{A}$. Let n be the number of states of $\mathcal{A}$. Let r be an accepting run of $\mathcal{A}$ with minimum height.

Suppose that $h(r) \geq B(a, n)$ holds. Then, by Lemma 5.21, there exists a global pumping r' on r. By Corollary 5.7, $h(r') < h(r)$ holds. Moreover, by the definition of global pumping, $r'(\lambda) = r(\lambda)$ holds. Finally, by Corollary 5.11, r' is a run of $\mathcal{A}$. Thus, r' contradicts the minimality of r. We conclude that $h(r) < B(a, n)$ holds.

The decidability of emptiness of $\mathcal{A}$ follows, since the existence of successful runs implies that one of them can be found among a computable and finite set of possibilities. $\square$

Using Corollary 4.20 and Theorem 5.22, we can conclude the decidability of emptiness for TABG, and more generally for TABG$[\approx, \not\approx, \mathbb{N}]$.

Corollary 5.23. *Emptiness is decidable for TABG.*

Corollary 5.24. *Emptiness is decidable for TABG*$[\approx, \not\approx, \mathbb{N}]$.

6. Unranked Ordered Trees

Our tree automata models and results can be generalized from ranked to unranked ordered terms. In this setting, Σ is called an *unranked signature*, meaning that there is no arity fixed for its symbols, i.e. that in a term $a(t_1, \ldots, t_n)$, the number n of children is arbitrary and does not depend on a. Let us denote by $\mathcal{U}(\Sigma)$ the set of unranked ordered terms over Σ. The notions of positions, subterms, etc., are defined for unranked terms of $\mathcal{U}(\Sigma)$ as for ranked terms of $\mathcal{T}(\Sigma)$.

We extend the definition of automata for unranked ordered terms, called hedge automata [Mur99], with global constraints. We do not consider constraints between brothers nor flat theories in this setting.

Definition 6.1. A *hedge automaton with global constraints* (HAG) over an unranked signature Σ is a tuple $\mathcal{A} = \langle Q, \Sigma, F, \Delta, C\rangle$ where Q is a finite set of states, $F \subseteq Q$ is the subset of final states, C is a Boolean combination of atomic constraints of the form $q \approx q'$ or $q \not\approx q'$, with $q, q' \in Q$, and Δ is a set of transition rules of the form $a(L) \to q$ where $a \in \Sigma$, $q \in Q$ and L is a regular (word) language over Q^*, assumed given by a finite state automaton with input alphabet Q.

We still use the notation HAG$[\tau_1, \ldots, \tau_n]$ where the types τ_i can be $\approx$, $\not\approx$, $|.|_{\mathbb{N}}$, $\|.\|_{\mathbb{N}}$, $\mathbb{N}$.

The notion of run of TAG is extended to HAG in the natural way. A *run* of a HAG $\mathcal{A}$ is a pair $r = \langle t, M\rangle$ where $t \in \mathcal{U}(\Sigma)$ is an unranked ordered term and M is a mapping from $Pos(t)$ into $\Delta_{\mathcal{A}}$ such that for each position $p \in Pos(t)$ with n children, if $M(p.1), \ldots, M(p.n)$ are rules with right-hand side states $q_1, \ldots, q_n \in Q_{\mathcal{A}}$, respectively, then $M(p)$ is a transition rule of the form $t(p)(L) \to q$ in Δ, and the word $q_1 \cdots q_n$ belongs to L. Moreover, $r \models C_{\mathcal{A}}$, where satisfiability of $C_{\mathcal{A}}$ by r is defined like in Section 3. A run r is called *successful* (or *accepting*) if $r(\lambda) \in F_{\mathcal{A}}$.

The emptiness decision results of Corollary 5.24 can be transposed from TAG into HAG using a standard transformation from unranked to ranked binary terms, like the *extension* encoding described in [CDG+07], Chapter 8.

Let us associate to the unranked signature Σ the (ranked) signature $\Sigma_@ := \{a : 0 \mid a \in \Sigma\} \cup \{@ : 2\}$ where @ is a new symbol not in Σ. The operator curry is a bijection from $\mathcal{U}(\Sigma)$ into $\mathcal{T}(\Sigma_@)$ recursively defined as follows:

$$\begin{array}{rcl} \mathtt{curry}(a) & = & a \quad \text{for all } a \in \Sigma \\ \mathtt{curry}\big(a(t_1, \ldots, t_n)\big) & = & @\big(\mathtt{curry}\big(a(t_1, \ldots, t_{n-1})\big), \mathtt{curry}(t_n)\big) \end{array}$$

An example of application of this operator is presented in Figure 7. We extend the application of the operator curry to sets of unranked ordered terms by $\mathtt{curry}(L) = \{\mathtt{curry}(t) \mid t \in L\}$.

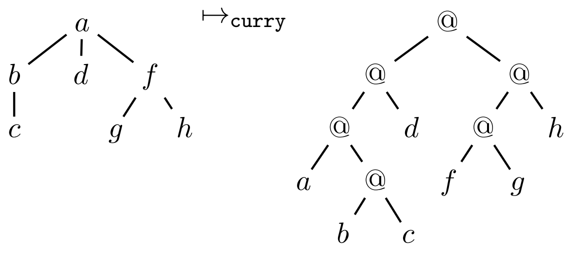

Figure 7: Currying an unranked term.

Proposition 6.2. *For all* HAG$[\approx, \not\approx, \mathbb{N}]$ $\mathcal{A}$ *over* Σ, *one can construct effectively in PTIME a* TAG$[\approx, \not\approx, \mathbb{N}]$ $\mathcal{A}'$ *over* $\Sigma_@$ *such that* $\mathcal{L}(\mathcal{A}') = \texttt{curry}\big(\mathcal{L}(\mathcal{A})\big)$.

Proof. Let $\mathcal{A}$ be $\langle Q, \Sigma, F, \Delta, C\rangle$ more explicitly written. Without loss of generality, we assume that for each $a \in \Sigma, q \in Q$, the set of rules Δ contains exactly one transition of the form $a(L) \to q$, and we denote by $\bar{A}_{a,q}$ the NFA recognizing the corresponding language L. Recall that such automata have Q as input alphabet. Without loss of generality, we assume that the sets of states of $\mathcal{A}$ and all $\bar{A}_{a,q}$ are pairwise disjoint. Let $\bar{Q}$ be the union of all states of all the automata $\bar{A}_{a,q}$. Intuitively, the transitions of the automaton $\mathcal{A}'$ will simulate both the transitions of $\mathcal{A}$ and the transitions of the NFAs $\bar{A}_{a,q}$, when running on $\texttt{curry}(t)$ for some $t \in \mathcal{U}(\Sigma)$.

Let $\mathcal{A}' = \langle Q \cup \bar{Q}, \Sigma, F, \Delta', C\rangle$ where Δ' contains the following transitions for each $a \in \Sigma, q \in Q$:

- $a \to q$ if $\bar{A}_{a,q}$ recognizes the empty word,
- $a \to \bar{q}$ where $\bar{q}$ is the initial state of $\bar{A}_{a,q}$,
- $@(\bar{q}, q') \to \bar{q}'$ if there is a transition $\bar{q} \xrightarrow{q'} \bar{q}'$ in $\bar{A}_{a,q}$, and
- $@(\bar{q}, q') \to q$ if there is a transition $\bar{q} \xrightarrow{q'} \bar{q}'$ in $\bar{A}_{a,q}$ and $\bar{q}'$ is a final state of $\bar{A}_{a,q}$.

It is not difficult to see that there exists an accepting run of $\mathcal{A}$ if and only if there exists an accepting run of $\mathcal{A}'$. ☐

There exist alternative encodings from unranked to ranked trees in the literature, e.g., the first-child next-sibling encoding: see Figure 8 for an example of this transformation. This alternative encoding makes the representation of equality and disequality between subterms of the original unranked term difficult, since the transformed subterms may have original siblings occurring now as their subterms. For example, in Figure 8, the two occurrences of the subterm c correspond to different terms in the result of the transformation.

The following emptiness decision result is a direct consequence of Proposition 6.2 and Corollary 5.24.

Corollary 6.3. *Emptiness is decidable for* HAG$[\approx, \not\approx, \mathbb{N}]$.

7. Logics on Trees

In this section, we discuss the application of our results to second order logics interpreted over domains defined by terms. We propose a strict extension of the second order monadic logic of the tree with equality, disequality and arithmetic constraints, and show that satisfiability is decidable for this extension thanks to a correspondence with TAG$[\approx, \not\approx, \mathbb{N}]$.

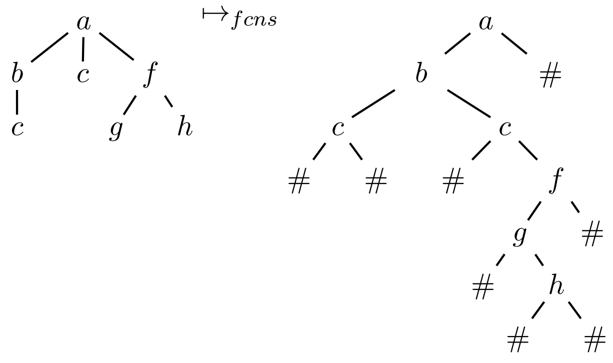

Figure 8: First-child next-sibling encoding of an unranked term.

7.1. **MSO on Ranked Terms.** A ranked term $t \in \mathcal{T}(\Sigma)$ over Σ can be seen as a model for logical formulae, with an interpretation domain which is the set of positions $Pos(t)$. We consider monadic second order formulae interpreted on such models, built with the usual Boolean connectors, with quantifications over first order variables (interpreted as positions), denoted $x, y \ldots$ and over unary predicates (i.e. second order variables interpreted as sets of positions), denoted $X, Y \ldots$, and with the following predicates,

- equality: $x = y$,
- membership: $X(x)$,
- labeling: $a(x)$, for $a \in \Sigma$
- navigation: $S_i(x, y)$, for all i smaller than or equal to the maximal arity of symbols of Σ (we call $+1$ the type of such predicates),
- term equality: $X \approx Y$, term disequality: $X \not\approx Y$ (predicate types $\approx$ and $\not\approx$),
- linear inequalities: $\sum a_i \cdot |X_i| \geq a$ or $\sum a_i \cdot \|X_i\| \geq a$, where every a_i and a belong to $\mathbb{Z}$ (predicate types $|.|_{\mathbb{Z}}$ and $\|.\|_{\mathbb{Z}}$).

We write $\mathtt{MSO}[\tau_1, \ldots, \tau_k]$ for the set of monadic second order logic formulae with equality, membership, labeling predicates and other predicates of types $\tau_1, \ldots, \tau_k$, amongst the above types $+1$, $\approx$, $\not\approx$, and $|.|_{\mathbb{Z}}$, $\|.\|_{\mathbb{Z}}$. We also use the notations $|.|_{\mathbb{N}}$ and $\|.\|_{\mathbb{N}}$ for natural linear inequalities (linear inequalities whose coefficient all have the same sign) and the abbreviations $\mathbb{Z}$ and $\mathbb{N}$ of Section 4.

Let $\exists\mathtt{MSO}[\tau_1, \ldots, \tau_k]$ be the fragment of $\mathtt{MSO}[\tau_1, \ldots, \tau_k]$ containing the formulae of the form $\exists X_1 \ldots \exists X_n\, \phi$ such that all the atoms of type $\approx$, $\not\approx$, $\mathbb{Z}$ or $\mathbb{N}$ in ϕ involve only second order variables amongst $X_1, \ldots, X_n$.

A variable assignment into a term $t \in \mathcal{T}(\Sigma)$ is a function σ mapping first order variables into positions of $Pos(t)$ and second order variables into subsets of $Pos(t)$. The satisfiability of a formula ϕ by a term $t \in \mathcal{T}(\Sigma)$ and a variable assignment σ, denoted $t, \sigma \models \phi$ is defined in the usual Tarskian manner, with:

$$\begin{array}{lll}
t,\sigma \models x = y & \text{iff} & \sigma(x) = \sigma(y) \\
t,\sigma \models X(x) & \text{iff} & \sigma(x) \in \sigma(X) \\
t,\sigma \models a(x) & \text{iff} & t(\sigma(x)) = a \\
t,\sigma \models S_i(x,y) & \text{iff} & \sigma(x).i = \sigma(y) \\
t,\sigma \models X \approx Y & \text{iff} & \forall p \in \sigma(X), p' \in \sigma(Y), p \neq p' :\ t|_p = t|_{p'} \\
t,\sigma \models X \not\approx Y & \text{iff} & \forall p \in \sigma(X), p' \in \sigma(Y), p \neq p' :\ t|_p \neq t|_{p'} \\
t,\sigma \models \sum a_i \cdot |X_i| \geq a & \text{iff} & \sum_i a_i \cdot |\sigma(X_i)| \geq a \\
t,\sigma \models \sum a_i \cdot \|X_i\| \geq a & \text{iff} & \sum_i a_i \cdot \big|\{t|_p \mid p \in \sigma(X_i)\}\big| \geq a
\end{array}$$

Example 7.1. The following formula of $\exists$MSO$[\approx, \not\approx]$ expresses that all the subterms headed by a in a term t are pairwise different: $\exists X_a\, ((\forall x\, X_a(x) \leftrightarrow a(x)) \wedge X_a \not\approx X_a)$. In other words, a is used to mark monadic keys in t (see Example 3.4).

A seminal result of [TW68] shows that MSO$[+1]$ has exactly the same expressiveness as TA, and therefore it is decidable. The extension MSO$[+1, \approx]$ is undecidable, see e.g. [FTT07]. The extension MSO$[+1, |.|_{\mathbb{Z}}]$ is undecidable as well [KR02].

On the other side, the fragment $\exists$MSO$[+1, |.|_{\mathbb{Z}}]$ is decidable [KR02], and a fragment of $\exists$MSO$[+1, \approx, \not\approx]$ is shown decidable in [FTT08] for a restricted variant of $\not\approx$, using a two way correspondence between these formulae and a decidable subclass of TAGED.

This latter construction can be straightforwardly adapted to establish a two way correspondence between $\exists$MSO$[+1, \approx, \not\approx, \mathbb{N}]$ and TAG$[\approx, \not\approx, \mathbb{N}]$.

Theorem 7.2. $\exists$MSO$[+1, \approx, \not\approx, \mathbb{N}]$ *is decidable on ranked terms.*

Proof. Following the same proof scheme as [FTT08], we show that for every closed formula ϕ in $\exists$MSO$[+1, \approx, \not\approx, \mathbb{N}]$, we can construct a TAG$[\approx, \not\approx, \mathbb{N}]$ recognizing exactly the set of models of ϕ. Then, the decidability of the logic follows from Theorem 5.24.

Without loss of generality, we may assume that ϕ is of the form

$$\exists X_1 \ldots \exists X_n\ (\phi_0(\overline{X}) \wedge \phi_{\approx}(\overline{X}) \wedge \phi_{\mathbb{N}}(\overline{X}))$$

where $\phi_0(\overline{X})$ is a MSO$[+1]$ formula with free variables $\overline{X} = X_1, \ldots, X_n$, and $\phi_{\approx}(\overline{X})$ and $\phi_{\mathbb{N}}(\overline{X})$ are Boolean combinations of atoms of the respective form $X_i \approx X_j$, $X_i \not\approx X_j$ and $\sum a_i \cdot |X_i| \geq a$, $\sum a_i \cdot \|X_i\| \geq a$. Moreover, we shall also assume that $\phi_{\approx}(\overline{X})$ and $\phi_{\mathbb{N}}(\overline{X})$ are conjunctions of atoms or negations of atoms of the above form. Otherwise, we put them into disjunctive normal form and then split ϕ into an equivalent formula $\phi_1 \vee \ldots \vee \phi_k$, where each ϕ_i, $i \leq k$, is of the form requested: $\phi_i = \exists X_1 \ldots \exists X_n\ (\phi^i_0(\overline{X}) \wedge \phi^i_{\approx}(\overline{X}) \wedge \phi^i_{\mathbb{N}}(\overline{X}))$, where $\phi^i_0(\overline{X}) \in$ MSO$[+1]$ and $\phi^i_{\approx}(\overline{X})$ and $\phi^i_{\mathbb{N}}(\overline{X})$ are conjunctions of atoms or negations of atoms as above, and we solve satisfiability separately for each ϕ_i.

First, we recall the definitions of [TW68] of the signature $\Sigma \times \{0,1\}^n$, where the arity of a symbol $\langle f, b_1, \ldots, b_n\rangle$ is the arity of f, and of the term $t \otimes \sigma$ over this signature obtained, from a term t over Σ and a mapping $\sigma : \{X_1, \ldots, X_n\} \to 2^{Pos(t)}$, by relabeling every position $p \in Pos(t)$ by $\langle t(p), b_1, \ldots, b_n\rangle$, where for each $i \leq n$, $b_i = 1$ if $p \in \sigma(X_i)$ and $b_i = 0$ otherwise. Also, from [TW68] we get the construction of a TA $\mathcal{A}_0 = \langle Q, \Sigma \times \{0,1\}^n, F, \Delta_0\rangle$ which recognizes the set of terms $\{t \otimes \sigma \in \mathcal{T}\big(\Sigma \times \{0,1\}^n\big) \mid t, \sigma \models \phi_0(\overline{X})\}$.

Second, following a construction in [NPTT05], we shift in $\mathcal{A}_0$ the bit-vectors from the signature into the state symbols, obtaining a TA $\mathcal{A}'_0 = \langle Q \times \{0,1\}^n, \Sigma, F \times \{0,1\}^n, \Delta\rangle$ where Δ contains all the transition rules

$$f\big(\langle q_1, b_{1,1}, \ldots, b_{1,n}\rangle, \ldots, \langle q_m, b_{m,1}, \ldots, b_{m,n}\rangle\big) \to \langle q, b_1, \ldots, b_n\rangle$$

such that $f \in \Sigma$, $\langle f, b_1, \ldots, b_n\rangle\big(q_1, \ldots, q_m\big) \to q \in \Delta_0$ and $b_{1,1}, \ldots, b_{1,n}, \ldots, b_{m,1}, \ldots, b_{m,n} \in \{0,1\}$. This automaton $\mathcal{A}'_0$ recognizes the projection (on the first components) of the terms recognized by $\mathcal{A}_0$, i.e. it recognizes the set of terms $t \in \mathcal{T}(\Sigma)$ such that there exists $\sigma : \{X_1, \ldots, X_n\} \to 2^{Pos(t)}$ satisfying $t, \sigma \models \phi_0(\overline{X})$.

Third, we obtain a constraint C by rewriting all the atoms of $\phi_{\approx}(\overline{X}) \wedge \phi_{\mathbb{N}}(\overline{X})$ with the following rules:

$$\begin{array}{rcl} X_i \approx X_j & \mapsto & \displaystyle\bigwedge_{b_i = b'_j = 1} \langle q, b_1, \ldots, b_n\rangle \approx \langle q', b'_1, \ldots, b'_n\rangle \\ X_i \not\approx X_j & \mapsto & \displaystyle\bigwedge_{b_i = b'_j = 1} \langle q, b_1, \ldots, b_n\rangle \not\approx \langle q', b'_1, \ldots, b'_n\rangle \\ \displaystyle\sum_i a_i \cdot |X_i| \geq a & \mapsto & \displaystyle\sum_i \sum_{b_i = 1} a_i \cdot |\langle q, b_1, \ldots, b_n\rangle| \geq a \\ \displaystyle\sum_i a_i \cdot \|X_i\| \geq a & \mapsto & \displaystyle\sum_i \sum_{b_i = 1} a_i \cdot \|\langle q, b_1, \ldots, b_n\rangle\| \geq a \end{array}$$

The `TAG`$[\approx, \not\approx, \mathbb{N}]$ $\mathcal{A} = \langle Q \times \{0,1\}^n, \Sigma, F \times \{0,1\}^n, \Delta, C\rangle$ recognizes $\{t \in \mathcal{L}(\mathcal{A}) \mid t \models \phi\}$. $\square$

The above transformation also works in the other direction (this result is not necessary for the proof of Theorem 7.2 though): for every `TAG`$[\approx, \not\approx, \mathbb{N}]$, we can construct a formula ϕ in $\exists$`MSO`$[+1, \approx, \not\approx, \mathbb{N}]$, whose set of models is $\mathcal{L}(\mathcal{A})$.

Note that $\exists$`MSO`$[+1, \approx]$ is strictly more expressive than `MSO`, since the equality between subterms is not expressible in `MSO` (see e.g. [CDG+07]). The `TA` construction of [TW68] for the decidability of `MSO`$[+1]$ involves the closure under projection on components for `TA` languages over signatures made of tuples of symbols (for the elimination of $\exists$ quantifiers). `TAG` languages are not closed under projection on some components of tuples, as it is already the case for simpler form tree automata with equality [Tre00]. Thus, the same approach cannot be used to prove decidability of emptiness of `TAG`.

7.2. **MSO on Unranked Ordered Terms.** In unranked ordered terms of $\mathcal{U}(\Sigma)$, the number of children of a position is unbounded. Therefore, for navigating in such terms with logical formulae, the successor predicates $S_i(x, y)$ of Section 7.1 are not sufficient. In order to describe unranked ordered terms as models, we replace these above predicates S_i by:

- $S_{\downarrow}(x, y)$ (y is a child of x),
- $S_{\rightarrow}(x, y)$ (y is the successor sibling of x).

The type of these predicates is still called $+1$. Note that the above predicates $S_1, S_2, \ldots$ can be expressed using these two predicates only.

The satisfiability of the above atoms by a term $t \in \mathcal{U}(\Sigma)$ and a variable assignment σ is defined as follows:

$$\begin{array}{rll} t, \sigma \models S_{\downarrow}(x, y) & \text{iff} & \text{there exists } i \text{ such that } \sigma(x).i = \sigma(y), \\ t, \sigma \models S_{\rightarrow}(x, y) & \text{iff} & \text{there exists } p \in Pos(t) \text{ and } i \text{ such that } \sigma(x) = p.i \\ & & \text{and } \sigma(y) = p.(i+1). \end{array}$$

It is shown in [SSM03] that the extension `MSO`$[+1, |.|_{\mathbb{Z}}]$ is undecidable for unranked ordered terms when counting constraints are applied to sibling positions.

Using the results of Section 6, and an easy adaptation of the automata construction in the proof of Theorem 7.2, we can generalize Theorem 7.2 to $\exists$`MSO` over unranked ordered terms.

Theorem 7.3. $\exists\texttt{MSO}[+1, \approx, \not\approx, \mathbb{N}]$ *is decidable on unranked ordered terms.*

8. Conclusion

We have answered (positively) the open problem of decidability of the emptiness problem for the TAGED [FTT08], by proposing a decision algorithm for a class TABG of tree automata with global constraints strictly extending the global constraints of TAGED in several directions. Moreover, the TABG combine the global constraints with local tests between brother subterms a la [BT92] and equality interpreted modulo flat theories. Our method for emptiness decision, presented in Section 5 appeared to be robust enough to deal with several extensions like global counting constraints, and generalization to unranked terms.

A challenging question would be to investigate the precise complexity of the emptiness problem, avoiding the use of Higman's Lemma in the algorithm. For instance, in [FTT08], it is shown, using a direct reduction into solving positive and negative set constraints [CP94, GTT94, Ste94], that emptiness is decidable in NEXPTIME for TAGED (i.e. for $\texttt{TAG}^\wedge[\not\approx]$ modulo an empty theory and such that in every atomic constraint $q \not\approx q'$, q and q' are distinct states). On the other hand, the best known lower bound for emptiness decision for TABG is EXPTIME-hardness (this holds already for $\texttt{TAG}^\wedge[\approx]$ as shown in [FTT08]).

Another interesting problem mentioned in the introduction is the combination of the HAG of Section 6 with the unranked tree automata with tests between siblings, UTASC [WL07, LW09]. Perhaps, the techniques of Section 5 could help for the emptiness decision for a formalism using for instance MSO binary querying (following e.g. [NPTT05]) for selecting the test position of global constraints.

Finally, another branch of research related to TABG concerns automata and logics for *data trees*, i.e. trees labeled over an infinite (countable) alphabet (see [Seg06] for a survey). Indeed, data trees can be represented by terms over a finite alphabet, with an encoding of the data values into terms. This can be done in several ways, and with such encodings, the data equality relation becomes the equality between subterms. Therefore, this could be worth studying in order to relate our results on TAG to decidability results on automata or logics on data trees like those in [JL07, BMSL09].

Acknowledgement

We thank Luc Segoufin for many valuable discussions and the anonymous referees at LICS 2010 for their useful comments and suggestions.

References

[ANR05] S. Anantharaman, P. Narendran, and M. Rusinowitch. Closure properties and decision problems of DAG automata. *Information Processing Letters*, 94(5):231–240, 2005.

[BCG$^+$10] L. Barguñó, C. Creus, G. Godoy, F. Jacquemard, and C. Vacher. The emptiness problem for tree automata with global constraints. In *Logic in Computer Science (LICS)*, pages 263–272, 2010.

[BMSL09] M. Bojanczyk, A. Muscholl, T. Schwentick, and Segoufin L. Two-variable logic on data trees and applications to XML reasoning. *JACM*, 56(3), 2009. A preliminary version was presented at PODS 06.

[BN98] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, New York, 1998.

[BT92] B. Bogaert and S. Tison. Equality and Disequality Constraints on Direct Subterms in Tree Automata. In *9th Symp. on Theoretical Aspects of Computer Science, STACS*, volume 577 of *LNCS*, pages 161–171. Springer, 1992.

[BT05] A. Bouajjani and T. Touili. On computing reachability sets of process rewrite systems. In Jürgen Giesl, editor, *Term Rewriting and Applications, 16th International Conference, RTA 2005, Nara, Japan, April 19-21, 2005, Proceedings*, volume 3467 of *Lecture Notes in Computer Science*, pages 484–499. Springer, 2005.

[CC05] H. Comon and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. *Theoretical Computer Science*, 331(1):143–214, February 2005.

[CDG$^+$07] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, C. Löding, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. `http://tata.gforge.inria.fr`, 2007.

[Cha99] W. Charatonik. Automata on DAG representations of finite trees. Technical Report Technical Report MPI-I-99-2-001, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1999.

[CHJ94] H. Comon, M. Haberstrau, and J.P. Jouannaud. Syntacticness, cycle-syntacticness, and shallow theories. *Information and Computation*, 111(1):154–191, 1994.

[CP94] W. Charatonik and L. Pacholski. Set constraints with projections are in NEXPTIME. In *Proceedings of the* 35^{th} *Symp. Foundations of Computer Science*, pages 642–653, 1994.

[DL06] S. Dal and D. Lugiez. XML schema, tree logic and sheaves automata. *Journal Applicable Algebra in Engineering, Communication and Computing*, 17(5):337–377, 2006.

[FGVTT04] G. Feuillade, T. Genet, and V. Viet Triem Tong. Reachability Analysis over Term Rewriting Systems. *Journal of Automated Reasoning*, 33 (3-4):341–383, 2004.

[FTT07] E. Filiot, J.-M. Talbot, and S. Tison. Satisfiability of a spatial logic with tree variables. In *Proceedings of the 21st International Workshop on Computer Science Logic (CSL 2007)*, volume 4646 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2007.

[FTT08] E. Filiot, J.-M. Talbot, and S. Tison. Tree automata with global constraints. In *12th International Conference in Developments in Language Theory (DLT 2008)*, volume 5257 of *Lecture Notes in Computer Science*, pages 314–326. Springer, 2008.

[Gal91] J. H. Gallier. What's so special about kruskal's theorem and the ordinal gamma_0? a survey of some results in proof theory. *Annals of Pure Applied Logic*, 53(3):199–260, 1991.

[GTT94] R. Gilleron, S. Tison, and M. Tommasi. Some new decidability results on positive and negative set constraints. In *Proceedings, First International Conference on Constraints in Computational Logics*, volume 845 of *LNCS*, pages 336–351. Spinger, 1994.

[JKV09] F. Jacquemard, F. Klay, and C. Vacher. Rigid tree automata. In Adrian Horia Dediu, Armand Mihai Ionescu, and Carlos Martín-Vide, editors, *Proceedings of the 3rd International Conference on Language and Automata Theory and Applications (LATA'09)*, volume 5457 of *Lecture Notes in Computer Science*, pages 446–457, Tarragona, Spain, April 2009. Springer.

[JL07] M. Jurdzinski and R. Lazic. Alternation-free modal mu-calculus for data trees. In *Logic in Computer Science (LICS)*, pages 131–140. IEEE Computer Society, 2007.

[KR02] F. Klaedtke and H. Ruess. Parikh automata and monadic second-order logics with linear cardinality constraints. Technical Report 177, Intitute of Computer Science at Freiburg University, 2002.

[LW09] C. Löding and K. Wong. On nondeterministic unranked tree automata with sibling constraints. In *In IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2009)*, Leibniz International Proceedings in Informatics. Schloss Dagstuhl - Leibniz Center for Informatics, 2009.

[Mon81] J. Mongy. *Transformation de noyaux reconnaissables d'arbres. Forêts RATEG.* PhD thesis, Laboratoire d'Informatique Fondamentale de Lille, Université des Sciences et Technologies de Lille, Villeneuve d'Ascq, France, 1981.

[Mur99] M. Murata. Hedge automata: a formal model for XML schemata. Technical report, Fuji Xerox INformation Systems, 1999.

[Nie96] R. Nieuwenhuis. Basic paramodulation and decidable theories (extended abstract). In *Logic in Computer Science (LICS)*, pages 473–482, 1996.

[NPTT05] J. Niehren, L. Planque, J.-M. Talbot, and S. Tison. N-ary queries by tree automata. In *Proceedings of the 10th International Symposium on Database Programming Languages (DBPL)*, volume 3774 of *Lecture Notes in Computer Science*, pages 217–231. Springer, 2005.

[Sch07] T. Schwentick. Automata for XML - a survey. *J. Comput. Syst. Sci.*, 73(3):289–315, 2007.

[Seg06] L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Computer Science Logic*, volume 4207 of *LNCS*. Springer, 2006.

[SSM03] H. Seidl, T. Schwentick, and A. Muscholl. Numerical document queries. In *Principle of Databases Systems (PODS)*, pages 155–166. ACM Press, 2003.

[Ste94] K. Stefansson. Systems of set constraints with negative constraints are nexptime-complete. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 137–141. IEEE Computer Society Press, 1994.

[Tre00] R. Treinen. Predicate logic and tree automata with tests. In J. Tiuryn, editor, *Proc. of the 3rd Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS*, volume 1784 of *LNCS*, pages 329–343. Springer, 2000.

[TW68] J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical System Theory*, 2:57–82, 1968.

[VGL07] K. N. Verma and J. Goubault-Larrecq. Alternating two-way ac-tree automata. *Information and Computation*, 205(6):817–869, 2007.

[WL07] K. Wong and C. Löding. Unranked tree automata with sibling equalities and disequalities. In *In Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of *LNCS*, pages 875–887. Springer, 2007.