Tree Automata with Constraints and Tree Homomorphisms

This dissertation has been submitted to the
Department of Computer Science (CS) from
Universitat Politècnica de Catalunya (UPC)

in partial fulfilment of the requirements
for the degree of *Doctor of Philosophy*

by

Carles Creus López

under the direction of

Guillem Godoy Balil

Barcelona, April 2016

Abstract

Automata are a widely used formalism in computer science as a concise representation for sets. They are interesting from a theoretical and practical point of view. This work is focused on automata that are executed on tree-like structures, and thus, define sets of trees. Moreover, we tackle automata that are enhanced with the possibility to check (dis)equality constraints, i.e., where the automata are able to test whether specific subtrees of the input tree are equal or different. Two distinct mechanisms are considered for defining which subtrees have to be compared in the evaluation of the constraints. First, in local constraints, a transition of the automaton compares subtrees pending at positions relative to the position of the input tree where the transition takes place. Second, in global constraints, the subtrees tested are selected depending on the state to which they are evaluated by the automaton during a computation.

In the setting of local constraints, we introduce tree automata with height constraints between brothers ($\mathtt{TACBB}_{\mathsf{H}}$). These constraints are predicates on sibling subtrees that, instead of evaluating whether the subtrees are equal or different, compare their respective heights. Such constraints allow to express natural tree sets like complete or balanced (like AVL) trees. We prove decidability of emptiness and finiteness for $\mathtt{TACBB}_{\mathsf{H}}$, and also for the combination of $\mathtt{TACBB}_{\mathsf{H}}$ with the tree automata with (dis)equality constraints between brothers of Bogaert and Tison (1992). We also define a new class of tree automata with constraints that allows arbitrary local disequality constraints and a particular kind of local equality constraints ($\mathtt{TA}_{\mathsf{ihom},\not\approx}$). We prove decidability of emptiness and finiteness for this class in exponential time. As a consequence, we obtain several EXPTIME-completeness results for problems on images of regular tree sets under tree homomorphisms, like set inclusion, finiteness of set difference, and regularity (also called HOM problem).

In the setting of global constraints, we study the class of tree automata with global reflexive disequality constraints ($\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$). Such kind of constraints is incomparable with the original notion of global disequality constraints of Filiot et al. (2007): the latter restricts disequality tests to only compare subtrees evaluated to distinct states, whereas in $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ it is possible to test that all subtrees evaluated to the same given state are pairwise different. The tests of $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ correspond to monadic key constraints, and thus, can be used to characterize unique identifiers, a typical integrity constraint of XML schemas. We study the emptiness and finiteness problems for $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$, and obtain decision algorithms that take triple exponential time.

Acknowledgments[1]

First and foremost, I have to thank Guillem. This thesis would have been impossible without him, and the reasons for that are probably too many to enumerate (within this page; I guess—hope—they are countable). He has always been able to shed some light on the darkest problems we tackled, and in many cases, anticipate dead ends when I was just beginning to grasp the setting. Working with him is an experience that I would recommend to any student (and non-student, too), as there is so much to learn just by following him along. In my case, I've done my best to keep up.

This thesis traces its origin to quite long ago, and I've been able to meet inspiring colleagues in that time; chronologically: Francesc, Luis, Camille, Florent, Lander, Adrià, Nil, and Pau. Also, and in between, Joachim, Adrien, Aurélien, and Sophie. Each brought new points of view, attitude, and insights; our paths might have diverged, but I hope that I still carry the lessons learned from all of you.

Al igual que el Dr. Gascón va mencionar en els seus acknowledgments les nostres sessions d'escriptura, jo no puc sinó fer el mateix. M'agradaria afegir el bon record que tinc de les llargues discussions sobre la notació que ens calia usar. Un dels aspectes interessants de parlar-ho en el seu despatx era veure com distreiem al "veí de taula" i li feiem abandonar les seves converses en francès amb la paret. De tant en tant, distreiem veïns més llunyans i tot, però ja no puc endevinar la mena d'activitats que deviem interrompre. Dedico el Capítol 5 a tots els al·ludits.

De les quatre plantes de l'Ω per les que he passat, el temps en l'S1 ha estat sens dubte el més enriquidor, i la convivència allí ha estat cabdal en que jo sobrevisqui a la tesi. Des del grup de gràfics, seguint amb les incorporacions des d'IA, els habitants de la zona alliberada i la no-alliberada, la pastissEva (i els no-pastissers que ens portaven els dolços de la mare), i acabant per l'última onada d'immigració que hi ha hagut: agraeixo a tots la vostra companyia. Guardaré el record dels sopars de despatxos, els jocs de taula, i especialment les discussions dialectals del dinar (cebada, ordi; avena, civada; ¿qué es eso? eso es queso).

Y por último, pero no por ello menos importante, también debo agradecer el apoyo de toda esa gente externa al trabajo sin cuya presencia mi camino hubiera resultado una cuesta muy empinada (se sobreentiende: hacia arriba). Os dejaré anónimos, pues sois muchos y temo dejarme a alguno por mi mala memoria; obvio: aquí entra toda mi familia y bola de amigos. En especial, dedico esta tesis a mi padre, madre, y hermano. Finalmente, aprovecho para pedir perdón por todas esas caras de póquer que obligué a adoptar a los incautos que me preguntaron: ¿de qué trata tu tesis?

[1]This work has been partially supported by funds from the Spanish Ministry of Economy and Competitiveness (MINECO) and the European Union (FEDER funds) under grant COMMAS (ref. TIN2013-46181-C2-1-R), and by funds from the Spanish Ministry of Science and Innovation (MICINN) under grant FORMALISM (ref. TIN2007-66523). Additionally, the author has been supported by an FPI-UPC 2011 grant from Universitat Politècnica de Catalunya, an FI-DGR 2012 grant from Generalitat de Catalunya, and an FPU-MECD 2010 grant from the Spanish Ministry of Education.

Contents

1 Introduction **3**
1.1 Local constraints, and tree homomorphisms . . . 5
1.2 Global constraints . . . 8
1.3 Organization and summary . . . 9

2 Basic concepts and notations **15**
2.1 General notation on sets and complexity classes . . . 15
2.2 Terms . . . 16
2.3 Positions . . . 17
2.4 Functions and operations on terms . . . 18
2.5 Tree automata . . . 19
2.6 Substitutions, tree homomorphisms, rewriting . . . 20

3 Decidability of height and equality constraints between brothers **23**
3.1 `TA` with height and equality constraints between brothers . . . 24
3.2 Decidability of simple height constraints . . . 25
3.3 Decidability of the general case . . . 29

4 EXPTIME-completeness of the HOM problem **39**
4.1 Intuition on the approach . . . 40
4.2 Summary of known results . . . 42
4.3 `TA` with disequality and implicit HOM equality constraints . . . 48
4.4 Independent sets . . . 51
4.4.1 Independent sets of tuples . . . 51
4.4.2 Independent sets of terms . . . 55
4.5 Constraint-satisfying replacements . . . 56
4.5.1 Equality constraints . . . 56
4.5.2 Classifying disequality constraints . . . 59
4.5.3 Close disequalities . . . 61
4.5.4 Far disequalities . . . 65
4.6 Emptiness decision algorithm . . . 72
4.7 Consequences . . . 78

5 Decidability of global reflexive disequality constraints 87
5.1 `TA` with global constraints 88
5.1.1 Analysis of the expressive power 89
5.2 Intuition on the approach 94
5.3 Compatible runs 96
5.4 Transformation of the automaton 101
5.5 Emptiness and finiteness decision algorithms 107
5.6 Unranked ordered terms 119

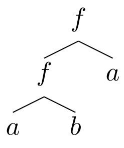
6 Conclusions 123
6.1 Local constraints, and tree homomorphisms 123
6.2 Global constraints 124

Bibliography 127

Chapter 1

Introduction

Computability has been one of the basic fields of study in computer science. It is focused on proving which problems can be solved in an effective manner, and also by what means they can be solved. A crucial aspect of such study is devising methods and machines for performing computations, and then reasoning about their capabilities. One particular kind of machine, called automaton, has proved to be a simple but powerful formalism to define sets (also called languages in this context). The goal of an automaton is to recognize the elements that belong to the language and reject all the rest. Usually, the automaton operates by iteratively reading the input element and, by means of transition rules, updating its state at each step of the execution. At the end, when the whole input has been consumed, the element is considered recognized if the automaton has reached a special final/accepting state, and otherwise, the element is rejected. This idea can be applied to different kinds of elements, the most basic ones being words, i.e., sequences of symbols of an underlying alphabet. This work focuses on a generalization of words into tree-like structures: terms. In this setting, the symbols in the alphabet are used for labeling the nodes of the trees, and moreover, we consider that such symbols have an associated fixed arity (the arity of the symbol labeling a node determines the exact number of children that such node must have). For instance, given an alphabet where the symbol f has arity 2, and the symbols a and b have arity 0, it is possible to define a term like:



Usually, such term is written simply as $f(f(a,b),a)$.

Tree automata (TA) is the essential class of automata for recognizing languages of terms. They characterize the regular tree languages [GS97, CDG$^+$07], and due to their good computational and expressiveness properties, TA have been a well studied formalism. In particular, they have been used to, e.g., describe the parse trees of a context-free grammar or the well-formed terms over a sorted signature [MW67], characterize the solutions of formulas in monadic second-order logic [Don70], or naturally capture type formalisms for tree-structured XML data [MLMK05, BCH+09]. A (bottom-up) TA can be described by a set of rules of the form $f(q_1, \ldots, q_m) \to q$, where $q_1, \ldots, q_m, q$ are state symbols and f is an alphabet symbol of arity m. Additionally, we must specify the set of final states. A term is recognized/accepted by the automaton if, and only if, there is an execution of the automaton on such term (starting from its leaves and proceeding upward to the root) that reaches one of the final states of the automaton.

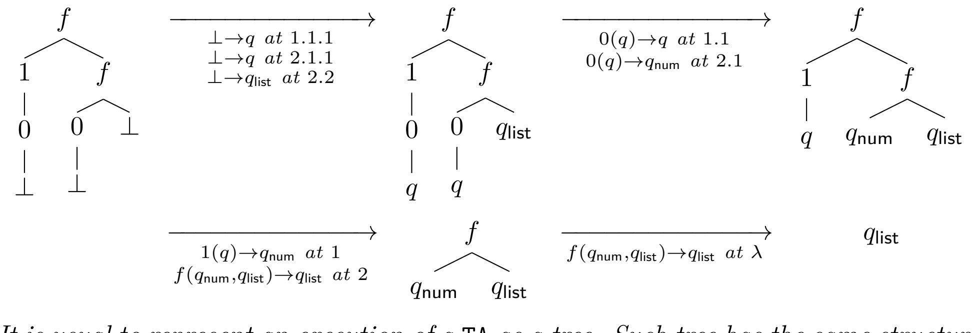
Example 1.1. *Consider the language L of terms that encode lists of binary numbers. More precisely, a symbol f with arity 2 is used to chain the elements in the list like $f(e_1, f(e_2, \ldots f(e_m, \bot) \ldots))$, where the e_i's are the binary numbers in the list and the symbol $\bot$ is used to mark the end of the list. Binary numbers are encoded as terms of the form $b_n(b_{n-1}(\ldots b_1(b_0(\bot)) \ldots))$, where the b_i's are symbols in $\{0,1\}$ and $\bot$ is used in this case to mark the start of a number. A* TA *recognizing L can be constructed as follows. First, it requires transition rules that recognize (non-empty) binary numbers:*

$$\bot \to q \qquad 0(q) \to q \qquad 1(q) \to q$$
$$0(q) \to q_{\mathsf{num}} \qquad 1(q) \to q_{\mathsf{num}}$$

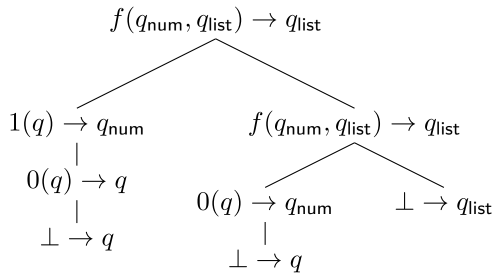
Second, it also needs transition rules that recognize lists of numbers:

$$\bot \to q_{\mathsf{list}} \qquad f(q_{\mathsf{num}}, q_{\mathsf{list}}) \to q_{\mathsf{list}}$$

Finally, the TA *has to specify q_{list} as single final state. Note that there might be several distinct executions of the* TA *on an input term, since there are multiple transition rules with identical left-hand sides (two rules with $0(q)$, two with $1(q)$, and two with $\bot$). Thus, such* TA *is not deterministic. But this is not a problem: it suffices to consider that a term is recognized by the* TA *if there* exists *an execution on the given term that reaches the final state q_{list}, even if there are other possible executions reaching different states. For instance, there is a successful execution on the term $f(e_1, f(e_2, \bot))$, where $e_1 = 1(0(\bot))$ and $e_2 = 0(\bot)$, that proceeds as follows (several transition rules are applied simultaneously at each step in order to shorten the presentation; for clarity, the specific positions where they are applied are made explicit, using λ to denote the root position, 1 for first child, and 2 for second child):*



It is usual to represent an execution of a TA *as a tree. Such tree has the same structure as the input term, and each of its nodes is labeled with the rule applied on the input* *term at the corresponding position. In this way, the previous example execution could be equivalently represented as follows:*


$f(q_{\mathsf{num}}, q_{\mathsf{list}}) \to q_{\mathsf{list}}$
$1(q) \to q_{\mathsf{num}}$
$f(q_{\mathsf{num}}, q_{\mathsf{list}}) \to q_{\mathsf{list}}$
$0(q) \to q$
$0(q) \to q_{\mathsf{num}}$
$\perp \to q_{\mathsf{list}}$
$\perp \to q$
$\perp \to q$

Unfortunately, many natural properties are not captured in the class of regular tree languages. For instance, the language of terms of the form $f(t,t)$, for an arbitrary term t, is a typical example of non-regularity, and the proof for this fact is straightforward: intuitively, it suffices to observe that `TA` have finite memory (the states), and thus, the information that a `TA` can record is not enough to always properly check whether the two children of the root are identical. Due to these limitations, different variations of tree automata that increase the expressive capabilities of plain `TA` have been considered in the literature. We study the case where the automata are enhanced with the possibility to check (dis)equality constraints. In this setting, the automata are able to test whether specific subterms of the input term are equal or different, and only accept such input term when it satisfies all those tests. There are diverse mechanisms for selecting which subterms have to be compared in the evaluation of the constraints, each leading to different expressiveness. We consider the cases where the selection is local (relative to a rule application) and global (over the whole input). The following sections describe these models of automata.

1.1 Local constraints, and tree homomorphisms

In automata with *local* constraints, each transition rule of the automaton has an associated Boolean expression c that restricts its applicability. Usually, the atoms occurring in c are predicates of the form $p_1 \approx p_2$ or $p_1 \not\approx p_2$, for positions p_1 and p_2. Such an atom holds for a given rule application if the subterms pending at p_1 and p_2, relative to the position where the rule is applied, are equal in the first case and different in the second one. The rule can be applied if the entire constraint c holds.

Example 1.2. *Consider again the* `TA` *of Example 1.1. With a slight modification introducing local constraints, we can guarantee that all the (encodings of) numbers occurring in a list are identical, i.e., that the automaton recognizes the language L' of terms of the form $f(e_1, f(e_2, \ldots f(e_m, \perp)\ldots))$ satisfying $e_1 = e_2 = \ldots = e_m$. Note that, e.g., $1(\perp)$ and $0(1(\perp))$ are considered different, even though both of them represent the same natural number 1.*

The precise construction of the automaton with local constraints is as follows. First, we must discard the original transition rule $f(q_{\mathsf{num}}, q_{\mathsf{list}}) \to q_{\mathsf{list}}$, and second, *introduce the following new rules:*

$$f(q_{\mathsf{num}}, q_{\mathsf{list}}) \xrightarrow{1\approx 2.1} q_{\mathsf{list}} \qquad f(q_{\mathsf{num}}, q'_{\mathsf{list}}) \to q_{\mathsf{list}} \qquad \bot \to q'_{\mathsf{list}}$$

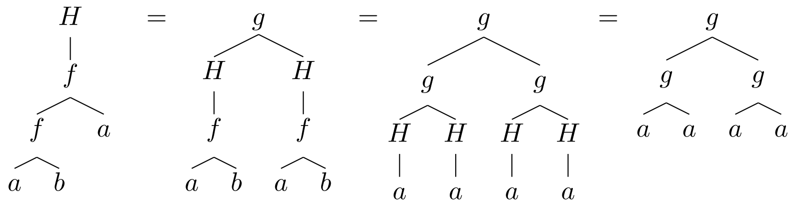
The local constraint $1 \approx 2.1$ *checks that the current number of the list (pending at position* 1*, i.e., first child) is identical to the next number of the list (pending at position* 2.1*, i.e., second child's first child). Only if such condition holds, the transition rule can be applied. Note that* $1 \approx 2.1$ *trivially fails when there is no next number, i.e., when position* 2.1 *does not exist. This can only happen when the second child is* $\bot$*, and thus, such edge case is properly handled by the state* q'_{list} *and the transition rule* $f(q_{\mathsf{num}}, q'_{\mathsf{list}}) \to q_{\mathsf{list}}$*, that performs no local test.*

Unfortunately, the increase in expressiveness obtained with local constraints comes at the expense of other desirable properties, e.g.: emptiness and finiteness of the recognized language are decidable properties for plain TA, but they become undecidable when local constraints are introduced in the automaton model [Mon81]. For this reason, many restrictions on the form of the constraints have been studied in the literature in order to obtain more tractable subclasses of automata. For instance, emptiness and finiteness of the recognized language are decidable for the class AWCBB of tree automata with local constraints where the positions p_1, p_2 involved in each atom have length 1, i.e., where the (dis)equality tests are only performed between brother positions [BT92]. This model of automata allowed to prove decidability of fragments of quantifier-free formulas on one-step rewriting [CSTT99], as well as the recognizability problem for regular tree languages under particular cases of tree homomorphisms [BT92]. Another relevant model is the class RA of tree automata with arbitrary local constraints but with a bound on the maximum number of equality tests that can be performed at each branch of the input term. Emptiness is undecidable for general RA, but it becomes decidable for the fragment of RA of deterministic and complete automata, and this latter result led to the decidability of fragments of the first-order theory of reduction [DCC95]. Additionally, emptiness is decidable in exponential time for the subclass $\mathtt{TA}_{\not\approx}$ of tree automata with only arbitrary local disequality constraints, and this was used to prove EXPTIME-completeness of ground reducibility [CJ03]. Recently, the class $\mathtt{TA}_{\mathsf{hom},\not\approx}$ of tree automata with arbitrary local disequality constraints combined with a restricted version of local equality constraints (called HOM equalities) has been introduced. Emptiness of the language recognized by $\mathtt{TA}_{\mathsf{hom},\not\approx}$ has been shown decidable in triple exponential time, allowing to prove decidability of the general case of the recognizability problem for regular tree languages under tree homomorphisms [GG13].

It is clear from the previously cited literature that local constraints have been a recurrent tool to tackle problems on tree homomorphisms. Since tree homomorphisms play a central role in our work, let us precise their definition and hint how they relate with local constraints. A tree homomorphism H is a special kind of function from terms to terms that can be defined by giving, for each alphabet symbol f with arity n, an equation of the following form:

$$H(f(x_1, \ldots, x_n)) = t$$

where t is a term labeled by either alphabet symbols, or by $H(x_1), \ldots, H(x_n)$, which may only appear at the leaves. Then, the image of a term under the tree homomorphism H is obtained by applying such equations on the term from its root position to the leaves. For instance, given the term $f(f(a,b),a)$, the application of the tree homomorphism H defined by $H(a)=a$, $H(b)=a$, and $H(f(x_1,x_2))=g(H(x_1),H(x_1))$ proceeds recursively as follows:



A tree homomorphism can also be applied to a language. Furthermore, for a language represented by a TA, a representation of its image language under a tree homomorphism can be obtained by transforming directly the TA. Such transformation consists in changing the left-hand side of each transition rule of the TA according to the given tree homomorphism, and adding an equality atom between each two positions where the tree homomorphism has a duplicated variable. For example, the language of all terms over nullaries a,b and binary f can be recognized by a TA with the following transition rules:

$$a \to q \qquad\qquad b \to q \qquad\qquad f(q,q) \to q$$

and by applying the previous H to it, and assuming $H(q)=q$, we obtain the following transition rules:

$$\begin{array}{ccc}
H(a) \to q & H(b) \to q & H(f(q,q)) \to q \\
\rotatebox{-90}{$\rightsquigarrow$} & \rotatebox{-90}{$\rightsquigarrow$} & \rotatebox{-90}{$\rightsquigarrow$} \\
a \to q & a \to q & g(q,q) \xrightarrow{1\approx 2} q
\end{array}$$

where the atom $1 \approx 2$ appears due to the duplication of the variable x_1 in the image of f under H. In this case, the obtained automaton is an AWCBB, i.e., a tree automaton with constraints between brothers [BT92], and as expected by the definition of H and the given TA, the language recognized by the obtained AWCBB is the set of complete trees over nullary a and binary g (which is a classical example of non-regular set). In general, the class AWCBB is not expressive enough to capture the result of such transformation: the resulting left-hand sides of the transition rules are not guaranteed to be of the form $h(q_1,\ldots,q_m)$, where h is an alphabet symbol and $q_1,\ldots,q_m$ are states, and moreover, the generated equality constraints might involve non-brother positions. For instance, if $H(f(x_1,x_2))$ had been defined as $g(g(H(x_1),H(x_2)),g(H(x_2),H(x_1)))$, the transformation of the transition rule $f(q,q) \to q$ would generate a rule with $g(g(q,q),g(q,q))$ as left-hand side, and with an equality constraint between the non-brother positions 1.1 and 2.2, as well as between 1.2 and 2.1. The class $\mathtt{TA}_{\mathsf{hom}}$ introduced in [GG13] properly captures these cases.

The previous examples already make an important property of tree homomorphisms clear: the image of a regular tree language under a tree homomorphism might be a non-regular set. The HOM problem questions, for a given regular tree language L (described by a `TA`) and a tree homomorphism H, whether $H(L)$ is regular. The study of preservation of tree regularity by tree homomorphisms was introduced in [Tha69]. In that paper, tree homomorphisms are defined for the first time, and it is proved that the application of linear tree homomorphisms (i.e., those without duplicated variables) preserves regularity. Tree homomorphisms are also introduced in [Eng75] as a particular case of tree transducers. In [HH92, VG92, KT95], HOM is proved decidable for the particular cases where images are represented as instances of term patterns, or as reducible terms of a term rewrite system. In [BST99], HOM is proved decidable for the particular case of shallow tree homomorphisms. For the same particular case, it is shown in [DTT02] that tree homomorphisms preserving regularity can be assumed to be linear. The HOM problem appears also in [Fül94], where the more general case of regularity of the range of a top-down tree transducer is shown undecidable. In [GMT08], HOM is proved decidable for the particular case where the regular tree language is defined over a monadic signature, and the case where images are represented as instances of term patterns constrained to regular tree languages. This particular case is proved to be EXPTIME-complete in [GGM11]. As a consequence, HOM is EXPTIME-hard. Recently, in [GG13], HOM has been proved decidable in triple exponential time.

1.2 Global constraints

An intrinsic limitation of local constraints is that the (dis)equality tests can only be performed between subterms of the input that are pending at a bounded distance: this is because a local constraint specifies fixed positions (interpreted relative to the position where the rule is to be applied) for the subterms that have to be tested. Recently, in [FTT07, FTT08, FTT10] a new kind of constraints has been proposed that allows to perform (dis)equality tests between subterms that might be arbitrarily faraway. In this new approach, the transition rules of the automata are unconstrained, and it is the automaton itself that has an associated *global* constraint. Such constraint is checked at the end of the computation, and the subterms of the input that are tested for (dis)equality are selected depending on the states to which they are evaluated during such computation. For instance, a global constraint can impose an equality test between all the subterms that are evaluated to a state q, that is, it can force all the subterms evaluated to q to be identical. Such restriction can be specified with the following global constraint:

$$q \approx q$$

Analogously, all subterms evaluated to q can be forced to be pairwise different with:

$$q \not\approx q$$

The precise interpretation of the previous atoms $q \approx q$ and $q \not\approx q$ is as follows: the subterms pending at any two distinct positions evaluated to the state q must be identical in the case $\approx$, and different in the case $\not\approx$.

Atoms of the form $q \approx q$ or $q \not\approx q$ already allow to express properties that cannot be defined by means of local constraints. For instance, reflexive disequality constraints like $q \not\approx q$ correspond to monadic *key constraints*, and thus, can be used to characterize unique identifiers, a typical integrity constraint of XML schemas. Nevertheless, global constraints can be further generalized by considering non-reflexive atoms, that is, atoms that relate different states q_1, q_2 as follows:

$$q_1 \approx q_2$$
$$q_1 \not\approx q_2$$

The interpretation of these atoms is straightforward: each subterm evaluated to state q_1 must be equal (in the case $\approx$) or different (in the case $\not\approx$) to each subterm evaluated to state q_2. Note that, regardless of whether the states q_1, q_2 related in an atom are identical or not, an expression of the form $\neg(q_1 \approx q_2)$ is not equivalent to $q_1 \not\approx q_2$ since an universal quantifier is involved in the interpretation of the atoms, and similarly for $\neg(q_1 \not\approx q_2)$ and $q_1 \approx q_2$. We denote the class of automata with global constraints as $\texttt{TAG}_{\approx,\not\approx}$, where the global constraint is an arbitrary Boolean combination of the previous atomic predicates.

Example 1.3. *Consider again the* TA *of Example 1.1, and the modified language L' proposed in Example 1.2. It is possible to recognize L' by adding to the plain* TA *of Example 1.1 a simple global reflexive constraint: $q_{\mathsf{num}} \approx q_{\mathsf{num}}$. Such constraint is only satisfied when all the (encodings of) numbers occurring in a list are identical.*

With global constraints it is also possible to recognize another interesting variant L'' of L: the language of lists that do not contain repeated (encodings of) numbers, i.e., the language of terms of the form $f(e_1, f(e_2, \ldots f(e_m, \bot) \ldots))$ where the e_i's are pairwise different. It suffices to use $q_{\mathsf{num}} \not\approx q_{\mathsf{num}}$ as the global constraint. It is easy to see that L'' is an example of language that cannot be recognized by automata with local constraints (this can be proved with a simple pumping argument).

As a final remark, note that the interpretations of the reflexive atoms $q \approx q$ and $q \not\approx q$ in $\texttt{TAG}_{\approx,\not\approx}$ only involve (dis)equality tests between subterms of the input that are pending at *distinct* positions. The fact that the positions tested must be distinct implies that no subterm is tested for (dis)equality against itself, which is irrelevant for $q \approx q$ but becomes crucial for $q \not\approx q$: if subterms were tested against themselves, $q \not\approx q$ would only be satisfied by computations that had no occurrence of the state q. In the definition from [FTT10] for tree automata with global constraints (TAGED), subterms are tested against themselves, and hence, it is not possible to define properties analogous to the interpretation of $q \not\approx q$ in $\texttt{TAG}_{\approx,\not\approx}$ (see, e.g., [Vac10, BCG$^+$13]). Even so, TAGED has been a useful formalism to decide a fragment of the spatial logic TQL, and fragments of monadic second order logic on trees extended with predicates for subtree (dis)equality tests [FTT07, FTT08], and the subclass of TAGED with only equality atoms of the form $q \approx q$, called RTA, closed under special kinds of term rewrite systems, has been used to analyze security protocols [JKV11].

1.3 Organization and summary

The remaining of this work is structured as follows. In Chapter 2 we formalize the previous notions of term, tree automata, tree homomorphism, and other related concepts. Chapters 3 to 5 present the studied problems and the corresponding obtained results; in particular, Chapters 3 and 4 focus on local constraints, whereas Chapter 5 deals with global constraints. In Chapter 6 we draw some conclusions and sketch possible avenues of future research.

The contents of Chapters 3 to 5 can be summarized as follows.

Chapter 3

We present our work on automata with local constraints published in:

[CG14] Carles Creus and Guillem Godoy. Tree automata with height constraints between brothers. In *Rewriting Techniques and Applications (RTA)*, pages 149–163, 2014.

We introduce a new kind of automata with local constraints between brother positions that differs from the previous literature in that, instead of testing subterms for either syntactic equality or some notion of equivalence (like, e.g., in [JRV08]), the restrictions are imposed on the *height* of the subterms involved in the constraints. We call them tree automata with height constraints between brothers ($\mathtt{TACBB}_{\mathsf{H}}$). More precisely, our atomic predicates are of the form $h(i) = h(j)$, for positions i and j of length 1, and are satisfied when the subterms pending at i and j, relative to the application of the rule, have identical height. We also consider inequality predicates of the form $h(i) < h(j)$ and comparisons introducing an integer constant x of the form $h(i) = h(j) + x$ and $h(i) < h(j) + x$, with the straightforward interpretations. It is easy to see that our notion of constraints is incomparable with syntactic (dis)equality constraints between brothers, i.e., `AWCBB`. For instance, the language of complete trees over a signature with two nullary symbols a, b and a binary symbol f cannot be recognized by `AWCBB`: intuitively, this is because, even if two terms t_1 and t_2 were inductively guaranteed to be complete, it is not possible to check with only (dis)equality constraints whether t_1 and t_2 have identical height, and thus, whether $f(t_1, t_2)$ is also complete. However, such language can be recognized by a $\mathtt{TACBB}_{\mathsf{H}}$ with the unconstrained rules $a \to q$ and $b \to q$, and the constrained rule:

$$f(q, q) \xrightarrow{h(1)=h(2)} q$$

Another interesting language that can be recognized by $\mathtt{TACBB}_{\mathsf{H}}$ is the language of AVL trees, i.e., the set of trees where the heights of the two direct children of any internal node differ by at most one. It suffices to replace the previous rule for f with:

$$f(q, q) \xrightarrow{\substack{h(1)=h(2)\ \vee \\ h(1)=h(2)+1\ \vee \\ h(1)=h(2)-1}} q$$

Note that the constraint requires the height of each child to be at most one more than the height of the other child. It is easy to see that the language of AVL trees cannot be recognized by the subclass of $\mathtt{TACBB}_{\mathsf{H}}$ whose atoms are restricted to be of the form $h(i) = h(j)$ or $h(i) < h(j)$. Hence, AVL's are an example of the fact that the atoms that introduce an integer constant x of the form $h(i) = h(j) + x$ and $h(i) < h(j) + x$ are strictly more expressive than the simple constraints $h(i) = h(j)$ and $h(i) < h(j)$.

We denote as $\texttt{TACBB}_\mathsf{h}$ the strict subclass of $\texttt{TACBB}_\mathsf{H}$ that only allows atoms of such simple forms.

We tackle the emptiness and finiteness problems for $\texttt{TACBB}_\mathsf{h}$ and $\texttt{TACBB}_\mathsf{H}$, and also for their extensions $\texttt{TACBB}_\mathsf{he}$ and $\texttt{TACBB}_\mathsf{He}$ that include the syntactic (dis)equality constraints between brothers of `AWCBB`. Note that $\texttt{TACBB}_\mathsf{he}$ strictly generalizes $\texttt{TACBB}_\mathsf{h}$ and `AWCBB`, and that $\texttt{TACBB}_\mathsf{He}$ strictly generalizes $\texttt{TACBB}_\mathsf{H}$ and `AWCBB`, since the expressive power of `AWCBB` is incomparable to the expressive powers of $\texttt{TACBB}_\mathsf{h}$ and $\texttt{TACBB}_\mathsf{H}$. To the best of our knowledge, our most general class $\texttt{TACBB}_\mathsf{He}$ has not been studied in the literature. In particular, the definition of `TACT` in [Tre00] is incomparable with $\texttt{TACBB}_\mathsf{He}$, although a given tree automaton with height constraints is transformable into a `TACT` by preserving emptiness (but not the language). Nevertheless, this does not help in our setting to decide emptiness of our model, since emptiness of `TACT` is undecidable. Also, the definition of $\texttt{VTAM}^{R}_{\neg R}$ in [CJP08] captures our automaton models, but emptiness is only decidable for some particular subclasses that are incomparable with $\texttt{TACBB}_\mathsf{He}$ since, even though they can recognize the particular set of complete trees, height of subtrees cannot be compared in general.

Chapter 4

We present an extension of our work on the HOM problem published in:

[CGGR12] Carles Creus, Adrià Gascón, Guillem Godoy, and Lander Ramos. The HOM problem is EXPTIME-complete. In *Logic in Computer Science (LICS)*, pages 255–264, 2012.

Recall that the HOM problem questions, for a given regular tree language L (described by a `TA`) and a tree homomorphism H, whether $H(L)$ is regular. In that publication we develop specific techniques to prove decidability of the HOM problem in exponential time. Here, we present a more general framework and obtain further results based on a new class of tree automata with local constraints. More precisely, we define the class of tree automata with disequality and implicit HOM equality constraints ($\texttt{TA}_{\mathsf{ihom},\not\approx}$). This class allows arbitrary local disequality constraints and a particular kind of equality constraints: the left-hand side of the transition rules are terms containing states at some leaf positions, and two positions with the same state implicitly define a local equality constraint between such positions. For example, consider the language of terms of the form $h(t_1, t_2)$, where t_1, t_2 are different complete trees over nullary a and binary g. Such language can be recognized by a $\texttt{TA}_{\mathsf{ihom},\not\approx}$ with the following transition rules:

$$\begin{array}{lll} a \to q & g(q,q) \to q & h(q,q') \xrightarrow{1\not\approx 2} q_{\mathsf{accept}} \\ a \to q' & g(q,q) \to q' & \end{array}$$

where q_{accept} is the only final state. Note that equality constraints are implicitly represented by the fact that the state q appears duplicated at positions 1 and 2 in the left-hand sides of the rules for g. Note also that the rule for h has two distinct states q and q' at its left-hand side, since we do not require that its children are equal (in fact, the constraint $1 \not\approx 2$ forces them to be different, as desired). This particular example does not show the full expressiveness of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$, and in fact, could have been equivalently defined as an AWCBB. In general, however, this is not possible since $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ are strictly more expressive than AWCBB: the class of languages recognizable by AWCBB is included in the class of languages recognizable by $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ since any AWCBB can be transformed into a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ recognizing the same language, and it is easy to prove that such inclusion is strict using the fact that $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ can define (dis)equality tests between non-brother positions. At first look, the transformation from AWCBB into $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ might seem surprising, since equalities in $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ can only be tested between positions reaching the same state, whereas in AWCBB equalities can be tested between any two brother positions, regardless of the states reached at them. Nevertheless, AWCBB can be determinized [BT92], and thus, the expressiveness of AWCBB is not reduced when restricting its equality constraints to only involve brother positions reaching the same state.

We tackle the HOM problem by building on the results from [GG13] and reasoning on $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. In that paper, HOM is reduced to the emptiness problem of a special kind of automata with local constraints (a class equivalent to $\mathtt{TA}_{\mathsf{ihom},\not\approx}$, but technically more complex), and this latter problem is shown decidable in triple exponential time. We prove decidability of the emptiness and finiteness problems for $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ in exponential time, and in this way, we lower the classification of HOM from 3EXPTIME in [GG13] to EXPTIME. This result is tight, since HOM is known to be EXPTIME-hard [GGM11]. The techniques developed in [GG13] also allow to reduce other problems to the emptiness of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$, like set inclusion and finiteness of set difference for sets defined as images of regular tree languages under tree homomorphisms. Hence, we also improve the classification of these problems from 3EXPTIME in [GG13] to EXPTIME, which is again tight [CDG+07]. A similar result is obtained in the setting of term rewriting, since the set of reducible terms of a term rewrite system can be described as the image of a tree homomorphism. In particular, we prove that inclusion of sets of normal forms of term rewrite systems can be decided in exponential time.

Chapter 5

We present our work on automata with global constraints published in:

[CGG13] Carles Creus, Adrià Gascón, and Guillem Godoy. Emptiness and finiteness for tree automata with global reflexive disequality constraints. *Journal of Automated Reasoning*, 51(4):371–400, 2013.

We focus on tree automata with a particular kind of global constraints: reflexive disequalities ($\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$). More precisely, we consider the model where the global constraints are restricted to be conjunctions of positive literals of the form $q \not\approx q$, or of the form $q_1 \not\approx q_2$ provided that the atoms $q_1 \not\approx q_1$ and $q_2 \not\approx q_2$ also occur in the constraint. In other words, the global constraint of a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ can alternatively be seen as a relation on a subset of its states (i.e., on those states occurring in the constraint) that is symmetric (since atoms are considered unordered pairs) and reflexive (due to the previous conditions on the form of the global constraints of $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$). These automata are significant in the context of XML definitions since they can characterize monadic key constraints: in this setting, a state q of a given automaton corresponds to an XML type, i.e., it defines a set of valid values, and a reflexive disequality constraint like $q \not\approx q$ specifies that all values of type q within an XML document must be distinct for that document to be valid.

We tackle the emptiness and finiteness problems for $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$. Note that decidability results for several fragments of tree automata with global constraints can be found in the literature: emptiness of the subclass where the global constraint is a conjunction of atoms over the predicates $\approx$ and $\not\approx$ such that $\approx$ defines a reflexive relation and $\not\approx$ defines an anti-reflexive relation is in NEXPTIME [Vac10], which coincides with the best known complexity for deciding emptiness of the subclass where the global constraint is a conjunction of atoms over the predicate $\not\approx$ such that $\not\approx$ defines an anti-reflexive relation [FTT08, FTT10]; emptiness and finiteness for the subclass where the global constraint is a conjunction of atoms over the predicate $\approx$ are EXPTIME-complete [FTT08], which still holds when restricting the constraints to just two atoms but decreases to PTIME when only a single atom is allowed [HHK12]; and emptiness and finiteness for the subclass $\mathtt{RTA}$ where the global constraint is a conjunction of atoms of the form $q \approx q$ are in PTIME, emptiness being actually decidable in linear time [JKV11]. Nevertheless, all the previous subclasses of automata with global constraints are incomparable with $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$, and thus far, the only known decidability result that captures our model $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ is presented in [Vac10, BCG^{+}13] for the full $\mathtt{TAG}_{\approx,\not\approx}$ class: emptiness is proved decidable in non-elementary time, and decidability of finiteness is left as an open question. We improve such results for the fragment $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ by proving that emptiness and finiteness for $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ are in 3EXPTIME.

Chapter 2

Basic concepts and notations

Here we introduce background concepts that are required in the following chapters. We focus on providing essential definitions; for a complete and detailed survey the reader is referred to [CDG⁺07] and [BN98].

2.1 General notation on sets and complexity classes

We fix some basic notations for sets. The *size* of a finite set S is denoted by $|S|$, and the *powerset* of S by 2^S. Given two disjoint sets A and B, we sometimes denote their union as $A \uplus B$ in order to emphasize the fact that A and B are disjoint. In some cases, we may simply define A and B as sets, and use $A \uplus B$ to implicitly state that they are disjoint.

A *partition* P of a set S is a set of non-empty sets $P_1, \dots, P_n$ satisfying that they are pairwise disjoint and $S = P_1 \cup \ldots \cup P_n$. Each P_i is said to be a *part* of the partition. We use the notation $\bigcup P$ as shorthand for $\bigcup_{i \in \{1,\dots,n\}} P_i$, that is, $S = \bigcup P$. The partition P induces an *equivalence relation* on $\bigcup P$, which we denote as $\sim_P$ and define as follows: $e \sim_P e'$ if and only if e and e' belong to the same part in P.

We use the *Landau notation* when arguing about the time and space complexity of the presented algorithms. That is, given a function $g : \mathbb{N} \to \mathbb{R}$, we denote by $\mathcal{O}(g)$ the class of all functions whose asymptotic growth rate is bounded by g. Formally:

$$\mathcal{O}(g) = \{f : \mathbb{N} \to \mathbb{R} \mid \exists c > 0 \; \exists n_0 \in \mathbb{N} \; \forall n \geq n_0 : |f(n)| \leq c \cdot |g(n)|\}$$

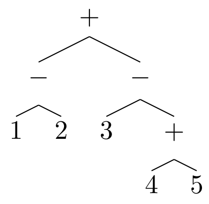
However, many of our algorithms belong to exponential complexity classes, and the standard $\mathcal{O}$-notation is not convenient enough in such setting. In particular, we are interested in abstracting away the specific base of exponential functions, or equivalently, ignore any constant factor in the exponents. Such abstractions are not easy with the standard $\mathcal{O}$-notation (consider, e.g., $\mathcal{O}(2^n) \subsetneq \mathcal{O}(3^n) \subsetneq \mathcal{O}(2^{2 \cdot n})$). For this reason, we adopt the usual notation $2^{\mathcal{O}(g)}$ to denote the following set of functions:

$$2^{\mathcal{O}(g)} = \{f : \mathbb{N} \to \mathbb{R} \mid \exists c > 0 \; \exists n_0 \in \mathbb{N} \; \forall n \geq n_0 : |f(n)| \leq 2^{c \cdot |g(n)|}\}$$

Note that, e.g., $2^{\mathcal{O}(n)}$ includes all functions of the form $a^{b \cdot n+c}$, for any constant numbers a, b, and c. This notation is straightforwardly generalized to higher exponential classes $2^{2^{\mathcal{O}(g)}}$, $2^{2^{2^{\mathcal{O}(g)}}}$, and so on.

2.2 Terms

A term is a generalization of a word into a tree structure, that is, a term is a tree where nodes are labeled by symbols of a given alphabet Σ. In order to illustrate this definition, consider the alphabet $\Sigma = \{+, -, 1, \ldots, 9\}$. It is clear that in the context of words, it is possible to represent with Σ operations of addition and subtraction of natural numbers, e.g., the word $1 - 2 + 3 - 4 + 5$. However, note that without the use of parentheses it is not possible to determine from such word the order of the operations. In the setting of terms, this order is given by the structure of the tree. For example, in the following term it is clear how the expression has to be evaluated:



The previous generalization from words to terms is rather broad, and leads in many cases to the intractability of decisional problems on tree languages. For this reason, it is usual to consider some extra conditions restricting the definition of term. We take the following three:

- We assume that terms are finite.
- We assume that the alphabet Σ is coupled with a total function $\mathsf{arity} : \Sigma \to \mathbb{N}$ that assigns to each of the symbols in the alphabet an *arity*. In such case, Σ is called a *signature*, and a node of a tree labeled by a symbol with arity n must have exactly n children. Clearly, it is necessary that there exists at least one symbol whose arity is 0, since otherwise, it would be impossible to construct any valid finite term.
- We assume that the children of a node of the tree have an order. In general, trees (and graphs) are unordered in the sense that there is no relation between the edges connecting a node to other nodes. In an ordered tree, we have an order for the edges that allows us to specifically refer to the i'th child of a node.

Combining these properties, we obtain the class of *finite ranked ordered trees*, which from this point on we simply call terms. The following two definitions formalize the notion of signature, and the set of all possible terms over a given signature.

Definition 2.1. *A* signature Σ *is a finite set of alphabet symbols with arity, which is partitioned as* $\bigcup_{i \in \mathbb{N}} \Sigma^{(i)}$ *satisfying that* $f \in \Sigma^{(m)}$ *if the arity of the alphabet symbol*

f is m. We sometimes denote Σ as $\{f_1{:}m_1,\dots,f_n{:}m_n\}$, where $f_1,\dots,f_n$ are the alphabet symbols and $m_1,\dots,m_n$ are the corresponding arities. We define $\mathsf{maxar}(\Sigma)$ *as* $\max\{m_1,\dots,m_n\}$*, and write simply* maxar *when Σ is known from the context. We denote by* $\mathsf{arity}(f)$ *the arity of symbol f, and say that f is a* constant/nullary *when* $\mathsf{arity}(f) = 0$.

Definition 2.2. *Let Σ be a signature. The* set of all terms *over Σ is defined recursively as $\mathcal{T}(\Sigma) = \{f(t_1,\dots,t_m) \mid f \in \Sigma^{(m)} \wedge t_1,\dots,t_m \in \mathcal{T}(\Sigma)\}$.*

Note that the base case of the previous recursive definition corresponds to the terms of the form $a()$, where $a \in \Sigma^{(0)}$. For such terms $a()$ we simply write a. Additionally, it is usual to denote terms of the form $g(\dots(g(a))\dots)$ with n occurrences of the unary symbol g simply as $g^n(a)$.

In some contexts, terms can contain nodes that are not labeled by symbols of the underlying signature, and instead are labeled by variables ranging over terms. Note that, since we consider that the range of the variables are terms, it is necessary that variables only occur at the leaves, i.e., that they behave as nullary symbols. In order to define the set of terms containing variables, we fix the set $\mathcal{X} = \{x_1, x_2, \dots\}$ of variables, and assume without loss of generality that $\mathcal{X}$ is disjoint from any given signature Σ. Then, we can refine Definition 2.2 as follows.

Definition 2.3. *Let Σ be a signature. The set $\mathcal{T}(\Sigma,\mathcal{X})$ of all terms over Σ and $\mathcal{X}$ is the smallest set containing $\mathcal{X}$ and such that $f(t_1,\dots,t_m)$ is in $\mathcal{T}(\Sigma,\mathcal{X})$ whenever $f \in \Sigma^{(m)}$ and $t_1,\dots,t_m \in \mathcal{T}(\Sigma,\mathcal{X})$.*

Terms that contain no variable, i.e., terms in $\mathcal{T}(\Sigma)$, are said to be ground*. A* language *over Σ is a set of ground terms.*

2.3 Positions

One of the benefits of dealing with ordered trees is that we can easily refer to specific nodes within a term. For instance, in the example of term presented at the beginning of Section 2.2, the node labeled by the symbol 2 can be unambiguously identified as the second child of the first child of the root node. In general, the position of a node within a term can be identified by a sequence of natural numbers as follows:

- The root node is identified with the empty sequence.
- The i'th child of a node at position p is identified with the position p concatenated with i.

Using this idea, we can formally define position and the set of positions of a term.

Definition 2.4. *A* position *is a sequence of natural numbers. The* empty sequence *is denoted by λ, the symbol . is used to denote the* concatenation *of two positions, and the* length *of a position p is denoted by $|p|$. Note that $|\lambda| = 0$ and $|i.p| = 1 + |p|$, where i is a natural number.*

Given a term $f(t_1,\dots,t_m)$, its set of positions *is defined as* $\mathsf{Pos}(f(t_1,\dots,t_m)) = \{\lambda\} \uplus \{i.p \mid i \in \{1,\dots,m\} \wedge p \in \mathsf{Pos}(t_i)\}$*. Given a set S and a term t, we use* $\mathsf{Pos}_S(t)$ *to denote the set of positions of t that are labeled by symbols in S.*

Reasoning about the relationship between nodes of a term is simplified by instead dealing with positions, and by using the following prefix relation.

Definition 2.5. *Given two positions p_1 and p_2, we say that p_1 is a* prefix *of p_2, denoted $p_1 \leq p_2$, if p_2 can be written of the form $p_1.p'$ for some position p'. Moreover, if p' is not λ, then p_1 is said to be a* strict prefix *of p_2, denoted $p_1 < p_2$. We define $p_2 - p_1$ as the suffix p'. When p_1 and p_2 are incomparable with respect to this prefix relation, i.e., when $p_1 \not\leq p_2$ and $p_2 \not\leq p_1$, they are said to be* parallel, *denoted $p_1 \parallel p_2$.*

2.4 Functions and operations on terms

The *size* of a term is its number of nodes, and its *height* is the length of the longest path from the root node to a leaf. Both concepts can easily be defined by means of positions as follows.

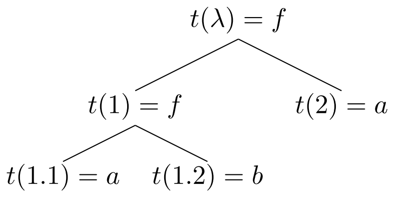
Definition 2.6. *Let t be a term. The* size *of t, denoted $|t|$, is $|\mathsf{Pos}(t)|$, and the* height *of t, denoted $\mathsf{height}(t)$, is $\max\{|p| : p \in \mathsf{Pos}(t)\}$. We say that t is* flat *if $\mathsf{height}(t) \leq 1$.*

We perform three basic operations on terms. First, we want to refer to the *symbol labeling* a concrete position of a term. Second, we want to refer to a specific *subterm* of a term by means of the position where it occurs. Third, we want to *replace* a specific subterm by a new term. These operations are naturally defined using a recursive formulation as follows.

Definition 2.7. *Let $t = f(t_1, \ldots, t_m)$ be a term. Let $p \in \mathsf{Pos}(t)$ be a position. Then:*

- *The* symbol labeling *t at position p, denoted $t(p)$, is defined as f when $p = \lambda$, and as $t_i(p')$ when $p = i.p'$.*
- *The* subterm *of t at position p, denoted $t|_p$, is defined as t when $p = \lambda$, and as $t_i|_{p'}$ when $p = i.p'$. We say that the subterm is* strict *if $p \neq \lambda$.*
- *The* replacement *of the subterm of t at position p by a given term s, denoted $t[s]_p$, is defined as s when $p = \lambda$, and as $f(t_1, \ldots, t_{i-1}, t_i[s]_{p'}, t_{i+1}, \ldots, t_m)$ when $p = i.p'$.*

Example 2.8. *Consider the term $t = f(f(a, b), a)$ over the nullaries a, b and the binary f. For all position $p \in \mathsf{Pos}(t)$, we depict $t(p)$ using the tree structure of t as follows:*



The subterm of t at position 1, i.e., the subterm $t|_1$, is $f(a, b)$. If we are given another term $s = a$, then we can replace the subterm $t|_1$ by s, i.e., perform the replacement $t[s]_1$, and in this way obtain as result the term $f(a, a)$.

2.5 Tree automata

The most essential class of tree languages defined by tree recognizers is composed of the so-called *regular tree languages*. These languages are characterized by means of automata that only have finite memory and whose transitions only depend on local information. We assume that the reader knows the Boolean closure properties and the decidability results on regular tree languages [GS84, GS97, CDG$^+$07]. Here we only recall the notion of tree automata.

Definition 2.9. *A (bottom-up)* tree automaton*, or* TA *for short, is a* 4*-tuple* $A = \langle Q, \Sigma, F, \Delta \rangle$*, where* Q *is a finite set of states,* Σ *is a signature,* $F \subseteq Q$ *is the subset of final states, and* Δ *is a set of transition rules of the form* $f(q_1, \ldots, q_m) \to q$*, where* $q_1, \ldots, q_m, q \in Q$ *and* $f \in \Sigma^{(m)}$*.*

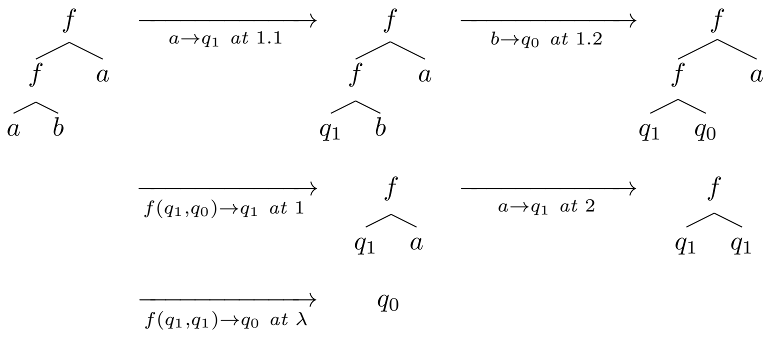
The size *of* A*, denoted* $|A|$*, is* $|Q|$ *plus the sum of sizes of all rules in* Δ*, where the size of a rule of the form* $f(q_1, \ldots, q_m) \to q$ *is* $m + 2$*.*

Note that the notion of size of TA ignores the final state set F and the signature Σ. This is because, when reasoning asymptotically with the size of a TA, they are usually considered irrelevant. In particular, $|F| \leq |Q|$ holds, and it is usual to assume that $|\Sigma|$ and $\mathsf{maxar}(\Sigma)$ are bounded by the sum of sizes of the rules in Δ (otherwise, it must be the case that some symbol in Σ does not occur in any of the rules in Δ, and thus such symbol is useless and could be discarded).

Example 2.10. *Consider the* TA A *described by the following set of rules:*

$$\begin{aligned} a &\to q_1 \\ b &\to q_0 \\ f(q_0, q_0) &\to q_0 \\ f(q_0, q_1) &\to q_1 \\ f(q_1, q_0) &\to q_1 \\ f(q_1, q_1) &\to q_0 \end{aligned}$$

where q_0 *is the only final state. The automaton* A *runs bottom-up on an input term like* $f(f(a, b), a)$ *by successively applying its rules:*



Note that the automaton reaches the state q_0 *at the root position of the input term. Since such state is final, the term* $f(f(a, b), a)$ *is considered to be in the language recognized by* A*. It is easy to see that the language recognized by* A *is the set of terms over binary* f *and nullaries* a, b *with an even number of* a*'s.*

The formal definition of run of an automaton, recognized language, and regularity is as follows.

Definition 2.11. *A* run *of a* `TA` $A = \langle Q, \Sigma, F, \Delta \rangle$ *on a term* $t \in \mathcal{T}(\Sigma)$ *is a mapping* $r : \mathsf{Pos}(t) \to \Delta$ *satisfying that, for each position* $p \in \mathsf{Pos}(t)$, *if* $t|_p$ *is of the form* $f(t_1, \ldots, t_m)$, *then* $r(p)$ *is a rule of the form* $f(q_1, \ldots, q_m) \to q$, *and* $r(p.1), \ldots, r(p.m)$ *are rules with right-hand sides* $q_1, \ldots, q_m$, *respectively. We say that* $r(p)$ *is the* rule applied *at position* p. *The* state reached *by* r *is the right-hand side of* $r(\lambda)$. *The run* r *is called* accepting *if it reaches a state in* F.

A term t *is* accepted/recognized *by* A *if there exists an accepting run of* A *on* t. *The* language recognized *by* A, *denoted* $\mathcal{L}(A)$, *is the set of terms accepted by* A. *By* $\mathcal{L}(A, q)$ *we denote the set of terms for which there exists a run of* A *on them reaching the state* q. *A language* L *is* regular *if there exists a* `TA` A *such that* $\mathcal{L}(A) = L$.

In many occasions, we will need to refer to the state occurring at the right-hand side of a rule. For this reason, we introduce the following notation.

Definition 2.12. *Given a rule* $f(q_1, \ldots, q_m) \to q$, *we define* $\mathsf{rhs}(f(q_1, \ldots, q_m) \to q)$ *as* q.

As a final remark, it is usual to treat runs as terms over the signature Δ, where a rule of the form $f(q_1, \ldots, q_m) \to q$ in Δ is treated as a symbol with arity m. The run on the term $f(f(a, b), a)$ presented in Example 2.10 can be represented as follows in the interpretation of runs as terms:

$$\begin{tikzcd}
 & & f(q_1,q_1)\to q_0 \arrow[dl, dash] \arrow[dr, dash] & \\
 & f(q_1,q_0)\to q_1 \arrow[dl, dash] \arrow[dr, dash] & & a\to q_1 \\
a\to q_1 & & b\to q_0 &
\end{tikzcd}$$

Note that the run provides enough information to deduce the term that it recognizes. For this reason, in many cases we just define a run and leave implicit the term being recognized. Furthermore, in order to simplify the presentation, we straightforwardly adapt notations on term to runs as follows.

Definition 2.13. *Let* r *be a run of a* `TA` A *on a term* t. *We define* $\mathsf{term}(r)$ *as* t, $\mathsf{Pos}(r)$ *as* $\mathsf{Pos}(t)$, *and* $\mathsf{height}(r)$ *as* $\mathsf{height}(t)$. *Given a position* $p \in \mathsf{Pos}(r)$, *the* subrun $r|_p$ *is the run of* A *on* $t|_p$ *described by* $r|_p(p') = r(p.p')$. *Moreover, we say that the subrun is* strict *if* $p \neq \lambda$. *Given two runs* r_1, r_2 *of* A, *and a position* $p \in \mathsf{Pos}(r_1)$ *such that* $r_1|_p$ *and* r_2 *reach the same state, the* replacement $r_1[r_2]_p$ *is the run* r *of* A *on* $\mathsf{term}(r_1)[\mathsf{term}(r_2)]_p$ *defined as follows:* $r(p') = r_2(p' - p)$ *if* $p \leq p'$, *and* $r(p') = r_1(p')$ *otherwise.*

2.6 Substitutions, tree homomorphisms, rewriting

Tree homomorphisms can be defined as functions commuting with a natural tree operator: application of *substitution*. However, such definition is not very intuitive, and thus, in Definition 2.15 we present them using a recursive formalization.

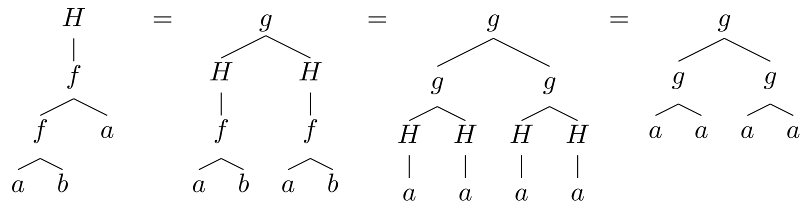
Definition 2.14. *A* substitution *σ is a mapping from variables to terms. Usually, a substitution is written as a finite set of pairs $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, where each x_i is a variable in $\mathcal{X}$ and each t_i is a term in $\mathcal{T}(\Sigma, \mathcal{X})$. Substitutions can be homomorphically extended to functions from terms to terms as follows: given a substitution σ and a term t, by $\sigma(t)$ we denote the result of simultaneously replacing in t every $x \in \mathsf{Dom}(\sigma)$ by $\sigma(x)$. For example, if σ is $\{x \mapsto f(b, y), y \mapsto a\}$, then $\sigma(g(x, y))$ is $g(f(b, y), a)$.*

Definition 2.15. *Let Σ_1, Σ_2 be two signatures. A* tree homomorphism *is a function $H : \mathcal{T}(\Sigma_1) \to \mathcal{T}(\Sigma_2)$ which can be defined as follows.*

Let $\mathcal{X}_m$ represent the set of variables $\{x_1, \ldots, x_m\}$ for each natural number m. The definition of $H : \mathcal{T}(\Sigma_1) \to \mathcal{T}(\Sigma_2)$ requires to define $H(f(x_1, \ldots, x_m))$ for each alphabet symbol $f \in \Sigma_1$ of arity m as a term $t_f \in \mathcal{T}(\Sigma_2, \mathcal{X}_m)$. Then, $H(f(t_1, \ldots, t_m))$ is defined for each term $f(t_1, \ldots, t_m) \in \mathcal{T}(\Sigma_1)$ as $\{x_1 \mapsto H(t_1), \ldots, x_m \mapsto H(t_m)\}(t_f)$.

The size *of H, denoted $|H|$, is the sum of the sizes of all the terms t_f.*

Example 2.16. *Consider the signatures $\Sigma_1 = \{a{:}0, b{:}0, f{:}2\}$, $\Sigma_2 = \{a{:}0, g{:}2\}$. Now consider the tree homomorphism $H : \mathcal{T}(\Sigma_1) \to \mathcal{T}(\Sigma_2)$ defined by $H(a) = a$, $H(b) = a$, and $H(f(x_1, x_2)) = g(x_1, x_1)$. The recursive application of H on the term $f(f(a, b), a)$ proceeds as follows:*



We conclude by introducing the concept of *term rewriting*. Intuitively, a rewrite rule $l \to r$ specifies a way to modify terms: if some subterm of a term t matches the term l, then such subterm can be replaced by the term r (both l and r might contain variables).

Definition 2.17. *A* rewrite rule *is a pair of terms $l \to r$. The application of a rewrite rule $l \to r$ to a term $s[\sigma(l)]_p$ at position p produces the term $s[\sigma(r)]_p$. A* term rewrite system *R is a set of rewrite rules. The application of a rule of R to a term s resulting into a term t is denoted by $s \to_R t$, the transitive closure of this relation is denoted by $\to_R^+$, and the reflexive-transitive closure by $\to_R^*$.*

Chapter 3

Decidability of height and equality constraints between brothers

In this chapter we focus on automata with height constraints ($\texttt{TACBB}_{\textsf{H}}$). Recall that this model of automaton extends TA by adding local constraints to the transition rules, and that such constraints compare the heights of brother subterms. More precisely, the atomic constraints in the transition rules of $\texttt{TACBB}_{\textsf{H}}$ are expressions of the form $h(i) = h(j) + x$ or $h(i) < h(j) + x$, where i, j are positions of length 1 and x is an integer constant. Such atoms are satisfied if the expressions evaluate to true after replacing $h(i)$ and $h(j)$ by the heights of the i'th and j'th child, respectively, relative to the position of the input term where the constrained rule is to be applied. We prove decidability of the emptiness and finiteness problems for $\texttt{TACBB}_{\textsf{H}}$. Our approach consists in transforming the automaton into a normalized form, and obtaining a recursive formulation to describe the set of reachable states when recognizing terms of a specific height. The decision algorithm follows directly from such result. We also adapt the method to automata ($\texttt{TACBB}_{\textsf{He}}$) that combine the height constraints of $\texttt{TACBB}_{\textsf{H}}$ with the (dis)equality constraints between brothers of AWCBB from [BT92].

The remaining of this chapter is structured as follows. In Section 3.1 we formally define the class of automata with height constraints. In Section 3.2 we tackle the subclass of $\texttt{TACBB}_{\textsf{H}}$ where only simple height constraints are allowed, i.e., where the integer constant x of all atomic predicates is 0 (thus, the atoms are of the form $h(i) = h(j) + 0$ or $h(i) < h(j) + 0$, which we write as $h(i) = h(j)$ and $h(i) < h(j)$, respectively). We call such subclass $\texttt{TACBB}_{\textsf{h}}$. This section is intended as intuition for the general case, i.e., for $\texttt{TACBB}_{\textsf{He}}$, since both approaches are analogous. The general case is tackled in Section 3.3, where we also analyse the time complexities of the decision algorithms for $\texttt{TACBB}_{\textsf{H}}$, and for a class $\texttt{TACBB}_{\textsf{he}}$ that mixes the simple height constraints of $\texttt{TACBB}_{\textsf{h}}$ with the (dis)equality constraints of AWCBB.

3.1 TA with height and equality constraints between brothers

In this section we give a general definition of tree automata with constraints between brothers. This definition is instantiated later, by considering distinct forms and interpretations for the constraints. To simplify notations, along this chapter we consider a fixed signature Σ.

Definition 3.1. *A* constraint structure *is a tuple* $\langle C, \mathsf{PosVar}, \models, |.|, \mathsf{Width}\rangle$*. Here,* C *(notion of syntax) is a set of elements called constraints.* PosVar *is a function that maps each element of* C *to a finite set of natural numbers. Given* $c \in C$*, any partial mapping* $I : \mathbb{N} \to \mathcal{T}(\Sigma)$ *satisfying* $\mathsf{PosVar}(c) \subseteq \mathsf{Dom}(I)$ *is called an* interpretation *of* c*.* $\models$ *(notion of satisfaction) maps each pair* $c \in C$ *and interpretation* I *of* c *to either true (denoted* $I \models c$*) or false (denoted* $I \not\models c$*).* $|.|$ *(notion of size) maps each constraint* c *to a natural number.* Width *maps each constraint* c *to a natural number.*

Given two constraints c_1, c_2 *(not necessarily from the same constraint structure) satisfying* $\mathsf{PosVar}(c_1) = \mathsf{PosVar}(c_2)$*, we say that* c_1, c_2 *are* compatible *if there exists* I *satisfying* $I \models c_1$ *and* $I \models c_2$ *(where* $\models$ *refers to the satisfaction notion of each corresponding structure). Otherwise, we say that* c_1, c_2 *are* incompatible.

A set of constraints $\{c_1, \ldots, c_n\}$ *of the same constraint structure and satisfying* $\mathsf{PosVar}(c_1) = \cdots = \mathsf{PosVar}(c_n)$ *is called* deterministic *if all the constraints are pairwise incompatible, and it is called* complete *if, for each* $I : \mathsf{PosVar}(c_1) \to \mathcal{T}(\Sigma)$*, there is some* c_i *such that* $I \models c_i$*.*

Given two constraints c_1, c_2 *(not necessarily from the same constraint structure) satisfying* $\mathsf{PosVar}(c_1) \supseteq \mathsf{PosVar}(c_2)$*, we say that* c_1 implies c_2*, denoted* $c_1 \models c_2$*, if* $(I \models c_1) \Rightarrow (I \models c_2)$ *for each interpretation* I *of* c_1 *(where* $\models$ *refers to the satisfaction notion of each corresponding structure).*

Definition 3.2. *Let* $\mathsf{S} = \langle C, \mathsf{PosVar}, \models, |.|, \mathsf{Width}\rangle$ *be a constraint structure. A* tree automaton with constraints between brothers *based on* S*,* $\mathtt{TACBB}_{\mathsf{S}}$ *for short, is a tuple* $A = \langle Q, \Sigma, F, \Delta\rangle$*, where* Q *is a finite set of states,* Σ *is a signature,* $F \subseteq Q$ *is the subset of final states, and* Δ *is a finite set of rules of the form* $f(q_1, \ldots, q_m) \xrightarrow{c} q$*, where* $q_1, \ldots, q_m, q \in Q$*,* f *is a symbol in* Σ *of arity* m*, and* c*, called the* (local) constraint of the rule*, is a constraint of* C *satisfying* $\mathsf{PosVar}(c) \subseteq \{1, \ldots, m\}$*. The rule is* fully constrained *if* $\mathsf{PosVar}(c) = \{1, \ldots, m\}$*. The* size *of such rule is* $m+2+|c|$*, and its* width *is* $\mathsf{Width}(c)$*. The* size *of* A*, denoted* $|A|$*, is* $|Q|$ *plus the sum of sizes of all its rules, and the* width *of* A *is the maximum between* 1 *and the widths of its rules. We say that* A *is* fully constrained*, denoted* $\mathtt{TACBB}^{\mathsf{F}}_{\mathsf{S}}$*, if each of its rules is fully constrained.*

A run *of* A *on a term* $t \in \mathcal{T}(\Sigma)$ *is a mapping* $r : \mathsf{Pos}(t) \to \Delta$ *such that, for each position* $p \in \mathsf{Pos}(t)$*, if* $t|_p$ *is of the form* $f(t_1, \ldots, t_m)$*, then* $r(p)$ *is a rule of the form* $f(q_1, \ldots, q_m) \xrightarrow{c} q$*, the rules* $r(p.1), \ldots, r(p.m)$ *have* $q_1, \ldots, q_m$ *as right-hand sides, respectively, and* $I = \{1 \mapsto t|_{p.1}, \ldots, m \mapsto t|_{p.m}\} \models c$*. We say that* $r(p)$ *is the* rule applied *at position* p*. The* state reached *by* r *is the right-hand side of* $r(\lambda)$*. The run* r *is called* accepting *if it reaches a state in* F*. A term* t *is* accepted/recognized *by* A *if there exists an accepting run of* A *on* t*. The* language recognized *by* A*, denoted*

$\mathcal{L}(A)$, is the set of terms accepted by A. By $\mathcal{L}(A, q)$ we denote the set of terms for which there exists a run of A on them reaching q.

Definition 3.3. *A* height and equality constraint *c is a Boolean combination (including negation) of atoms of the form either $h(i) = h(j) + x$ or $h(i) < h(j) + x$ or $i \approx j$ or $i \not\approx j$ for distinct natural numbers i, j and integer number x. The* width *of c, denoted $\mathsf{Width}(c)$, is 1 plus the maximum of the absolute values of such integers x. The* size *of c, denoted $|c|$, is the number of such atoms occurring in c. By $\mathsf{PosVar}(c)$ we denote the set of such naturals i, j occurring in c. An* interpretation *of c is a partial mapping $I : \mathbb{N} \to \mathcal{T}(\Sigma)$ such that $\mathsf{PosVar}(c) \subseteq \mathsf{Dom}(I)$, and we say that I* satisfies *(*is a solution of*) c, denoted $I \models c$, if, by replacing in c each natural i by the term $I(i)$, the expression evaluates to true by interpreting h as the* height *function, $\approx$ and $\not\approx$ as syntactic equality and disequality between terms, respectively, $=$ and $<$ as respectively equality and less of the evaluated expressions on integers (where $+$ is the addition operator on integers), and the Boolean operators in the usual way.*

By C_{He} we denote the set of all height and equality constraints. By C_H we denote the subset of C_{He} of constraints with only atoms of the form $h(i) = h(j) + x$ or $h(i) < h(j) + x$ (i.e., only height constraints). By C_{he} we denote the subset of C_{He} of constraints with only atoms of the form $h(i) = h(j) + 0$ or $h(i) < h(j) + 0$ or $i \approx j$ or $i \not\approx j$ (i.e., equality constraints and simple height constraints, that is, height constraints where the integer constant is always 0). By C_h we denote the subset of C_{He} of constraints with only atoms of the form $h(i) = h(j)+0$ or $h(i) < h(j)+0$ (i.e., only simple height constraints). We denote simple height constraint atoms simply as $h(i) = h(j)$ and $h(i) < h(j)$, by omitting the integer constant 0. The constraint structures He, he, H, h *are defined as $\langle C, \mathsf{PosVar}, \models, |.|, \mathsf{Width}\rangle$, by replacing C by C_{He}, C_{he}, C_H, C_h, respectively, and where $\mathsf{PosVar}, \models, |.|, \mathsf{Width}$ are defined as above.*

3.2 Decidability of simple height constraints

In this section we prove decidability of emptiness and finiteness of the language recognized by $\mathtt{TACBB}_\mathsf{h}$. A usual way to deal with automata with constraints is to transform them into a normalized form that is easier to deal with. We proceed by normalizing simple height constraints according to the following definition and lemma.

Definition 3.4. *A* normalized simple height constraint *is an expression c of the form $S_1 < S_2 < \cdots < S_n$, where $S_1, \ldots, S_n$ are non-empty, pairwise disjoint, finite sets of natural numbers. By $\mathsf{PosVar}(c)$ we denote $S_1 \uplus \ldots \uplus S_n$. Note that $\{S_1, \ldots, S_n\}$ is a partition of $\mathsf{PosVar}(c)$. An* interpretation *of c is a partial mapping $I : \mathbb{N} \to \mathcal{T}(\Sigma)$ such that $\mathsf{PosVar}(c) \subseteq \mathsf{Dom}(I)$. We say that I* satisfies *(*is a solution of*) c, denoted $I \models c$, if:*

- $\mathsf{height}(I(i_1)) = \mathsf{height}(I(i_2))$ *holds for each $i \in \{1, \ldots, n\}$, $i_1, i_2 \in S_i$,*
- $\mathsf{height}(I(i_1)) < \mathsf{height}(I(i_2))$ *holds for each $i \in \{1, \ldots, n-1\}$, $i_1 \in S_i$, $i_2 \in S_{i+1}$.*

The size *of c is $|c| = |\mathsf{PosVar}(c)|$. The* width *of c is $\mathsf{Width}(c) = 1$ (in simple height constraints the width plays no important role). The* structure of normalized simple height constraints *is the constraint structure* $\mathsf{nh} = \langle C_{nh}, \mathsf{PosVar}, \models, |.|, \mathsf{Width}\rangle$*, where* C_{nh} *is the set of all normalized simple height constraints, and* $\mathsf{PosVar}, \models, |.|, \mathsf{Width}$ *are defined as above.*

Lemma 3.5. *Let* $A = \langle Q, \Sigma, F, \Delta\rangle$ *be a* $\mathtt{TACBB}_{\mathsf{h}}$. *Then, a* $\mathtt{TACBB}^{\mathsf{F}}_{\mathsf{nh}}$ $A' = \langle Q, \Sigma, F, \Delta'\rangle$ *satisfying* $\mathcal{L}(A') = \mathcal{L}(A)$ *can be computed with time in* $\mathcal{O}(|A| \cdot 2^{\mathsf{maxar}^2})$*.*

Proof. We define $\Delta' := \{f(q_1, \dots, q_m) \xrightarrow{c'} q \mid c' \in C_{nh} \wedge \mathsf{PosVar}(c') = \{1, \dots, m\} \wedge \exists (f(q_1, \dots, q_m) \xrightarrow{c} q) \in \Delta : (c' \models c)\}$. Observe that a rule $(f(q_1, \dots, q_m) \xrightarrow{c} q) \in \Delta$ can be applied at a position in a term if and only if one of the rules of the form $(f(q_1, \dots, q_m) \xrightarrow{c'} q) \in \Delta'$ generated from it can be applied at such position. Thus, any run of A can be transformed into a run of A' reaching the same states at each position, and the reverse transformation is also possible. Hence, $\mathcal{L}(A') = \mathcal{L}(A)$.

To construct Δ' we need to decide $c' \models c$ for each normalized simple height constraint c' and simple height constraint c. This is easy, since it suffices to check if c evaluates to true after replacing each atom of the form $h(i) = h(j)$ by true if i, j are in the same set in c', and by false otherwise, and after replacing each atom of the form $h(i) < h(j)$ by true if i occurs in c' in a set previous to the one where j occurs, and by false otherwise. The number of different normalized simple height constraints c' to consider is bounded by 2^{maxar^2}. Thus, $|\Delta'| \leq |\Delta| \cdot 2^{\mathsf{maxar}^2}$, and the time complexity of the statement follows. ■

Example 3.6. *Consider the* $\mathtt{TACBB}_{\mathsf{h}}$ $A = \langle \{q, q_f\}, \Sigma, \{q_f\}, \Delta\rangle$*, where the signature* Σ *is* $\{a{:}0,\ g{:}1,\ f{:}2\}$*, and the set of rules* Δ *is:*

$$\begin{array}{ccc}
 & & f(q,q) \xrightarrow{\neg(h(1)=h(2))} q_f \\
a \rightarrow q & & f(q_f,q) \xrightarrow{\neg(h(1)=h(2))} q_f \\
g(q) \rightarrow q & & f(q,q_f) \xrightarrow{\neg(h(1)=h(2))} q_f \\
g(q_f) \rightarrow q_f & & f(q_f,q_f) \xrightarrow{\neg(h(1)=h(2))} q_f
\end{array}$$

Note that the rules guarantee that any accepted term has at least one occurrence of the alphabet symbol f*, and moreover, the constraints ensure that there are no siblings with identical height, that is, the language recognized by* A *is* $\mathcal{L}(A) = \{t \in \mathcal{T}(\Sigma) \mid (\exists p \in \mathsf{Pos}(t) : t(p) = f) \wedge (\forall p \in \mathsf{Pos}(t) : (t(p) = f \Rightarrow \mathsf{height}(t|_{p.1}) \neq \mathsf{height}(t|_{p.2})))\}$*.*

The construction in the proof of Lemma 3.5 allows to obtain a $\mathtt{TACBB}^{\mathsf{F}}_{\mathsf{nh}}$ A' *from* A *by just defining a new set of rules. In particular, for each of the rules of* A*, we have to consider any possible normalized constraint* c'*, and create a new rule with each* c' *that implies the original (non-normalized) constraint of the rule. For the rule* $a \rightarrow q$ *there is a single normalized constraint to consider: the empty one. For the rules* $g(q) \rightarrow q$ *and* $g(q_f) \rightarrow q_f$*, since the alphabet symbol has arity* 1*, there is also one single normalized constraint to consider,* $\{1\}$*, which clearly implies the empty constraint of the original rules. For the remaining rules, i.e., the ones with the non-normalized constraint* $c := \neg(h(1) = h(2))$*, since they have arity* 2 *we need to consider the normalized constraints* $c_1 := \{1, 2\}$*,* $c_2 := \{1\} < \{2\}$*, and* $c_3 := \{2\} < \{1\}$*. Note that* $c_1 \not\models c$*, whereas* $c_2 \models c$ *and* $c_3 \models c$*. Thus, each rule of* A *with the non-normalized* *constraint c produces two rules in A', one with the constraint c_2 and the other one with c_3. In summary, A' has the following set of rules:*

$$\begin{array}{ccc}
 & f(q,q) \xrightarrow{\{1\}<\{2\}} q_f & f(q,q) \xrightarrow{\{2\}<\{1\}} q_f \\
a \longrightarrow q & f(q_f,q) \xrightarrow{\{1\}<\{2\}} q_f & f(q_f,q) \xrightarrow{\{2\}<\{1\}} q_f \\
g(q) \xrightarrow{\{1\}} q & f(q,q_f) \xrightarrow{\{1\}<\{2\}} q_f & f(q,q_f) \xrightarrow{\{2\}<\{1\}} q_f \\
g(q_f) \xrightarrow{\{1\}} q_f & f(q_f,q_f) \xrightarrow{\{1\}<\{2\}} q_f & f(q_f,q_f) \xrightarrow{\{2\}<\{1\}} q_f
\end{array}$$

To decide emptiness of a $\mathtt{TACBB}^{\mathsf{F}}_{\mathsf{nh}}$ A, we iteratively consider terms of increasing height and, for each $h \geq 0$, we compute which states are reachable by runs of A on terms with height h. Clearly, such an approach is analogous to compute, for each $h \geq 0$, which rules of A can be applied at the root position of some term of height h. Consider that we have already performed such computations up to some h, and that now we tackle $h+1$. For a rule of the form $f(q_1,\ldots,q_m) \xrightarrow{c} q$ to be applicable at the root position of some term of height $h+1$, it suffices to guarantee that (i) there are runs on terms $t_1,\ldots,t_m$ of height at most h reaching the states $q_1,\ldots,q_m$, respectively, (ii) at least one of $t_1,\ldots,t_m$ has height h, and (iii) the interpretation $I = \{1 \mapsto t_1,\ldots,m \mapsto t_m\}$ is a solution of the normalized constraint c. All these conditions can be checked by considering the results obtained thus far for heights up to h.

In order to deduce a termination criterion for such process, we need to refine the approach. In particular, for each $h \geq 0$, we compute some extra information that allows to check the previous conditions (i) to (iii) for height $h+1$ by considering *only* the results obtained for height h, instead of all heights up to h. This extra information consists in memorizing whether a part of a constraint (a prefix, or infix) is satisfied by taking into account runs on terms with height bounded by h. The next definition formalizes the information computed for each $h \geq 0$.

Definition 3.7. *Let $A = \langle Q,\Sigma,F,\Delta\rangle$ be a $\mathtt{TACBB}^{\mathsf{F}}_{\mathsf{nh}}$. Let h be a natural number. Let q be a state in Q. We define $\mathsf{ExistTerm}_A(h,q)$ as true if there exists a term with height h in $\mathcal{L}(A,q)$, and as false otherwise. Let $N : \mathbb{N} \to Q$ be a partial mapping. We say that N and a partial mapping $I : \mathbb{N} \to \mathcal{T}(\Sigma)$ are* compatible *(with respect to A) if $\mathsf{Dom}(N) \supseteq \mathsf{Dom}(I)$ and $I(i) \in \mathcal{L}(A,N(i))$ for each $i \in \mathsf{Dom}(I)$. Let c be a normalized simple height constraint such that $\mathsf{Dom}(N) \supseteq \mathsf{PosVar}(c)$. We define $\mathsf{ExistSol}_A(h,c,N)$ as true if there exists $I : \mathsf{PosVar}(c) \to \mathcal{T}(\Sigma)$ such that I is compatible with N and a solution of c, and the highest term in $I(\mathsf{PosVar}(c))$ has height h, and as false otherwise. We define $\mathsf{AccExistSol}_A(h,c,N)$ as $\bigvee_{h'\leq h} \mathsf{ExistSol}_A(h',c,N)$. We omit the subindex of $\mathsf{ExistTerm}_A$, $\mathsf{ExistSol}_A$, $\mathsf{AccExistSol}_A$ when A is clear from the context.*

We define C_A as the set of the pairs $\langle c,N\rangle$ satisfying that c is a normalized simple height constraint, and there exists a transition rule $(f(q_1,\ldots,q_m) \xrightarrow{c'} q) \in \Delta$ such that c is a non-empty prefix of c' or a set occurring in c', and $N = \{1 \mapsto q_1,\ldots,m \mapsto q_m\}$. Note that $|C_A| \leq 2\cdot\mathsf{maxar}\cdot|\Delta|$. We define the configuration *of A for height h as the description of all values $\mathsf{ExistTerm}_A(h,q)$, $\mathsf{ExistSol}_A(h,c,N)$, $\mathsf{AccExistSol}_A(h,c,N)$ for each $\langle c,N\rangle \in C_A$ and $q \in Q$.*

Example 3.8. *Following Example 3.6, note that $C_{A'}$ is the set of the pairs $\langle\{1\},N\rangle$, $\langle\{2\},N'\rangle$, $\langle\{1\} < \{2\},N'\rangle$, and $\langle\{2\} < \{1\},N'\rangle$, where N and N' are any of the* *mappings* $\{1 \mapsto q,\ 2 \mapsto q\}$, $\{1 \mapsto q_f,\ 2 \mapsto q\}$, $\{1 \mapsto q,\ 2 \mapsto q_f\}$, *or* $\{1 \mapsto q_f,\ 2 \mapsto q_f\}$ *(i.e., the mappings corresponding to the rules with the alphabet symbol* f*), and* N *may additionally be any of the mappings* $\{1 \mapsto q\}$ *or* $\{1 \mapsto q_f\}$ *(i.e., the mappings corresponding to the rules with the alphabet symbol* g*).*

It is easy to see that $\mathsf{ExistTerm}(h, q)$ *is true for any height* $h \geq 0$*, since any term of the form* $g^n(a)$ *reaches* q*, with* $n \geq 0$*. On the other hand, note that the smallest terms with an occurrence of* f *and with siblings of different height are* $f(a, g(a))$ *and* $f(g(a), a)$*, with height* 2*. These are precisely the smallest terms reaching* q_f*, and thus,* $\mathsf{ExistTerm}(h, q_f)$ *is true only for* $h \geq 2$*.*

By the previous observations, it is clear that $\mathsf{ExistSol}(0, c, N)$ *and* $\mathsf{ExistSol}(1, c, N)$*, with* $\langle c, N\rangle \in C_{A'}$*, are necessarily false whenever* $q_f \in N(\mathsf{PosVar}(c))$*, since there is no term of height* 0 *or* 1 *reaching the state* q_f*. In the case of* $\mathsf{ExistSol}(0, c, N)$*, it is also false whenever* c *is of the form* $\{1\} < \{2\}$ *or* $\{2\} < \{1\}$*, since there is no interpretation where the highest term has height* 0 *and there is another smaller term. In the rest of cases,* $\mathsf{ExistSol}(0, c, N)$ *and* $\mathsf{ExistSol}(1, c, N)$ *are true. When considering greater heights, the constraints are easier to satisfy. In particular,* $\mathsf{ExistSol}(2, c, N)$ *is only false when* c *is of the form* $\{i\} < \{j\}$ *and* $N(i) = q_f$*, since the smallest term reaching* q_f *has height* 2*, and thus, it is not possible to find any solution for such* c *and* N*. Finally,* $\mathsf{ExistSol}(h, c, N)$ *with* $h \geq 3$ *is true for all* $\langle c, N\rangle \in C_{A'}$*.*

The following lemma gives an equivalent definition of ExistSol, AccExistSol, and ExistTerm using a recursive formalization. It can be proved by induction on h and $|c|$. This new definition shows that such values are computable, and moreover, it allows to argue that the configuration of A for height $h > 0$ depends only on the configuration for $h - 1$.

Lemma 3.9. *Let* $A = \langle Q, \Sigma, F, \Delta\rangle$ *be a* $\mathtt{TACBB}^{\mathsf{F}}_{\mathsf{nh}}$*.*

Then, ExistTerm*,* ExistSol*,* AccExistSol *can also be defined recursively as follows:*

- *Assume* $h = 0$*. Then,* $\mathsf{ExistTerm}(h, q)$ *is true if and only if there exists* $a \in \Sigma^{(0)}$ *satisfying* $a \in \mathcal{L}(A, q)$*. Moreover, if* c *is not just a set, then* $\mathsf{ExistSol}(h, c, N) = \mathsf{AccExistSol}(h, c, N) = \mathsf{false}$*, and otherwise if* c *is a set* S*, then* $\mathsf{ExistSol}(h, c, N) = \mathsf{AccExistSol}(h, c, N) = \bigwedge_{q \in N(S)} \mathsf{ExistTerm}(h, q)$*.*

  In the rest of cases assume $h > 0$*.*

- $\mathsf{AccExistSol}(h, c, N) = \mathsf{AccExistSol}(h - 1, c, N) \vee \mathsf{ExistSol}(h, c, N)$

- $\mathsf{ExistSol}(h, c' < S, N) = \mathsf{AccExistSol}(h - 1, c', N) \wedge \mathsf{ExistSol}(h, S, N)$

- $\mathsf{ExistSol}(h, S, N) = \bigwedge_{q \in N(S)} \mathsf{ExistTerm}(h, q)$

- $\mathsf{ExistTerm}(h, q) = \bigvee_{(f(q_1, \ldots, q_m) \xrightarrow{c} q) \in \Delta,\ m > 0} \mathsf{ExistSol}(h - 1, c, \{1 \mapsto q_1, \ldots, m \mapsto q_m\})$

Corollary 3.10. *Let* $A = \langle Q, \Sigma, F, \Delta\rangle$ *be a* $\mathtt{TACBB}^{\mathsf{F}}_{\mathsf{nh}}$*. Let* h *be a natural number.*

Then, the configuration of A *for height* $h+1$ *can be computed from the configuration of* A *for height* h *with time in* $\mathcal{O}(\mathsf{maxar}^2 \cdot |\Delta| + |Q|)$*.*

Proof. Straightforward from the definitions in Lemma 3.9. Observe that $\mathsf{ExistTerm}$ can be computed with time in $\mathcal{O}(|Q|+|\Delta|)$, afterwards $\mathsf{ExistSol}$ can be computed with time in $\mathcal{O}(|C_A|\cdot\mathsf{maxar})$, and finally $\mathsf{AccExistSol}$ with time in $\mathcal{O}(|C_A|)$. The statement follows since $|C_A| \leq 2\cdot\mathsf{maxar}\cdot|\Delta|$. ■

The previous result leads to the fact that, when we find the same configuration for heights $h_1 < h_2$, the sequence of configurations is periodic with period $h_2 - h_1$ starting from h_1. Thus, deciding emptiness corresponds to check whether there exists a final state q satisfying $\mathsf{ExistTerm}(h,q)$ for some $h < h_2$, and deciding finiteness corresponds to check whether there exists a final state q satisfying $\mathsf{ExistTerm}(h,q)$ for some $h_1 \leq h < h_2$. Hence, the time complexity of both decision algorithms depends on the cost from Corollary 3.10 to compute each configuration, and also the number of different possible configurations stated in the following lemma.

Lemma 3.11. *Let $A = \langle Q,\Sigma,F,\Delta\rangle$ be a $\mathtt{TACBB}^{\mathsf{F}}_{\mathsf{nh}}$.*
Then, the number of different configurations of A considering all possible heights is bounded by $2^{|Q|}\cdot(2\cdot\mathsf{maxar}\cdot|\Delta|+1)$.

Proof. Consider any two heights $0 < h_1 < h_2$ satisfying that $\mathsf{AccExistSol}$ for $h_1 - 1$ is equal to $\mathsf{AccExistSol}$ for $h_2 - 1$, and $\mathsf{ExistTerm}$ for h_1 is equal to $\mathsf{ExistTerm}$ for h_2. Since $\mathsf{ExistSol}$ for any height $h > 0$ depends only on $\mathsf{AccExistSol}$ for $h-1$ and $\mathsf{ExistTerm}$ for h, it follows that $\mathsf{ExistSol}$ for h_1 is equal to $\mathsf{ExistSol}$ for h_2. Moreover, it also follows that $\mathsf{AccExistSol}$ for h_1 is equal to $\mathsf{AccExistSol}$ for h_2. Thus, the configurations of A for h_1 and h_2 are equal, and hence, to prove the statement it suffices to bound the number of different combinations of $\mathsf{AccExistSol}$ and $\mathsf{ExistTerm}$. To this end, first note that there are at most $2^{|Q|}$ different definitions of $\mathsf{ExistTerm}$ in the configurations of A. Second, note that $\mathsf{AccExistSol}(h,c,N) \Rightarrow \mathsf{AccExistSol}(h+1,c,N)$ for any h and $\langle c,N\rangle \in C_A$, and thus, there are at most $|C_A|+1 \leq 2\cdot\mathsf{maxar}\cdot|\Delta|+1$ different definitions of $\mathsf{AccExistSol}$ in the configurations of A. The statement follows by combining both bounds. ■

Corollary 3.12. *Emptiness and finiteness of the language recognized by a $\mathtt{TACBB}^{\mathsf{F}}_{\mathsf{nh}}$ $A = \langle Q,\Sigma,F,\Delta\rangle$ are decidable with time in $2^{\mathcal{O}(|Q|+\log(\mathsf{maxar}\cdot|\Delta|))}$.*

As a consequence of Lemma 3.5 we also obtain decidability for $\mathtt{TACBB}_{\mathsf{h}}$.

Corollary 3.13. *Emptiness and finiteness of the language recognized by a $\mathtt{TACBB}_{\mathsf{h}}$ $A = \langle Q,\Sigma,F,\Delta\rangle$ are decidable with time in $2^{\mathcal{O}(|Q|+\log(|A|)+\mathsf{maxar}^2)}$.*

3.3 Decidability of the general case

The global approach to prove decidability of emptiness and finiteness for the general case of height and equality constraints is analogous to the case of simple height constraints. We start with a process of normalization of the automata, where the notion of normalized constraint is given in the following definition. Recall that, given a set of sets S, we use the notation $\bigcup S$ as shorthand for $\bigcup_{P\in S} P$, and that we write $e_1 \sim_S e_2$ to denote that the elements e_1 and e_2 belong to the same set in S.

Definition 3.14. *A* normalized height and equality constraint *is an expression c of the form $P_1 \otimes_1 P_2 \otimes_2 \cdots \otimes_{n-1} P_n$, where $P_1, \dots, P_n$ are partitions of non-empty finite sets of natural numbers, each natural number occurs in at most one P_i, and each $\otimes_i$ is an operator of the form either $=_h$ or $\leq_h$ for a natural number $h > 0$. By* $\mathsf{PosVar}(c)$ *we denote the set of natural numbers occurring in $P_1, \dots, P_n$. Note that $P_1 \uplus \dots \uplus P_n$ is a partition of* $\mathsf{PosVar}(c)$*. An* interpretation *of c is a partial mapping $I : \mathbb{N} \to \mathcal{T}(\Sigma)$ such that* $\mathsf{PosVar}(c) \subseteq \mathsf{Dom}(I)$*. We say that I* satisfies *(*is a solution of*) c, denoted $I \models c$, if:*

- $\left[(I(i_1) = I(i_2)) \Leftrightarrow (i_1 \sim_{P_i} i_2)\right]$ *holds for each $i \in \{1, \dots, n\}$, $i_1, i_2 \in \bigcup P_i$,*
- $\mathsf{height}(I(i_1)) = \mathsf{height}(I(i_2))$ *holds for each $i \in \{1, \dots, n\}$, $i_1, i_2 \in \bigcup P_i$,*
- $\mathsf{height}(I(i_1)) + h = \mathsf{height}(I(i_2))$ *holds for each $i \in \{1, \dots, n-1\}$, $i_1 \in \bigcup P_i$, $i_2 \in \bigcup P_{i+1}$ such that the operator $\otimes_i$ is of the form $=_h$,*
- $\mathsf{height}(I(i_1)) + h \leq \mathsf{height}(I(i_2))$ *holds for each $i \in \{1, \dots, n-1\}$, $i_1 \in \bigcup P_i$, $i_2 \in \bigcup P_{i+1}$ such that the operator $\otimes_i$ is of the form $\leq_h$.*

The size *of c is $|c| = |\mathsf{PosVar}(c)|$. The* width *of c, denoted* $\mathsf{Width}(c)$*, is the maximum between 1 and the natural numbers h occurring in the subscripts of the operators in c. The* structure of normalized height and equality constraints *is the constraint structure* $\mathsf{nHe} = \langle C_{nHe}, \mathsf{PosVar}, \models, |.|, \mathsf{Width}\rangle$*, where C_{nHe} is the set of all normalized height and equality constraints, and* $\mathsf{PosVar}, \models, |.|, \mathsf{Width}$ *are defined as above.*

Example 3.15. *Consider the language of AVL trees over $\Sigma = \{a{:}0,\ b{:}0,\ f{:}2\}$, i.e., the set of trees where the heights of the two direct children of any internal node differ by at most one. Moreover, assume that we impose that all the siblings must be different, i.e., that the two direct children of any internal node must be distinct trees. Such language can be recognized by the* $\mathtt{TACBB_{He}}$ *$A = \langle \{q\}, \Sigma, \{q\}, \Delta\rangle$, where the set of rules Δ contains $a \to q$, $b \to q$, and also:*

$$f(q,q) \xrightarrow{\begin{array}{l} 1 \not\approx 2 \wedge (h(1)=h(2) \vee \\ \qquad h(1)=h(2)+1 \vee \\ \qquad h(1)=h(2)-1) \end{array}} q$$

Note that the atom $1 \not\approx 2$ corresponds to the requirement that the siblings are distinct terms, and that the rest of atoms correspond to the notion of AVL trees. It is easy to see that such constraint cannot be directly expressed with a single normalized constraint, and that it requires to be decomposed into three distinct cases. First, the case where the siblings have identical height and are distinct terms can be expressed with the normalized constraint $\{\{1\},\{2\}\}$. Second, the case where the height of the second child is 1 plus the height of the first child can be expressed with $\{\{1\}\} =_1 \{\{2\}\}$. And third, the case where the height of the first child is 1 plus the height of the second child can be expressed with $\{\{2\}\} =_1 \{\{1\}\}$.

One of the essential differences between tackling $\mathtt{TACBB_{He}}$ and $\mathtt{TACBB_h}$ is that, for a $\mathtt{TACBB_{He}}$ A, it is not sufficient to iteratively consider increasing values for h, and compute which states are reachable by runs of A on terms with height h. This is because, in later iterations of the process, such information is not enough to easily deduce whether atoms of the form $i \approx j$ or $i \not\approx j$ can be satisfied. Instead, for each $h \geq 0$, we need to count (up to a certain bound) the number of distinct terms of height h that are recognized by each state of A. To ease the task of counting the terms, in the normalization process for $\mathtt{TACBB}_{\mathsf{He}}$ we also determinize the automaton, according to the following definition.

Definition 3.16. *We say that a* $\mathtt{TACBB}^{\mathsf{F}}_{\mathsf{nHe}}$ $A = \langle Q, \Sigma, F, \Delta \rangle$ *is* deterministic and complete *(or a* $\mathtt{dcTACBB}^{\mathsf{F}}_{\mathsf{nHe}}$*) if:*

- *for each* $f \in \Sigma^{(m)}$*, states* $q_1, \ldots, q_m \in Q$*, and normalized height and equality constraint* c*, there is at most one* $q \in Q$ *such that* $(f(q_1, \ldots, q_m) \xrightarrow{c} q) \in \Delta$*,*
- *for each* $f \in \Sigma^{(m)}$ *and states* $q_1, \ldots, q_m \in Q$*, the set of normalized height and equality constraints* $\{c \mid \exists q \in Q : (f(q_1, \ldots, q_m) \xrightarrow{c} q) \in \Delta\}$ *is non-empty, deterministic, and complete.*

Given a $\mathtt{dcTACBB}^{\mathsf{F}}_{\mathsf{nHe}}$ A *over* Σ *and a term* $t \in \mathcal{T}(\Sigma)$*, we denote by* $A(t)$ *the state reached by the unique run of* A *on* t *(such run exists thanks to the previous conditions).*

The construction of a $\mathtt{dcTACBB}^{\mathsf{F}}_{\mathsf{nHe}}$ from the given $\mathtt{TACBB}_{\mathsf{He}}$ is presented in several steps. First, we show how to obtain a deterministic and complete set of normalized constraints by constructing them from the following limited number of operators.

Definition 3.17. *Let* m *and* $w > 0$ *be natural numbers. A* normalized constraint with respect to m *and* w *is a normalized height and equality constraint* c *over the operators* $=_1$, $=_2$, $\ldots$, $=_{w-1}$ *and* $\leq_w$ *and satisfying* $\mathsf{PosVar}(c) = \{1, \ldots, m\}$.

We prove that the set C of the normalized constraints with respect to some m and $w > 0$ is deterministic and complete. For C to be deterministic, it must satisfy that its constraints are pairwise incompatible. For C to be complete, it must satisfy that any interpretation is a solution of at least one of the constraints of the set. We prove each property separately, and in Corollary 3.21 conclude that C is deterministic and complete.

Lemma 3.18. *Let* m *and* $w > 0$ *be natural numbers. Let* c *and* c' *be two different normalized constraints with respect to* m *and* w*.*

Then, c *and* c' *are incompatible.*

Proof. Since $\mathsf{PosVar}(c) = \mathsf{PosVar}(c') = \{1, \ldots, m\}$ and c and c' are different by assumption, it follows that necessarily $m > 1$. Consider the first difference of c and c' found by reading them from left to right, i.e., c and c' have prefixes of the form $P_1 \otimes_1 P_2 \otimes_2 \cdots \otimes_{n-1} P_n \otimes_n P$ and $P_1 \otimes_1 P_2 \otimes_2 \cdots \otimes_{n-1} P_n \otimes'_n P'$, respectively, where either $\otimes_n \neq \otimes'_n$ or $P \neq P'$. We consider the following cases:

- Consider the case where $n > 0$ and $\otimes_n \neq \otimes'_n$. In such case, the natural numbers h, h' occurring in $\otimes_n, \otimes'_n$, respectively, are different, and without loss of generality suppose $h < h'$. Consider an i occurring in P_n, and a j occurring in P. Any solution I of c satisfies $\mathsf{height}(I(i)) + h = \mathsf{height}(I(j))$, and any solution I of c' satisfies $\mathsf{height}(I(i)) + h < \mathsf{height}(I(j))$. Thus, c and c' are incompatible.

- Consider the case where either $n = 0$ or $\otimes_n = \otimes'_n$, and therefore, $P \neq P'$. We consider two subcases. First, assume that there is an i occurring in only one of P and P', say P. Let j be any natural number occurring in P'. Any solution I of c satisfies $\mathsf{height}(I(i)) \leq \mathsf{height}(I(j))$, and any solution I of c' satisfies $\mathsf{height}(I(j)) < \mathsf{height}(I(i))$. Thus, c and c' are incompatible.

  Second, assume that there are different i, j that occur in the same part in either P or P', say P, and i, j do not occur in the same part in P'. Any solution I of c satisfies $I(i) = I(j)$, and any solution I of c' satisfies $I(i) \neq I(j)$. Thus, c and c' are incompatible. ■

Definition 3.19. *Let m and $w > 0$ be natural numbers. Let $I : \{1, \ldots, m\} \to \mathcal{T}(\Sigma)$ be a mapping. Let $h_1 < h_2 < \cdots < h_n$ be the elements of* $\mathsf{height}(I(\{1, \ldots, m\}))$. *The constraint* induced *from I and w is the normalized constraint $P_1 \otimes_1 P_2 \otimes_2 \cdots \otimes_{n-1} P_n$ with respect to m and w where:*

- *each P_i is a partition of $\{j \in \{1, \ldots, m\} \mid \mathsf{height}(I(j)) = h_i\}$ satisfying that $(I(j_1) = I(j_2)) \Leftrightarrow (j_1 \sim_{P_i} j_2)$ for each $j_1, j_2 \in \bigcup P_i$,*

- *each $\otimes_i$ is the operator $=_{h_{i+1}-h_i}$ if $h_{i+1} - h_i < w$, and $\leq_w$ otherwise.*

Lemma 3.20. *Let m and $w > 0$ be natural numbers. Let $I : \{1, \ldots, m\} \to \mathcal{T}(\Sigma)$ be a mapping. Let c be the constraint induced from I and w.*
Then, $I \models c$.

Proof. Straightforward from the definition of induced constraint. ■

Corollary 3.21. *Let m and $w > 0$ be natural numbers. Let C be the set of normalized constraints with respect to m and w.*
Then, C is deterministic and complete.

Proof. By Lemma 3.18, any two different constraints in C are incompatible, and thus C is deterministic. By Lemma 3.20, any interpretation $I : \{1, \ldots, m\} \to \mathcal{T}(\Sigma)$ satisfies the constraint c induced from I and w. Since such c is in C by definition, it follows that C is complete. ■

We are now ready to introduce the normalization process for $\mathtt{TACBB}_{\mathsf{He}}$. The idea of the construction is that, for any term t, the normalized automaton simulates all possible runs of the original automaton on t, and accepts those terms where the original automaton could reach some final state.

Definition 3.22. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{TACBB}_{\mathsf{He}}$. For each natural number m, let C_m be the set of normalized constraints with respect to m and the width of A. The* normalization *of A is the $\mathtt{TACBB}^{\mathsf{F}}_{\mathsf{nHe}}$ $\langle 2^Q, \Sigma, \{F' \in 2^Q \mid F' \cap F \neq \emptyset\}, \Delta'\rangle$, where Δ' is:*

$$\begin{aligned}\{f(S_1, \ldots, S_m) \xrightarrow{c} S \mid f \in \Sigma^{(m)} \wedge c \in C_m \wedge S_1, \ldots, S_m \in 2^Q \wedge \\ S = \{q \in Q \mid \exists (f(q_1, \ldots, q_m) \xrightarrow{c'} q) \in \Delta : \\ q_1 \in S_1 \wedge \ldots \wedge q_m \in S_m \wedge c \models c'\}\}\end{aligned}$$

Example 3.23. *Following Example 3.15, first note that the width of A is 2. The normalization of A has the state set $\{\emptyset, \{q\}\}$, the final state set $\{\{q\}\}$, and its rule set has $a \to \{q\}$, $b \to \{q\}$, and also:*

$$\begin{aligned} f(\emptyset, \emptyset) &\xrightarrow{c_1} \emptyset \\ f(\emptyset, \{q\}) &\xrightarrow{c_1} \emptyset \\ f(\{q\}, \emptyset) &\xrightarrow{c_1} \emptyset \\ f(\{q\}, \{q\}) &\xrightarrow{c_2} \emptyset \\ f(\{q\}, \{q\}) &\xrightarrow{c_3} \{q\} \end{aligned}$$

where c_1 has to be replaced by each normalized constraint (with respect to $m = 2$ and $w = 2$), c_2 by each normalized constraint (with respect to $m = 2$ and $w = 2$) that does not imply the original non-normalized constraint of the rule for f, and c_3 by each normalized constraint (with respect to $m = 2$ and $w = 2$) that implies such original non-normalized constraint, i.e., by $\{\{1\}, \{2\}\}$, $\{\{1\}\} =_1 \{\{2\}\}$, or $\{\{2\}\} =_1 \{\{1\}\}$ as explained in Example 3.15.

Lemma 3.24. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\mathtt{TACBB}_{\mathsf{He}}$ with width w.*

Then, the normalization A' of A is deterministic and complete, i.e., a $\mathtt{dcTACBB}^{\mathsf{F}}_{\mathsf{nHe}}$, and can be computed with time in $2^{\mathcal{O}(\log(|\Sigma|) + \mathsf{maxar} \cdot (|Q| + \mathsf{maxar} + \log(w)) + \log(|A|))}$.

Proof. The fact that A' is a $\mathtt{dcTACBB}^{\mathsf{F}}_{\mathsf{nHe}}$ follows from Definition 3.16, Corollary 3.21 and Definition 3.22. The time complexity follows from these observations: each rule of A' can be computed with time in $\mathcal{O}(|A|)$, the number of rules of A' is bounded by $|\Sigma| \cdot 2^{|Q| \cdot \mathsf{maxar}} \cdot |C_{\mathsf{maxar}}|$, where C_{maxar} is the set of normalized constraints with respect to maxar and w, and $|C_{\mathsf{maxar}}|$ is in $2^{\mathcal{O}(\mathsf{maxar}^2 + \mathsf{maxar} \cdot \log(w))}$. ■

It remains to see that the normalized automaton A' preserves the language recognized by the original automaton A. To this end, we first observe that the normalized constraints are, in some sense, more precise than the original ones: given a normalized constraint c_1 and an original constraint c_2, either c_1 and c_2 are incompatible, or $c_1 \models c_2$ holds. Then, we show that, for any term t, the unique state $A'(t)$ is precisely the set of states that are reachable by runs of A on t.

Lemma 3.25. *Let m and $w > 0$ be natural numbers. Let c_1 be a normalized constraint with respect to m and w. Let c_2 be a height and equality constraint whose width is smaller than or equal to w and satisfying $\mathsf{PosVar}(c_2) \subseteq \{1, \ldots, m\}$.*

Then, either c_1 and c_2 are incompatible, or $c_1 \models c_2$ holds.

Proof (Sketch). It suffices to note that for any two solutions I_1, I_2 of c_1 and any atom c_2' occurring in c_2, either $I_1 \models c_2'$ and $I_2 \models c_2'$, or $I_1 \not\models c_2'$ and $I_2 \not\models c_2'$. ■

Lemma 3.26. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\mathtt{TACBB}_{\mathsf{He}}$. Let A' be the normalization of A. Let $t \in \mathcal{T}(\Sigma)$ be a term.*

Then, the state reached by the unique run of A' on t is $S = \{q \in Q \mid t \in \mathcal{L}(A, q)\}$.

Proof. We prove it by induction on $\mathsf{height}(t)$. Let t be more explicitly written of the form $f(t_1, \ldots, t_m)$. By induction hypothesis, the states reached by the unique runs of A' on $t_1, \ldots, t_m$ are $S_1, \ldots, S_m$, respectively, where $S_i = \{q \in Q \mid t_i \in \mathcal{L}(A, q)\}$. Let c be the constraint induced from $I = \{1 \mapsto t_1, \ldots, m \mapsto t_m\}$ and the width of A. By Definition 3.22, A' has the rule $f(S_1, \ldots, S_m) \xrightarrow{c} S'$, where $S' = \{q \in Q \mid \exists (f(q_1, \ldots, q_m) \xrightarrow{c'} q) \in \Delta : q_1 \in S_1 \wedge \ldots \wedge q_m \in S_m \wedge c \models c'\}$. Such rule can be applied at the root position of t since $I \models c$ holds by Lemma 3.20, and moreover, by Corollary 3.21 no other rule can. It remains to prove $S' = S$:

$\subseteq$) Let $q \in S'$. By definition there is a rule $(f(q_1, \ldots, q_m) \xrightarrow{c'} q) \in \Delta$ such that $q_1 \in S_1, \ldots, q_m \in S_m$ and $c \models c'$. Thus, $I \models c'$, and since we had that $t_1 \in \mathcal{L}(A, q_1), \ldots, t_m \in \mathcal{L}(A, q_m)$, it follows $t \in \mathcal{L}(A, q)$, and hence, $q \in S$.

$\supseteq$) Let $q \in S$. By definition there is a run of A on t with a rule of the form $f(q_1, \ldots, q_m) \xrightarrow{c'} q$ applied at the root position. Note that $I \models c'$, and since $I \models c$, by Lemma 3.25, $c \models c'$. Since we had that $q_1 \in S_1, \ldots, q_m \in S_m$, it follows $q \in S'$. ■

Corollary 3.27. *Let A be a $\texttt{TACBB}_{\mathsf{He}}$. Let A' be the normalization of A. Then, $\mathcal{L}(A') = \mathcal{L}(A)$.*

To decide emptiness of a $\texttt{dcTACBB}^{\mathsf{F}}_{\mathsf{nHe}}$ A, we iteratively consider increasing values for h, and compute how many runs (up to a certain bound) of A on terms with height h reach each state. We start by a previous definition describing which terms (and not only how many of them) are reached, in order to ease later arguments.

Definition 3.28. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\texttt{dcTACBB}^{\mathsf{F}}_{\mathsf{nHe}}$. Let h be a natural number. Let q be a state in Q. We define $\mathsf{Terms}_A(h, q)$ as $\{t \in \mathcal{T}(\Sigma) \mid A(t) = q \wedge \mathsf{height}(t) = h\}$. Let $N : \mathbb{N} \to Q$ be a partial mapping. We say that N and a partial mapping $I : \mathbb{N} \to \mathcal{T}(\Sigma)$ are* compatible *(with respect to A) if $\mathsf{Dom}(N) \supseteq \mathsf{Dom}(I)$ and $A(I(i)) = N(i)$ for each $i \in \mathsf{Dom}(I)$. Let c be a normalized height and equality constraint such that $\mathsf{Dom}(N) \supseteq \mathsf{PosVar}(c)$. We define $\mathsf{Sols}_A(h, c, N)$ as the set of interpretations $I : \mathsf{PosVar}(c) \to \mathcal{T}(\Sigma)$ such that I is compatible with N and a solution of c, and the highest term occurring in $I(\mathsf{PosVar}(c))$ has height h. We define $\mathsf{AccSols}_A(h, c, N)$ as $\bigcup_{h' \leq h} \mathsf{Sols}_A(h', c, N)$. We omit the subindex of Terms_A, Sols_A, $\mathsf{AccSols}_A$ when A is clear from the context.*

We define C_A as the set of the pairs $\langle c, N\rangle$ satisfying that c is a normalized height and equality constraint, and there exists a transition rule $(f(q_1, \ldots, q_m) \xrightarrow{c'} q) \in \Delta$ such that c is a non-empty prefix of c' or a partition occurring in c', and $N = \{1 \mapsto q_1, \ldots, m \mapsto q_m\}$. Note that $|C_A| \leq 2 \cdot \mathsf{maxar} \cdot |\Delta|$.

The following lemma gives an equivalent definition of Sols, $\mathsf{AccSols}$, and Terms using a recursive formalization. It can be proved by induction on h and $|c|$. This new definition shows that such values are computable, and moreover, it allows to argue that the values for a height $h > 0$ depend only on the values for heights $h - i$, with i bounded by the width of A. In the statement, we use a special operator $\boxtimes$ between sets of sets that is defined as follows. Given two sets of sets S_1 and S_2, by $S_1 \boxtimes S_2$ we denote the set of the sets that are obtained by unioning any set $\hat{S}_1$ from S_1 with any set $\hat{S}_2$ from S_2, i.e., $S_1 \boxtimes S_2 = \{\hat{S}_1 \cup \hat{S}_2 \mid \hat{S}_1 \in S_1 \ \wedge \ \hat{S}_2 \in S_2\}$. For example, $\{\{a, b\}, \{c\}\} \boxtimes \{\{d, e\}, \{f\}\} = \{\{a, b, d, e\}, \{a, b, f\}, \{c, d, e\}, \{c, f\}\}$.

Lemma 3.29. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a* $\texttt{dcTACBB}^{\mathsf{F}}_{\mathsf{nHe}}$.
Then, Terms, Sols, AccSols *can also be defined recursively as follows:*

- *If $h < 0$, then* $\mathsf{Terms}(h, q) = \mathsf{Sols}(h, c, N) = \mathsf{AccSols}(h, c, N) = \emptyset$ *for any q, c, N.*
- *Assume $h = 0$. Then,* $\mathsf{Terms}(h, q) = \{a \in \Sigma^{(0)} \mid A(a) = q\}$. *Moreover, if c is not just a partition, then* $\mathsf{Sols}(h, c, N) = \mathsf{AccSols}(h, c, N) = \emptyset$*, and otherwise if c is a partition P, then* $\mathsf{Sols}(h, c, N) = \mathsf{AccSols}(h, c, N) = \{I : \mathsf{PosVar}(P) \to \Sigma^{(0)} \mid (\forall i \in \mathsf{Dom}(I) : I(i) \in \mathsf{Terms}(h, N(i))) \wedge (\forall i, j \in \mathsf{Dom}(I) : ((I(i) = I(j)) \Leftrightarrow (i \sim_P j)))\}$.

  In the rest of cases assume $h > 0$.
- $\mathsf{AccSols}(h, c, N) = \mathsf{AccSols}(h - 1, c, N) \uplus \mathsf{Sols}(h, c, N)$
- $\mathsf{Sols}(h, c' =_{h'} P, N) = \mathsf{Sols}(h - h', c', N) \boxtimes \mathsf{Sols}(h, P, N)$
- $\mathsf{Sols}(h, c' \leq_{h'} P, N) = \mathsf{AccSols}(h - h', c', N) \boxtimes \mathsf{Sols}(h, P, N)$
- $\mathsf{Sols}(h, P, N) = \{I : \mathsf{PosVar}(P) \to \mathcal{T}(\Sigma) \mid (\forall i \in \mathsf{Dom}(I) : I(i) \in \mathsf{Terms}(h, N(i)))$ $\wedge(\forall i, j \in \mathsf{Dom}(I) : ((I(i) = I(j)) \Leftrightarrow (i \sim_P j)))\}$
- $\mathsf{Terms}(h, q) = \{f(I(1), \ldots, I(m)) \mid (f(q_1, \ldots, q_m) \xrightarrow{c} q) \in \Delta \ \wedge\ m > 0 \ \wedge\ I \in \mathsf{Sols}(h - 1, c, \{1 \mapsto q_1, \ldots, m \mapsto q_m\})\}$

As we have mentioned above, we are not interested in computing all the terms of height h that can be recognized by each of the states, but only how many of them there are, up to a certain bound. This bound is actually maxar: this is enough since disequalities can only be tested between sibling positions, and hence, there are at most maxar siblings that can be forced to be different in order to satisfy a constraint.

Definition 3.30. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a* $\texttt{dcTACBB}^{\mathsf{F}}_{\mathsf{nHe}}$. *Let h be a natural number. Let q be a state in Q. We define* $\#\mathsf{Terms}_A(h, q)$ *as* $\min\{\mathsf{maxar}, |\mathsf{Terms}_A(h, q)|\}$. *Let c be a normalized height and equality constraint. Let $N : \mathbb{N} \to Q$ be a partial mapping such that* $\mathsf{Dom}(N) \supseteq \mathsf{PosVar}(c)$. *We define* $\#\mathsf{Sols}_A(h, c, N)$ *and* $\#\mathsf{AccSols}_A(h, c, N)$ *as* $\min\{\mathsf{maxar}, |\mathsf{Sols}_A(h, c, N)|\}$ *and* $\min\{\mathsf{maxar}, |\mathsf{AccSols}_A(h, c, N)|\}$*, respectively. We omit the subindex of* $\#\mathsf{Terms}_A$, $\#\mathsf{Sols}_A$, $\#\mathsf{AccSols}_A$ *when A is clear from the context.*

We define the configuration *of A for height h as the description of all values* $\#\mathsf{Terms}_A(h, q)$, $\#\mathsf{Sols}_A(h, c, N)$, $\#\mathsf{AccSols}_A(h, c, N)$ *for each $\langle c, N\rangle \in C_A$ and $q \in Q$.*

The following recursive definition follows from Lemma 3.29 and Definition 3.30. To simplify the presentation, we introduce a special function for counting the number of distinct ways to satisfy a constraint composed of a single partition. More precisely, for any height h and pair $\langle c, N\rangle \in C_A$ such that c is a partition P, we define $\mathsf{permutations}(h, P, N)$ as 0 when there is a part $P' \in P$ such that $|N(P')| > 1$ (note that such partition P is necessarily unsatisfiable for the given N since two distinct states of a deterministic automaton recognize disjoint languages), and otherwise when there is no such part P', as $\prod_{q \in N(\mathsf{PosVar}(P))} \prod_{0 \leq i < |\{P' \in P : N(P') = \{q\}\}|} (\#\mathsf{Terms}(h, q) - i)$.

Lemma 3.31. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a* $\texttt{dcTACBB}^{\mathsf{F}}_{\mathsf{nHe}}$.
Then, #Terms, #Sols, #AccSols *can also be defined recursively as follows:*

- *If $h < 0$, then $\#\mathsf{Terms}(h, q) = \#\mathsf{Sols}(h, c, N) = \#\mathsf{AccSols}(h, c, N) = 0$ for any q, c, N.*
- *Assume $h = 0$. Then, $\#\mathsf{Terms}(h, q) = \min\{\mathsf{maxar}, |\{a \in \Sigma^{(0)} \mid A(a) = q\}|\}$. Moreover, if c is not just a partition, then $\#\mathsf{Sols}(h, c, N) = \#\mathsf{AccSols}(h, c, N) = 0$, and otherwise if c is a partition P, then $\#\mathsf{Sols}(h, c, N) = \#\mathsf{AccSols}(h, c, N) = \min\{\mathsf{maxar}, \mathsf{permutations}(h, P, N)\}$.*

  In the rest of cases assume $h > 0$.
- $\#\mathsf{AccSols}(h, c, N) = \min\{\mathsf{maxar}, \#\mathsf{AccSols}(h-1, c, N) + \#\mathsf{Sols}(h, c, N)\}$
- $\#\mathsf{Sols}(h, c' =_{h'} P, N) = \min\{\mathsf{maxar}, \#\mathsf{Sols}(h-h', c', N) \cdot \#\mathsf{Sols}(h, P, N)\}$
- $\#\mathsf{Sols}(h, c' \leq_{h'} P, N) = \min\{\mathsf{maxar}, \#\mathsf{AccSols}(h-h', c', N) \cdot \#\mathsf{Sols}(h, P, N)\}$
- $\#\mathsf{Sols}(h, P, N) = \min\{\mathsf{maxar}, \mathsf{permutations}(h, P, N)\}$
- $\#\mathsf{Terms}(h, q) = \min\{\mathsf{maxar}, \sum_{(f(q_1,\ldots,q_m) \xrightarrow{c} q) \in \Delta,\ m>0} \#\mathsf{Sols}(h-1, c, \{1 \mapsto q_1, \ldots, m \mapsto q_m\})\}$

The cost of computing the configuration for a given height, provided that the configurations for the w previous heights have already been computed, can easily be deduced from the previous lemma. To simplify the arguments, we assume that each of the arithmetic operations involved in the computation of #Terms, #Sols, #AccSols can be done in constant time, and thus, to justify the cost of computing the current configuration it suffices to consider the total amount of such operations. This simplification will not affect the final complexity of the decision algorithm, since the cost of the arithmetic operations will be subsumed by other factors.

Corollary 3.32. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{dcTACBB}^{\mathsf{F}}_{\mathsf{nHe}}$ with width w. Let h be a natural number.*

Then, the configuration of A for height $h + w$ can be computed from the configurations of A for heights $h, h+1, \ldots, h+w-1$ with time in $\mathcal{O}(\mathsf{maxar}^2 \cdot |\Delta| + |Q|)$.

Proof. Straightforward from the definitions in Lemma 3.31. Observe that #Terms can be computed with time in $\mathcal{O}(|Q| + |\Delta|)$. Afterwards, #Sols for the case where the considered constraint is a single partition can be computed with time in $\mathcal{O}(|C_A| \cdot \mathsf{maxar})$, and for the remaining cases in $\mathcal{O}(|C_A|)$. Overall, #Sols can be computed with time in $\mathcal{O}(|C_A| \cdot \mathsf{maxar})$. Finally, #AccSols can be computed with time in $\mathcal{O}(|C_A|)$. The statement follows since $|C_A| \leq 2 \cdot \mathsf{maxar} \cdot |\Delta|$. ■

The previous result leads to the fact that, when we find the same w consecutive configurations starting at two different heights $h_1 < h_2$, the sequence of configurations is periodic with period $h_2 - h_1$ starting from h_1. Thus, deciding emptiness corresponds to check whether there exists a final state q satisfying $\#\mathsf{Terms}(h, q) \geq 1$ for some $h < h_2$, and deciding finiteness corresponds to check whether there exists a final state q satisfying $\#\mathsf{Terms}(h, q) \geq 1$ for some $h_1 \leq h < h_2$. Hence, the time complexity of both decision algorithms depends on the cost from Corollary 3.32 to compute each configuration, and also the number of different possible groups of w consecutive configurations.

Lemma 3.33. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\mathtt{dcTACBB}^{\mathsf{F}}_{\mathsf{nHe}}$.*

Then, the number of different configurations of A considering all possible heights is bounded by $(\mathsf{maxar}+1)^{|Q|+2\cdot\mathsf{maxar}\cdot|\Delta|} \cdot (2\cdot\mathsf{maxar}^2\cdot|\Delta|+1)$.

Proof. Similar to Lemma 3.11, except that instead of dealing with truth values we have numbers in $\{0,\ldots,\mathsf{maxar}\}$, and that we need to take into account the number of different $\#\mathsf{Sols}$. More precisely, there are at most $(\mathsf{maxar}+1)^{|Q|}$ different definitions of $\#\mathsf{Terms}$ in the configurations of A, there are at most $(\mathsf{maxar}+1)^{|C_A|}$ different definitions of $\#\mathsf{Sols}$ in the configurations of A, and finally, since $\#\mathsf{AccSols}(h,c,N) \leq \#\mathsf{AccSols}(h+1,c,N)$ for any h and $\langle c, N\rangle \in C_A$, there are at most $|C_A|\cdot\mathsf{maxar}+1$ different definitions of $\#\mathsf{AccSols}$ in the configurations of A. The statement follows by combining these bounds and noting that $|C_A| \leq 2\cdot\mathsf{maxar}\cdot|\Delta|$. ■

Corollary 3.34. *Emptiness and finiteness of the language recognized by a $\mathtt{dcTACBB}^{\mathsf{F}}_{\mathsf{nHe}}$ $A = \langle Q, \Sigma, F, \Delta \rangle$ with width w are decidable with time in $2^{\mathcal{O}(w\cdot\log(\mathsf{maxar})\cdot(|Q|+\mathsf{maxar}\cdot|\Delta|))}$.*

As a consequence of Lemma 3.24 we also obtain decidability for $\mathtt{TACBB}_{\mathsf{He}}$.

Corollary 3.35. *Emptiness and finiteness of the language recognized by a $\mathtt{TACBB}_{\mathsf{He}}$ $A = \langle Q, \Sigma, F, \Delta \rangle$ are decidable with time in $2^{2^{\mathcal{O}(\log(|\Sigma|)+\mathsf{maxar}\cdot(|Q|+\mathsf{maxar}+\log(w))+\log(|A|))}}$, where w is the width of A.*

As a final remark, we consider the simpler cases of $\mathtt{TACBB}_{\mathsf{he}}$ and $\mathtt{TACBB}_{\mathsf{H}}$. For the normalization A' of a $\mathtt{TACBB}_{\mathsf{he}}$ A, since the normalized constraints would only have operators of the form $\leq_1$, we could refine Lemma 3.33 as we did in Lemma 3.11 and ignore the number of different $\#\mathsf{Sols}$, thus obtaining that the number of possible configurations of A' is bounded by $(\mathsf{maxar}+1)^{|Q'|}\cdot(2\cdot\mathsf{maxar}^2\cdot|\Delta'|+1)$. Moreover, for the sequence of configurations to become periodic it would suffice that two configurations coincided (since we look for identical groups of $w = 1$ consecutive configurations), and hence we would get the following result.

Corollary 3.36. *Emptiness and finiteness of the language recognized by a $\mathtt{TACBB}_{\mathsf{he}}$ A are decidable with time in $2^{\mathcal{O}(\log(\mathsf{maxar})\cdot 2^{|Q|}+\log(|\Sigma|)+\mathsf{maxar}\cdot(|Q|+\mathsf{maxar})+\log(|A|))}$, where Q is the state set of A and Σ its signature.*

For the normalization A' of a $\mathtt{TACBB}_{\mathsf{H}}$ A, it is not required to obtain a deterministic and complete automaton, and instead of using partitions in the normalized constraints we can simply use sets, as we did for $\mathtt{TACBB}_{\mathsf{h}}$. Thus, a refined normalization process can preserve the same state set Q and generate the rule set Δ' combining ideas from Lemmas 3.5 and 3.24 with time in $2^{\mathcal{O}(\mathsf{maxar}^2+\mathsf{maxar}\cdot\log(w)+\log(|A|))}$. Also, we do not need to count the number of terms for each state, just a truth value. Hence, Lemma 3.33 could be refined to state that the number of configurations of A' is bounded by $2^{|Q|+2\cdot\mathsf{maxar}\cdot|\Delta'|}\cdot(2\cdot\mathsf{maxar}\cdot|\Delta'|+1)$.

Corollary 3.37. *Emptiness and finiteness of the language recognized by a $\mathtt{TACBB}_{\mathsf{H}}$ A are decidable with time in $2^{2^{\mathcal{O}(\mathsf{maxar}^2+\mathsf{maxar}\cdot\log(w)+\log(|A|))}}$, where w is the width of A.*

Chapter 4

EXPTIME-completeness of the HOM problem

Here we focus on automata with arbitrary local disequality constraints, and with implicit HOM equality constraints ($\mathtt{TA}_{\mathsf{ihom},\not\approx}$). The equalities are called implicit since they are not explicitly written in the transitions rules, and instead, are encoded in the left-hand sides as follows: two positions of the left-hand side with the same state implicitly define a local equality constraint between such positions. Moreover, the left-hand sides are allowed to be arbitrary terms containing states at some leaf positions, and thus, it is possible to define equality constraints between non-brother positions. This definition of equality constraints is specially suited to recognize sets defined as images of regular tree languages under tree homomorphisms. For this reason, several decision problems on such kind of sets can be reduced to the emptiness problem for $\mathtt{TA}_{\mathsf{ihom},\not\approx}$, like set inclusion, finiteness of set difference, or regularity [GG13]. The latter problem, also called the HOM problem, can be formally defined as follows:

Input: a $\mathtt{TA}$ A and a tree homomorphism H.
Question: is $H(\mathcal{L}(A))$ regular?

We tackle the emptiness problem for $\mathtt{TA}_{\mathsf{ihom},\not\approx}$, and in this way, conclude EXPTIME-completeness of all those problems.

The remaining of this chapter is organized as follows. In Section 4.1 we summarize the approach to prove decidability of emptiness of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$, providing intuition on the crucial insights. In Section 4.2 we recall some known hardness results, the definition of $\mathtt{TA}_{\mathsf{hom},\not\approx}$ from [GG13], and some results of that paper. In Section 4.3 we formally define $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ and show their equivalence with $\mathtt{TA}_{\mathsf{hom},\not\approx}$. In Sections 4.4 and 4.5 we introduce some technical results on a combinatorial property, define a special kind of positions and replacements, and obtain a necessary condition for minimal accepting runs of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. In Section 4.6 we give the algorithm deciding emptiness of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ in exponential time. In Section 4.7 we show the consequences of this result by reducing other problems to it. In particular, we prove that HOM is EXPTIME-complete.

4.1 Intuition on the approach

Tree automata with disequality constraints ($\mathtt{TA}_{\not\approx}$) are a particular case of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ where only disequality constraints are allowed. Emptiness of $\mathtt{TA}_{\not\approx}$ was proved decidable in exponential time in [CJ03], and we generalize the techniques of that paper to $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Nevertheless, the proofs in [CJ03] are too complex to allow direct generalizations, and need to be refined. In this section we give an overview of our approach, its common points with [CJ03], and its differences. The explanation uses an example of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ that also allows to justify the advantages of this formalism with respect to the notion of $\mathtt{TA}_{\mathsf{hom},\not\approx}$ introduced in [GG13].

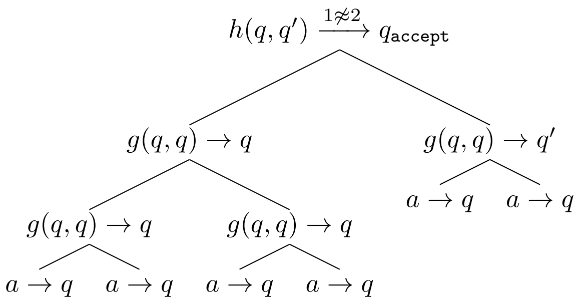
The algorithm deciding emptiness of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ looks for an accepting run by iteratively generating all possible runs in increasing order of size: new runs are constructed using the previous runs as direct subruns. In order to guarantee termination, some runs are discarded when the algorithm realizes that they cannot be subruns of the minimum accepting run. Below we briefly describe the involved discarding criterion.

In order to determine that a run r' cannot be subrun of the minimum accepting run, we prove that any run r having r' as subrun can be transformed into a smaller run reaching the same state. The transformation proceeds by replacing subruns of r by other smaller subruns. Note that the result of the replacement must be a new correct run, and in particular, it must still satisfy the equality and disequality constraints. Recall that an equality constraint demands identity of the respective pending subterms. To this end, it is convenient to replace all subruns that the constraints force to be identical by the same smaller subrun.

Arguments based on multiple replacements have already appeared in the literature, and, in particular, in [GG13] they are used to deal with images of regular tree languages under tree homomorphisms. In fact, $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ are expressively equivalent to the $\mathtt{TA}_{\mathsf{hom},\not\approx}$ defined in [GG13]. Nevertheless, the fact that equalities of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ are implicitly forced by identity of states provides a great advantage to reason about replacements. To support this statement, consider the language of terms of the form $h(t_1, t_2)$, where t_1, t_2 are different complete trees over nullary a and binary g. Such language can be recognized by a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ with the following transition rules:

$$\begin{array}{lll} a \to q & g(q,q) \to q & h(q,q') \xrightarrow{1\not\approx 2} q_{\mathtt{accept}} \\ a \to q' & g(q,q) \to q' & \end{array}$$

where $q_{\mathtt{accept}}$ is the only final state. Now, consider the following run r of such automaton on the term $h(g(g(a,a),g(a,a)),g(a,a))$:



Note that the terms pending at the positions 1.1.1, 1.1.2, 1.2.1, 1.2.2 are forced to be identical by the implicit equality constraints: the same sequence of states $q_{\texttt{accept}}.q.q.q$ is found while traversing the run from the root position to any of them. Moreover, note that the subruns pending at the positions 1.1.1, 1.1.2, 1.2.1, 1.2.2 are the same run $a \to q$. Without loss of generality, we can assume that runs are "uniform" in the sense that the implicit equality constraints force not only identity of the pending subterms but also of the pending subruns.

For a given arbitrary run, a sequence of states $q_1.q_2 \dots q_n$ describes a set of positions whose pending subterms (and subruns if the given run is uniform) are forced to be identical. We call such a sequence of states an abstract position, and adapt typical notations of subterm location and replacement to abstract positions. For example, $r|_{q_1.q_2\dots q_n}$ denotes the subrun pending at any of the positions referred by $q_1.q_2 \dots q_n$, and $r[r']_{q_1.q_2\dots q_n}$ denotes the simultaneous replacement of all such subruns by a new subrun r'. For instance, in the previous example run, $r[r|_{q_{\texttt{accept}}.q.q.q}]_{q_{\texttt{accept}}.q.q}$ is:

$$\begin{tikzcd}
 & & & h(q,q') \xrightarrow{1\not\approx 2} q_{\texttt{accept}} \arrow[dll, no head] \arrow[drr, no head] & & & \\
 & g(q,q) \to q \arrow[dl, no head] \arrow[dr, no head] & & & & g(q,q) \to q' \arrow[dl, no head] \arrow[dr, no head] & \\
a \to q & & a \to q & & a \to q & & a \to q
\end{tikzcd}$$

Note that the implicit equality constraints are still satisfied after the replacement, but the result is not a correct run since the disequality constraint of the rule applied at the root is not satisfied.

In general, given a replacement $r[r']_{q_1\dots q_n}$ on a uniform run r of a $\texttt{TA}_{\texttt{ihom},\not\approx}$ A, the subruns of r whose disequality constraints may be affected by the replacement are the ones occurring in positions defined by prefixes of $q_1 \dots q_n$, i.e., the subruns $r|_{q_1\dots q_i}$ for $i \leq n$. The number of different such subruns is just n, although there could be many occurrences of them in r. For each rule application at each of such positions, the number of different disequality constraints appearing in them is bounded by the size of the automaton. Hence, $n \cdot |A|$ bounds the number of disequality constraints that might be falsified by replacements at $q_1 \dots q_n$. Suppose that we have K runs $r_1, \dots, r_K$ reaching the same state as $r|_{q_1\dots q_n}$. Suppose also that no two replacements $r[r_i]_{q_1\dots q_n}$, $r[r_j]_{q_1\dots q_n}$ falsify the same disequality. Obviously, in the case where $K > n \cdot |A|$, some $r[r_i]_{q_1\dots q_n}$ does not falsify any disequality, and hence, it is a correct run. However, we are interested in defining a K depending only on $|A|$ that guarantees the existence of a decreasing replacement satisfying all disequalities. To this end, we extend an argument from [CJ03] to prove that, for each $\texttt{TA}_{\texttt{ihom},\not\approx}$ A, there exists a natural number K exponentially bounded by $|A|$ satisfying the following property: given a run r, an abstract position $P = q_1 \dots q_n$, and runs $r_1, \dots, r_K$ smaller than $r|_P$, if all the replacements $r[r_1]_P, \dots, r[r_K]_P$ falsify different disequalities, then there exists a prefix P' of P and a run r' smaller than $r|_{P'}$ such that $r[r']_{P'}$ does not falsify any disequality.

In general, finding such runs $r_1, \dots, r_K$ satisfying the above conditions is not an easy task. The following notion of independence, given in [CJ03], helps to overcome this problem: the runs $r_1, \dots, r_K$ on terms $t_1, \dots, t_K$ are independent with respect to a set of positions $\mathcal{P}$ if, for each position $p \in \mathcal{P}$, either all the terms $t_1|_p, \dots, t_K|_p$ are identical, or they are pairwise different. It is not difficult to see that, under certain additional conditions, if the runs $r_1, \dots, r_K, \bar{r}$ are independent with respect to the set of positions that can be affected by constraints, then all the replacements $r[r_1]_P, \dots, r[r_K]_P$ falsify different disequalities, for any run r and abstract position P holding $r|_P = \bar{r}$.

The previous property allows the algorithm to discard a new generated run $\bar{r}$ if there exist certain $r_1, \dots, r_K$ chosen among the previously generated runs and such that $r_1, \dots, r_K, \bar{r}$ are independent (with respect to the set of positions $\mathcal{P}$ that can be affected by constraints).

In [CJ03], the existence of such $r_1, \dots, r_K$ for the given $\bar{r}$ is determined using the following combinatorial property: given a natural number K and a set of runs S with size $K^{|\mathcal{P}|} \cdot |\mathcal{P}|!$, there always exists an independent (with respect to $\mathcal{P}$) subset $\tilde{S}$ of S with size K. But the arguments of that paper are complicated because the property itself does not assure that $\bar{r}$ is part of $\tilde{S}$, making it difficult to adapt their arguments to our setting. For this reason, we define a new notion on sets of runs and natural numbers K, namely K-smallness. With this notion, when a set S is K-small, but the addition of a new run $\bar{r}$ gives rise to a non-K-small set $S \uplus \{\bar{r}\}$, it follows the existence of an independent subset $\tilde{S}$ of $S \uplus \{\bar{r}\}$ with size K and containing $\bar{r}$.

We summarize the algorithm deciding emptiness of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ as follows. We iteratively generate all possible runs in increasing order of size: new runs are constructed using as direct subruns the previous runs that have not been discarded as part of the minimum counterexample. A new run $\bar{r}$ is discarded if it produces non-K-smallness. Recall that this implies that $\bar{r}$ is the biggest run of an independent set of size K, and thus it is not part of a minimum counterexample. We prove that this iterative process is complete and terminates in exponential time.

4.2 Summary of known results

We first list some known hardness results that are related to our setting.

Proposition 4.1. *The following problems are EXPTIME-hard:*

- *The HOM problem (from [GGM11]).*
- *The problems of deciding equivalence and inclusion between the languages recognized by two* TA *(since universality of* TA *is EXPTIME-hard and can be reduced to equivalence and to inclusion [CDG+07]), and finiteness of their difference (since inclusion can be reduced to finite difference).*

In the remaining of this section we cite definitions and results from [GG13]. We also sketch some of the constructions of that paper, in order to give intuition for the stated complexities and for other constructions presented later in this chapter.

Definition 4.2. *A* tree automaton with disequality and HOM equality constraints*, $\mathtt{TA}_{\mathsf{hom},\not\approx}$ for short, is a tuple $A = \langle Q, \Sigma, F, \Delta \rangle$, where Q is a finite set of states, Σ is a signature, $F \subseteq Q$ is the subset of final states, and Δ is a finite set of rules of* *the form $l \xrightarrow{c} q$, where l is a term in $\mathcal{T}(\Sigma \uplus Q) \setminus Q$, interpreting the states of Q as nullary symbols, and c is a conjunction/set of unordered pairs of the form $\bar{p}_1 \not\approx \bar{p}_2$, for arbitrary positions $\bar{p}_1, \bar{p}_2$, or of the form $p_1 \approx p_2$, where p_1 and p_2 are different positions in $\mathsf{Pos}(l)$ satisfying $l(p_1) = l(p_2) \in Q$. Moreover, for all distinct positions p_1, p_2, p_3, if $p_1 \approx p_2$ and $p_2 \approx p_3$ occur in c, then $p_1 \approx p_3$ also occurs in c. When no atom of the form $\bar{p}_1 \not\approx \bar{p}_2$ occurs in the rules of A, we say that A is a $\mathtt{TA}_{\mathsf{hom}}$. When only atoms of the form $\bar{p}_1 \not\approx \bar{p}_2$ occur in the rules of A, and moreover, all the left-hand sides of the rules are flat and only their root position is labeled by an alphabet symbol, we say that A is a $\mathtt{TA}_{\not\approx}$.*

A run *of A on a term $t \in \mathcal{T}(\Sigma)$ is a partial mapping $r : \mathsf{Pos}(t) \to \Delta$ such that $r(\lambda)$ is defined, and satisfying the following conditions for each position p for which $r(p)$ is defined, say, as a rule $l \xrightarrow{c} q$. For each position $p' \in \mathsf{Pos}(l)$, it holds that $r(p.p')$ is defined if and only if $l(p') \in Q$. Moreover, if $l(p')$ is in Q, then $r(p.p')$ is a rule with the state $l(p')$ as right-hand side. Otherwise, if $l(p')$ is in Σ, then $l(p') = t(p.p')$. In addition, for each $(p_1 \approx p_2) \in c$, $t|_{p.p_1} = t|_{p.p_2}$ holds, and for each $(\bar{p}_1 \not\approx \bar{p}_2) \in c$, either one of $p.\bar{p}_1, p.\bar{p}_2$ is not in $\mathsf{Pos}(t)$ or both of them are and $t|_{p.\bar{p}_1} \neq t|_{p.\bar{p}_2}$ holds.*

The state reached *by r is the right-hand side of $r(\lambda)$. The run r is* accepting *if the state reached by r is in F. By $\mathcal{L}(A)$ we denote the* language recognized *by A, that is the set of terms t such that there exists an accepting run of A on t.*

Given a run r of A on a term t, and a position p such that $r(p)$ is defined, we define the subrun *$r|_p$ as the run of A on $t|_p$ described by $r|_p(p') = r(p.p')$. A run r on t is* deterministic *if, for each two positions p_1, p_2 where r is defined and $t|_{p_1} = t|_{p_2}$ holds, the subruns $r|_{p_1}$ and $r|_{p_2}$ are identical. We say that a $\mathtt{TA}_{\mathsf{hom},\not\approx}$ A* admits deterministic accepting runs *if, for each $t \in \mathcal{L}(A)$, there is a deterministic accepting run of A on t.*

The above definition differs from the one given in [GG13], where an atom $p_1 \approx p_2$ requires not only identity of subterms, but also identity of subruns. Nevertheless, it is also shown there that both conditions lead to the same expressiveness.

In order to be able to reason about the complexity of the constructions, the following notions of size are required.

Definition 4.3. *The size of a $\mathtt{TA}_{\mathsf{hom},\not\approx}$ $A = \langle Q, \Sigma, F, \Delta \rangle$, denoted $|A|$, is $|Q|$ plus the sum of sizes of all rules in Δ, where the size of a rule $l \xrightarrow{c} q$ is $|l| + |c| + 1$, and $|c|$ is the sum of the lengths of all the occurrences of positions in c.*

By $\mathsf{n}_{\approx}(A)$ (respectively, $\mathsf{n}_{\not\approx}(A)$) we denote the number of distinct equality atoms (respectively, disequality atoms) in the rules of A. By $\mathsf{h}_{\approx}(A)$ (respectively, $\mathsf{h}_{\not\approx}(A)$) we denote the maximum among the lengths of the positions occurring in equality atoms (respectively, disequality atoms) in the rules of A. By $\mathsf{Pos}_{\mathrm{lhs}}(A)$ we denote the set of positions of left-hand sides of rules of A, i.e., $\bigcup_{(l \xrightarrow{c} q) \in \Delta} \mathsf{Pos}(l)$. By $\mathsf{h}_{\mathrm{lhs}}(A)$ we denote the maximum among the heights of the left-hand sides of the rules of A, i.e., $\max\{|p| : p \in \mathsf{Pos}_{\mathrm{lhs}}(A)\}$. We just write $\mathsf{n}_{\approx}$, $\mathsf{n}_{\not\approx}$, $\mathsf{h}_{\approx}$, $\mathsf{h}_{\not\approx}$, $\mathsf{Pos}_{\mathrm{lhs}}$, $\mathsf{h}_{\mathrm{lhs}}$ when A is clear from the context.

The following proposition establishes that the class of $\mathtt{TA}_{\mathsf{hom}}$ can be used to represent images of regular tree languages under tree homomorphisms.

Proposition 4.4. *Let $A = \langle Q, \Sigma_1, F, \Delta \rangle$ be a $\mathtt{TA}$. Let $H : \mathcal{T}(\Sigma_1) \to \mathcal{T}(\Sigma_2)$ be a tree homomorphism.*

Then, a $\mathtt{TA}_{\mathsf{hom}}$ A' *can be computed satisfying* $\mathcal{L}(A') = H(\mathcal{L}(A))$ *with time and space in* $\mathcal{O}(|A'|)$, *where* $|A'| \leq (|A| \cdot (|H| + |H|^3))^2$.

Proof (Sketch of the construction). The construction of A' is a direct application of H to the rules of A. More precisely, for each rule $f(q_1, \ldots, q_m) \to q$ of A, a rule $H(f(q_1, \ldots, q_m)) \xrightarrow{c} q$ is added to A', where H is assumed to be the identity for all symbols in Q, and the constraint c has an atom $p_1 \approx p_2$ for each two distinct positions p_1, p_2 that are labeled by the same variable x_i in the term $H(f(x_1, \ldots, x_m))$. Hence, for each rule of A, a rule with size bounded by $|H| + |H|^3 + 1$ is added to A', where $|H|$ bounds the size of its left-hand side, and $|H|^3$ bounds the size of its constraint (note that there are up to $|H|^2/2$ different atoms, and each atom has size at most $2 \cdot |H|$). Thus, the summed size of the constructed rules is bounded by $|\Delta| \cdot (|H| + |H|^3 + 1)$, and hence, $|A| \cdot (|H| + |H|^3)$ bounds the size of the automaton constructed thus far. The last step of the construction of A' consists in eliminating the rules of the form $q_1 \to q_2$, since they are not allowed in the definition of $\mathtt{TA}_{\mathsf{hom}}$. This produces at most an additional quadratic increase in size. ■

In [GG13], the regularity of a $\mathtt{TA}_{\mathsf{hom}}$ A is characterized in terms of an operation that “linearizes” the automaton. This linearization depends on a natural number $\mathtt{h}$ and consists in computing a new automaton where the equality tests can only be satisfied if the height of the involved subterms is bounded by $\mathtt{h}$. In other words, the equality tests between terms with height greater than $\mathtt{h}$ are always falsified, even if the terms are indeed equal. It follows that there are only finitely many terms that can satisfy equality tests after the linearization.

Definition 4.5. *Let* $A = \langle Q, \Sigma, F, \Delta \rangle$ *be a* $\mathtt{TA}_{\mathsf{hom}}$. *Let* $\mathtt{h}$ *be a natural number. The* linearization *of* A *by* $\mathtt{h}$ *is the* $\mathtt{TA}_{\mathsf{hom}}$ $\langle Q, \Sigma, F, \Delta' \rangle$, *denoted* $\mathsf{linearize}(A, \mathtt{h})$, *where* Δ' *is the set of all rules of the form* $l[s_1]_{p_1} \ldots [s_n]_{p_n} \to q$ *such that:*

- *a rule of the form* $l \xrightarrow{c} q$ *occurs in* Δ,
- $p_1, \ldots, p_n$ *are the positions occurring in* c,
- *for each* $i \in \{1, \ldots, n\}$, s_i *is a term such that* $\mathsf{height}(s_i) \leq \mathtt{h}$ *and there is a run of* A *on it reaching the state* $l(p_i)$,
- *for each* $i, j \in \{1, \ldots, n\}$ *such that* $p_i \approx p_j$ *occurs in* c, $s_i = s_j$ *holds.*

It is straightforward that a linearization of any $\mathtt{TA}_{\mathsf{hom}}$ is computable and recognizes a regular tree language, since no equality constraints appear. It is also clear that $\mathcal{L}(A)$ includes the language of any of its linearizations. Moreover, in the case where $\mathcal{L}(A)$ is included in some of its linearizations, we can conclude that $\mathcal{L}(A)$ is regular. The following lemma and definition from [GG13] bound the $\mathtt{h}$ for which we should test such inclusion.

Lemma 4.6. *There effectively exists a polynomial* $\mathcal{P}$ *satisfying the following condition. Let* A *be a* $\mathtt{TA}_{\mathsf{hom}}$. *Let* $\check{h}$ *be* $2^{\mathcal{P}(|A|)}$.

Then, $\mathcal{L}(A)$ *is regular if and only if* $\mathcal{L}(A) \subseteq \mathcal{L}(\mathsf{linearize}(A, \check{h}))$.

Definition 4.7. *Let A be a $\mathtt{TA}_{\mathsf{hom}}$. By $\check{h}(A)$ we denote the function $2^{\mathcal{P}(|A|)}$ given by Lemma 4.6, and write $\check{h}$ when A is clear from the context.*

The previous lemma leads to a simple decision algorithm of HOM: given a $\mathtt{TA}_{\mathsf{hom}}$ A recognizing the image of a regular tree language under a tree homomorphism, $\mathcal{L}(A)$ is regular if and only if $\mathcal{L}(A) \subseteq \mathcal{L}(\mathsf{linearize}(A, \check{h}))$ if and only if $\mathcal{L}(A) \cap \overline{\mathcal{L}(\mathsf{linearize}(A, \check{h}))}$ is empty. Hence, the straightforward approach to decide regularity of $\mathcal{L}(A)$ consists in (i) constructing an automaton A' that recognizes $\overline{\mathcal{L}(\mathsf{linearize}(A, \check{h}))}$, then (ii) constructing an automaton A'' that recognizes $\mathcal{L}(A) \cap \mathcal{L}(A')$, and finally (iii) deciding emptiness of $\mathcal{L}(A'')$. The following two propositions summarize the results from [GG13] establishing that, given two $\mathtt{TA}_{\mathsf{hom}}$ A_1 and A_2, we can compute in exponential time a $\mathtt{TA}_{\mathsf{hom},\not\approx}$ recognizing $\mathcal{L}(A_1) \cap \overline{\mathcal{L}(A_2)}$. Note that this construction is more general than what is required for steps (i) and (ii) above, since the complement computed in (i) is done on a linearized automaton, i.e., on a $\mathtt{TA}_{\mathsf{hom}}$ as expressive as a $\mathtt{TA}$ since it has no equality constraints. However, by reasoning with two $\mathtt{TA}_{\mathsf{hom}}$, this result allows to tackle other decision problems on homomorphisms apart from regularity.

Proposition 4.8. *Let A be a $\mathtt{TA}_{\mathsf{hom}}$ over signature Σ.*

Then, a $\mathtt{TA}_{\not\approx}$ A' satisfying $\mathcal{L}(A') = \overline{\mathcal{L}(A)}$ can be computed with time and space in $2^{\mathcal{O}(\mathsf{maxar}(\Sigma)\cdot|\Sigma|\cdot|A|)}$, and such that A' admits deterministic accepting runs and the following bounds hold:

- $\mathsf{h}_{\not\approx}(A') = \mathsf{h}_{\approx}(A)$,
- $\mathsf{n}_{\not\approx}(A') = \mathsf{n}_{\approx}(A)$.

Proof (Sketch of the construction). The construction proceeds in two steps. In the first step, the rules of the $\mathtt{TA}_{\mathsf{hom}}$ A are flattened. Recall that rules of $\mathtt{TA}_{\mathsf{hom}}$ have arbitrary terms as left-hand sides, whereas rules of $\mathtt{TA}_{\not\approx}$ have flat left-hand sides. However, due to the restrictions imposed on the positions occurring in the equality constraints of $\mathtt{TA}_{\mathsf{hom}}$, in general it is not possible to convert a $\mathtt{TA}_{\mathsf{hom}}$ into an equivalent $\mathtt{TA}_{\mathsf{hom}}$ with only flat rules. For this reason, the class of automata $\mathtt{TA}_{\approx}$ is introduced, defined analogously to $\mathtt{TA}_{\not\approx}$ but with arbitrary equality constraints instead of arbitrary disequality constraints. Then, the transformation of the $\mathtt{TA}_{\mathsf{hom}}$ A into an equivalent, flattened $\mathtt{TA}_{\approx}$ A'' is straightforward:

- The states of A'' include all the states of A and also new states of the form $q_{r,p}$, where r is a rule of A and p is a position of the left-hand side of r (except λ and positions labeled by states). The final states of A'' coincide with the final states of A.
- For each rule $r = (f(t_1, \ldots, t_m) \xrightarrow{c} q)$ of A, a rule $f(s_1, \ldots, s_m) \xrightarrow{c} q$ is added to A'', where $s_i = q_{r,i}$ when t_i is not a state, and $s_i = t_i$ otherwise.
- For each state of the form $q_{r,p}$, a single rule of the form $f(s_1, \ldots, s_m) \to q_{r,p}$ is added to A'', where $f \in \Sigma^{(m)}$ is the symbol labeling the left-hand side l of the rule r at position p, i.e., $f = l(p)$, and $s_i = q_{r,p.i}$ when $l(p.i)$ is not a state, and $s_i = l(p.i)$ otherwise.

It is easy to see that the size of A'' is bounded by $4 \cdot |A|$: note that (i) the number of states of A'' is bounded by $|A|$, that (ii) the sum of sizes of the left-hand sides of the rules of A'' is bounded by twice the sum of sizes of the left-hand sides of the rules of A, and thus, by $2 \cdot |A|$, that (iii) the sum of sizes of the right-hand sides of the rules of A'', i.e., the number of rules of A'', is bounded by once the sum of sizes of the left-hand sides of the rules of A, and thus, by $|A|$, and finally that (iv) A'' contains exactly the same constraints as A.

The second step computes the complement of the $\mathtt{TA}_{\approx}$ A'' to obtain the desired $\mathtt{TA}_{\not\approx}$ A'. The idea of this last construction is to define an automaton A' whose runs compute sets of unreachable states of A''. Hence, A' accepts those terms reaching a set of states that contains all the final states of A''. More precisely, the states of A' are sets of states of A'', and the rules of A' ensure that there exists a run of A' on a term t reaching a state S if and only if, for each state $q \in S$, there is no run of A'' on t reaching q. Therefore, the rules of A' are of the form $f(S_1, \ldots, S_m) \xrightarrow{D} S$, where $S_1, \ldots, S_m, S$ are sets of states of A'', and D is a disequality constraint such that, for each of its atoms $p_1 \not\approx p_2$, the atom $p_1 \approx p_2$ occurs in some rule of A''. Moreover, the right-hand-side state S only contains states q of A'' that are guaranteed to be unreachable given the information provided by f, $S_1, \ldots, S_m$, and D, that is, states q satisfying the following property: for each rule $g(q_1, \ldots, q_n) \xrightarrow{c} q$ of A'', either the alphabet symbol g is not f, or a state q_i occurs in S_i (and thus, by induction it is unreachable), or $p_1 \approx p_2$ occurs in c and its negation $p_1 \not\approx p_2$ occurs in D. The number of states of A' is bounded by $2^{|Q''|}$, where Q'' is the set of states of A'', the number of rules of A' is bounded by $2^{|Q''| \cdot (\mathsf{maxar}(\Sigma)+1)} \cdot |\Sigma| \cdot 2^{\mathsf{n}_{\approx}(A'')}$, and the size of each of such rules is bounded by $|A''|$. Thus, $|A'|$ can be bounded by $2^{|Q''|} + |A''| \cdot 2^{|Q''| \cdot (\mathsf{maxar}(\Sigma)+1)} \cdot |\Sigma| \cdot 2^{\mathsf{n}_{\approx}(A'')} \leq 2^{|A|} + 4 \cdot |A| \cdot 2^{|A| \cdot (\mathsf{maxar}(\Sigma)+1)} \cdot |\Sigma| \cdot 2^{\mathsf{n}_{\approx}(A)}$.

The complexity for the construction of the $\mathtt{TA}_{\not\approx}$ A' from the $\mathtt{TA}_{\mathsf{hom}}$ A given in the statement is a weaker expression with respect to the previous explanations, but more convenient for further reasonings. ■

Proposition 4.9. *Let A_1 be a $\mathtt{TA}_{\mathsf{hom}}$. Let A_2 be a $\mathtt{TA}_{\not\approx}$ over the same signature as A_1 and such that admits deterministic accepting runs.*

Then, a $\mathtt{TA}_{\mathsf{hom},\not\approx}$ A satisfying $\mathcal{L}(A) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$ can be computed with time and space in $2^{\mathcal{O}((|\mathsf{Pos}_{\mathrm{lhs}}(A_1)|+\mathsf{n}_{\approx}(A_1)+|\mathsf{Pos}_{\mathrm{lhs}}(A_1)|\cdot\mathsf{n}_{\not\approx}(A_2))\cdot\log(|A_1|\cdot|A_2|))}$, and such that the following bounds hold:

- $\mathsf{h}_{\approx}(A) = \mathsf{h}_{\approx}(A_1)$,
- $\mathsf{n}_{\approx}(A) = \mathsf{n}_{\approx}(A_1)$,
- $\mathsf{h}_{\not\approx}(A) \leq \mathsf{h}_{\mathrm{lhs}}(A_1) + \mathsf{h}_{\not\approx}(A_2)$,
- $\mathsf{n}_{\not\approx}(A) \leq |\mathsf{Pos}_{\mathrm{lhs}}(A_1)| \cdot \mathsf{n}_{\not\approx}(A_2)$,
- $\mathsf{Pos}_{\mathrm{lhs}}(A) = \mathsf{Pos}_{\mathrm{lhs}}(A_1)$.

Proof (Sketch of the construction). This algorithm is a variation of the typical product construction. The basic idea is that the states of A are pairs of the form $\langle q_1, q_2 \rangle$, where q_1 is a state of A_1 and q_2 is a state of A_2, and the rules of A simulate both automata A_1 and A_2 simultaneously. The only difficulty stems from the fact that the rules of A_1 are not necessarily flat, and moreover, as argued in the proof of Proposition 4.8, in general they cannot be flattened since the definition of HOM equality constraint imposes that the two positions involved in an equality atom $p_1 \approx p_2$ must be defined in the left-hand side of the rule where such atom occurs. Furthermore, the symbols labeling the left-hand side of the rule at such positions p_1, p_2 must be identical states (this conditions is why we require that A_2 admits deterministic accepting runs: determinism guarantees that the same pair $\langle q_1, q_2\rangle$ can be reached at each identical subterm in accepting runs).

To overcome the previous problem, it suffices to simulate an execution of A_2 on the left-hand side of each rule in order to define which state of A_2 is reached at the root. Moreover, the disequality constraints that have to be satisfied along such simulated execution must be added to the constraint of the rule. For instance, if the simulated execution of A_2 has tested the disequality $p_1 \not\approx p_2$ at a position p of the left-hand side of the rule (guessing that it is satisfied), then we must add the atom $p.p_1 \not\approx p.p_2$ to the rule to guarantee that the simulation is valid (i.e., that the guess is correct). Note that there might be several possible executions of A_2 on the same rule, and all of them must be considered independently.

In summary, each rule of A is obtained from a rule of A_1 by (i) replacing each state of A_1 by a pair of states $\langle q_1, q_2\rangle$, where q_1 is a state of A_1 and q_2 is a state of A_2, satisfying that the same state pair is used at positions p_1, p_2 if the rule has the equality constraint $p_1 \approx p_2$, (ii) simulating an execution of A_2 on the left-hand side of the rule in order to define the state of A_2 occurring in the right-hand side of the rule, and (iii) extending the constraint of the rule with the disequality atoms that have been tested along the simulated execution of A_2. To bound the number of rules of A, note that the number of possible left-hand sides is bounded by $(|\Sigma| + |Q_1| \cdot |Q_2|)^{|\mathsf{Pos}_{\mathrm{lhs}}(A_1)|}$, the number of possible right-hand sides is bounded by $|Q_1| \cdot |Q_2|$, and the number of possible constraints is bounded by $2^{\mathsf{n}_{\approx}(A_1)+|\mathsf{Pos}_{\mathrm{lhs}}(A_1)|\cdot\mathsf{n}_{\not\approx}(A_2)}$, where Q_1 is the set of states of A_1, Q_2 is the set of states of A_2, and Σ is the underlying signature. Hence, the number of rules of A is bounded by $(|A_1| \cdot |A_2|)^{|\mathsf{Pos}_{\mathrm{lhs}}(A_1)|+1} \cdot 2^{\mathsf{n}_{\approx}(A_1)+|\mathsf{Pos}_{\mathrm{lhs}}(A_1)|\cdot\mathsf{n}_{\not\approx}(A_2)}$, assuming that $|\Sigma|$ is bounded by $|A_1|$. Finally, note that the size of each rule of A is bounded by the maximum size of the rules of A_1 (in turn bounded by $|A_1|$) plus the maximum size of the disequality constraints that can be defined (i.e., $|\mathsf{Pos}_{\mathrm{lhs}}(A_1)| \cdot \mathsf{n}_{\not\approx}(A_2) \cdot 2 \cdot (\mathsf{h}_{\mathrm{lhs}}(A_1) + \mathsf{h}_{\not\approx}(A_2))$), and that A has $|Q_1| \cdot |Q_2|$ states.

The complexity for the construction of the $\mathtt{TA}_{\mathsf{hom},\not\approx}$ A from the $\mathtt{TA}_{\mathsf{hom}}$ A_1 and the $\mathtt{TA}_{\not\approx}$ A_2 given in the statement is a weaker expression with respect to the previous explanations, but more convenient for further reasonings. ■

We cite one additional result from [GG13] that reasons about the finiteness of the language recognized by $\mathtt{TA}_{\mathsf{hom},\not\approx}$. Intuitively, the authors define an exponential function $\tilde{h}$ such that, for any given $\mathtt{TA}_{\mathsf{hom},\not\approx}$ A, the language recognized by A is finite if and only if there is no term in $\mathcal{L}(A)$ with height greater than $\tilde{h}(A)$. Note that, in this way, in order to decide finiteness of $\mathcal{L}(A)$, it suffices to construct a new $\mathtt{TA}_{\mathsf{hom},\not\approx}$ A' recognizing the language $\{t \in \mathcal{L}(A) \mid \mathsf{height}(t) > \tilde{h}(A)\}$ and test emptiness of $\mathcal{L}(A')$ instead. The following proposition formally establishes the complexity of computing such A'.

Proposition 4.10. *Let A be a* $\mathtt{TA}_{\mathsf{hom},\not\approx}$.

Then, a $\mathtt{TA}_{\mathsf{hom},\not\approx}$ *A' satisfying that $\mathcal{L}(A')$ is empty if and only if $\mathcal{L}(A)$ is finite can be computed with time and space in* $2^{\mathcal{O}((\mathsf{n}_{\approx}(A)+\mathsf{n}_{\not\approx}(A))\cdot\max\{\mathsf{h}_{\mathrm{lhs}}(A),\mathsf{h}_{\not\approx}(A)\}\cdot|\mathsf{Pos}_{\mathrm{lhs}}(A)|\cdot\log|A|)}$*, and such that the following bounds hold:*

- $\mathsf{h}_{\approx}(A') = \mathsf{h}_{\approx}(A)$,
- $\mathsf{n}_{\approx}(A') = \mathsf{n}_{\approx}(A)$,
- $\mathsf{h}_{\not\approx}(A') = \mathsf{h}_{\not\approx}(A)$,
- $\mathsf{n}_{\not\approx}(A') = \mathsf{n}_{\not\approx}(A)$,
- $\mathsf{Pos}_{\mathrm{lhs}}(A') = \mathsf{Pos}_{\mathrm{lhs}}(A)$.

Proof (Sketch of the construction). The function $\tilde{h}$ mentioned above is more precisely defined in [GG13] as $\tilde{h}(A) = |\Delta| \cdot (h^2 + h) \cdot (2 \cdot n \cdot h)^{4\cdot n\cdot h}$, where Δ is the set of rules of A, $h = \max\{\mathsf{h}_{\mathrm{lhs}}(A), \mathsf{h}_{\not\approx}(A)\}$, and $n = \mathsf{n}_{\approx}(A) + \mathsf{n}_{\not\approx}(A)$. This definition guarantees that any term accepted by A with height greater than $\tilde{h}(A)$ can be pumped to obtain another higher term also accepted by A. Thus, the automaton A' of the statement can be defined as the $\mathtt{TA}_{\mathsf{hom},\not\approx}$ recognizing $\{t \in \mathcal{L}(A) \mid \mathsf{height}(t) > \tilde{h}(A)\}$, and such A' can be obtained from A straightforwardly: it suffices that the states record the height of the recognized term (up to $\tilde{h}(A)+1$), and that no state with recorded height smaller than $\tilde{h}(A)+1$ is final. The size of A' is bounded by $|A| \cdot (\tilde{h}(A)+2)^{|\mathsf{Pos}_{\mathrm{lhs}}(A)|}$. ■

4.3 TA with disequality and implicit HOM equality constraints

The notion of $\mathtt{TA}_{\mathsf{hom}}$ from [GG13] detailed in Section 4.2 is a straightforward application of a homomorphism to the rules of a TA. Its main disadvantage is that dealing with constrained rules makes the presentation of technical proofs a laborious task. For this reason, we define an equivalent kind of automata in which equality constraints are implicitly encoded in the left-hand sides of the rules. Intuitively, a state that appears duplicated in the left-hand side l of a rule, implicitly forces an equality test between all the positions of l where such state occurs.

Definition 4.11. *A* tree automaton with disequality and implicit HOM equality constraints, $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ *for short, is a tuple $A = \langle Q, \Sigma, F, \Delta\rangle$, where Q is a finite set of states, Σ is a signature, $F \subseteq Q$ is the subset of final states, and Δ is a finite set of rules of the form $l \xrightarrow{c} q$, where l is a term in $\mathcal{T}(\Sigma \uplus Q) \setminus Q$, interpreting the states of Q as nullary symbols, and c, called the disequality constraint of the rule, is a conjunction/set of unordered pairs of the form $\bar{p}_1 \not\approx \bar{p}_2$, for arbitrary positions $\bar{p}_1, \bar{p}_2$.*

A run *of A on a term $t \in \mathcal{T}(\Sigma)$ is a partial mapping $r : \mathsf{Pos}(t) \to \Delta$ such that $r(\lambda)$ is defined, and satisfying the following conditions for each position p for which $r(p)$ is defined, say, as a rule $l \xrightarrow{c} q$. For each position $p' \in \mathsf{Pos}(l)$, it holds that $r(p.p')$ is defined if and only if $l(p') \in Q$. Moreover, if $l(p')$ is in Q, then $r(p.p')$ is a rule with the state $l(p')$ as right-hand side. Otherwise, if $l(p')$ is in Σ, then $l(p') = t(p.p')$.*

In addition, (i) for each two positions $p_1, p_2 \in \mathsf{Pos}(l)$ satisfying $l(p_1) = l(p_2) \in Q$, $t|_{p.p_1} = t|_{p.p_2}$ holds, and (ii) for each two positions $\bar{p}_1, \bar{p}_2$ satisfying $(\bar{p}_1 \not\approx \bar{p}_2) \in c$, either one of $p.\bar{p}_1, p.\bar{p}_2$ is not in $\mathsf{Pos}(t)$ or both of them are and $t|_{p.\bar{p}_1} \neq t|_{p.\bar{p}_2}$ holds. We say that r is a weak run *when conditions (i) and (ii) are not enforced. Moreover, since t can be deduced from r, we often do not make explicit t and just say that r is a (weak) run of A.*

The state reached *by a weak run r is the right-hand side of $r(\lambda)$, and we say that r is* accepting *if the state reached by r is in F. By $\mathcal{L}(A)$ we denote the* language recognized *by A, that is the set of terms t for which there exists an accepting run of A on t. Given a weak run r of A on a term t, we define $\mathsf{Pos}(r)$ as $\mathsf{Pos}(t)$, $\mathsf{height}(r)$ as $\mathsf{height}(t)$, and $\mathsf{term}(r)$ as t. Moreover, given a position p such that $r(p)$ is defined, we define the weak* subrun *$r|_p$ as the weak run of A on $t|_p$ described by $r|_p(p') = r(p.p')$. Note that if r is a run, then so is $r|_p$. Given two weak runs r_1, r_2 and a position p such that $r_1(p)$ is defined and $r_1|_p, r_2$ reach the same state, we define the* replacement *$r_1[r_2]_p$ as the weak run r on $\mathsf{term}(r_1)[\mathsf{term}(r_2)]_p$ described as follows: $r(p') = r_2(\hat{p})$ if p' is of the form $p.\hat{p}$, and $r(p') = r_1(p')$, otherwise.*

The notions of size of automata, and $\mathsf{n}_{\not\approx}$, $\mathsf{h}_{\not\approx}$, $\mathsf{h}_{\mathrm{lhs}}$ and $\mathsf{Pos}_{\mathrm{lhs}}$ are defined identically for $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ as for $\mathtt{TA}_{\mathsf{hom},\not\approx}$ (see Definition 4.3).

Example 4.12. *In order to give intuition on $\mathtt{TA}_{\mathsf{ihom},\not\approx}$, and in particular, on their implicit equality constraints, we define a $\mathtt{TA}_{\mathsf{hom},\not\approx}$ A and then transform it into a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ A' that recognizes the same language. Consider a signature Σ consisting of a nullary symbol a, and binary symbols f and g. Let $A = \langle Q, \Sigma, F, \Delta\rangle$, where $Q = \{q, q_{\mathsf{accept}}\}$, $F = \{q_{\mathsf{accept}}\}$, and $\Delta = \{a \to q,\ f(f(q,q),q) \xrightarrow{1.1\approx 2} q,\ g(q,q) \xrightarrow{1.1\not\approx 2} q_{\mathsf{accept}}\}$. Note that the left-hand side of the second rule is not flat, and that the disequality atom of the third rule involves a position which is not defined in its left-hand side. Now we show how to transform A into an equivalent $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ $A' = \langle Q', \Sigma, F, \Delta'\rangle$. Since duplication of states occurring in left-hand sides of rules of a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ implicitly encode equality constraints, in order to preserve the recognized language we need to introduce a new state q' as a synonym for q. Hence, $Q' = \{q, q', q_{\mathsf{accept}}\}$. To present the rules of Δ', we use the compact notation $l \xrightarrow{c} q_1|q_2$ to simultaneously denote the rules $l \xrightarrow{c} q_1$ and $l \xrightarrow{c} q_2$. In this way, $\Delta' = \{a \to q|q',\ f(f(q,q'),q) \to q|q',\ g(q,q') \xrightarrow{1.1\not\approx 2} q_{\mathsf{accept}}\}$. Note that the equality atom $1.1 \approx 2$ no longer appears explicitly, since it is implied by the fact that the state q occurs at positions 1.1 and 2 in the left-hand side $f(f(q,q'),q)$.*

The implicit equality constraints of a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ can be assumed to ask for equality not only of subterms, but also of subruns. Runs holding this property are called uniform.

Definition 4.13. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. A run r of A is called* uniform *if, for each positions p, p_1, p_2 such that $r(p)$ is defined as a rule $l \xrightarrow{c} q$ satisfying $l(p_1) = l(p_2) \in Q$, $r|_{p.p_1} = r|_{p.p_2}$ holds.*

Lemma 4.14. *Any run of a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ A can be transformed into a uniform run on the same term and reaching the same state.*

With the previous lemma it is clear that the classes of languages recognizable by $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ with runs and with uniform runs coincide. We straightforwardly extend the notion of uniform runs to the setting of weak runs, and call them uniform weak runs. Note that weak runs do not have to satisfy the disequality constraints or the implicit equality constraints occurring in the rules applied, whereas a uniform weak run guarantees that equality constraints are satisfied. For this reason, uniform weak runs do not allow an equivalent of Lemma 4.14, i.e., it is possible that a weak run cannot be transformed into a uniform weak run recognizing the same term and reaching the same state. In fact, note that the class of languages recognizable by $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ with weak runs coincides with the class of regular tree languages, and that the class of languages recognizable by $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ with uniform weak runs is not regular.

The following definitions and lemmas show equivalence in expressiveness between $\mathtt{TA}_{\mathsf{hom},\not\approx}$ and $\mathtt{TA}_{\mathsf{ihom},\not\approx}$.

Definition 4.15 (transformation of $\mathtt{TA}_{\mathsf{hom},\not\approx}$ into $\mathtt{TA}_{\mathsf{ihom},\not\approx}$)**.** *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{TA}_{\mathsf{hom},\not\approx}$. We define the $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ $A' = \langle Q', \Sigma, F', \Delta'\rangle$ from A as the result of the following construction. Let k be $|\mathsf{Pos}_{\mathrm{lhs}}(A)|$. The set of states Q' is defined as $\{q^i \mid 1 \le i \le k,\ q \in Q\}$. The set of final states F' is defined as $\{q^i \mid 1 \le i \le k,\ q \in F\}$. In order to define Δ', we have to choose, for each rule $l \xrightarrow{c} q$ in Δ, a term $\mathsf{rename}(l, c)$ which is obtained by replacing each occurrence of a state of Q in l by a state of Q' according to the following condition: an occurrence of a state q in l is replaced by a state of the form q^i, for $1 \le i \le k$, and two occurrences of states at distinct positions p_1, p_2 in l are replaced by the same state if and only if $p_1 \approx p_2$ occurs in c. Once $\mathsf{rename}(l, c)$ is defined for each rule $l \xrightarrow{c} q$ in Δ, we define Δ' as the set of rules $\{\mathsf{rename}(l, c) \xrightarrow{c'} q^i \mid (l \xrightarrow{c} q) \in \Delta,\ c' = \{(\bar{p}_1 \not\approx \bar{p}_2) \in c\},\ 1 \le i \le k\}$.*

Lemma 4.16. *Let A be a $\mathtt{TA}_{\mathsf{hom},\not\approx}$. Let A' be the $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ obtained from A with the construction of Definition 4.15.*

Then, $\mathcal{L}(A') = \mathcal{L}(A)$ and the following bounds hold:

- $|A'| \le |A| \cdot |\mathsf{Pos}_{\mathrm{lhs}}(A)|$,
- $\mathsf{h}_{\not\approx}(A') = \mathsf{h}_{\not\approx}(A)$,
- $\mathsf{n}_{\not\approx}(A') = \mathsf{n}_{\not\approx}(A)$,
- $\mathsf{Pos}_{\mathrm{lhs}}(A') = \mathsf{Pos}_{\mathrm{lhs}}(A)$.

Moreover, A' can be computed with time and space in $\mathcal{O}(|A'|)$.

Example 4.17 (continuation of Example 4.12)**.** *The construction of the $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ A' from the $\mathtt{TA}_{\mathsf{hom},\not\approx}$ A detailed in Example 4.12 is done manually and obtains a rather small automaton. Here, we apply the construction of Definition 4.15 in order to obtain another $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ $A'' = \langle Q'', \Sigma, F'', \Delta''\rangle$ recognizing the same language as A. First, note that $|\mathsf{Pos}_{\mathrm{lhs}}(A)| = |\{\lambda,\ 1,\ 1.1,\ 1.2,\ 2\}| = 5$. Hence, the states in Q'' are of the form q^i and q^i_{accept}, for $1 \le i \le 5$, and $F'' = \{q^i_{\mathsf{accept}} \mid 1 \le i \le 5\}$. Second, in order to obtain Δ'', we need to define rename for each of the rules of A. We arbitrarily choose the following: $\mathsf{rename}(a, \emptyset) = a$, $\mathsf{rename}(f(f(q, q), q), 1.1 \approx 2) = f(f(q^1, q^2), q^1)$, and $\mathsf{rename}(g(q, q), 1.1 \not\approx 2) = g(q^1, q^2)$. Finally, using this definition for rename, the rules of Δ'' are of the form $a \to q^i$, $f(f(q^1, q^2), q^1) \to q^i$, and $g(q^1, q^2) \xrightarrow{1.1 \not\approx 2} q^i_{\mathsf{accept}}$,* *for $1 \leq i \leq 5$. It is easy to see that many of the states introduced in A'' are useless. In particular, all the states q^i with $i > 2$ are useless. Moreover, even though all the states of the form q^i_{accept} are useful, note that having just one of them suffices. By cleaning the useless and redundant states, we would obtain a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ that is essentially the one constructed in Example 4.12.*

Definition 4.18 (transformation of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ into $\mathtt{TA}_{\mathsf{hom},\not\approx}$)**.** *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. We define the $\mathtt{TA}_{\mathsf{hom},\not\approx}$ A' from A as $\langle Q, \Sigma, F, \Delta'\rangle$, where Δ' is the set of rules $\{l \xrightarrow{c'} q \mid (l \xrightarrow{c} q) \in \Delta,\ c' = c \uplus \{p_1 \approx p_2 \mid p_1, p_2 \in \mathsf{Pos}_Q(l),\ p_1 \neq p_2,\ l(p_1) = l(p_2)\}\}$.*

Lemma 4.19. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let A' be the $\mathtt{TA}_{\mathsf{hom},\not\approx}$ obtained from A with the construction of Definition 4.18.*

Then, $\mathcal{L}(A') = \mathcal{L}(A)$ and the following bounds hold:

- $|A'| \leq |A| + |A| \cdot |\mathsf{Pos}_{\mathrm{lhs}}(A)|^2 \cdot \mathsf{h}_{\mathrm{lhs}}(A)$,
- $\mathsf{h}_{\approx}(A') \leq \mathsf{h}_{\mathrm{lhs}}(A)$,
- $\mathsf{n}_{\approx}(A') \leq |\mathsf{Pos}_{\mathrm{lhs}}(A)|^2/2$,
- $\mathsf{h}_{\not\approx}(A') = \mathsf{h}_{\not\approx}(A)$,
- $\mathsf{n}_{\not\approx}(A') = \mathsf{n}_{\not\approx}(A)$,
- $\mathsf{Pos}_{\mathrm{lhs}}(A') = \mathsf{Pos}_{\mathrm{lhs}}(A)$.

Moreover, A' can be computed with time and space in $\mathcal{O}(|A'|)$.

4.4 Independent sets

The contents presented in this section is rather abstract and its results seem, at first look, to fit better in a handbook on combinatorics than in a work on tree automata. Nevertheless, in [CJ03], similar notions were needed to prove EXPTIME-completeness of emptiness for tree automata with disequality constraints. We explain the differences with such notions after Definition 4.20.

4.4.1 Independent sets of tuples

We assume a given set (the universe) U and a natural number n, and work with n-tuples $t = \langle e_1, \ldots, e_n\rangle$ of elements of U. For such a tuple t, with $t[i]$ we denote the i'th component e_i. We denote the set of all such possible tuples as T. For a given finite set of tuples, we are interested on finding a "big" subset which is independent according to the following definition.

Definition 4.20. *A finite set of tuples $\{t_1, \ldots, t_k\} \subseteq T$ is independent if for all $i \in \{1, \ldots, n\}$, either all the elements $t_1[i], \ldots, t_k[i]$ are the same, or the elements $t_1[i], \ldots, t_k[i]$ are pairwise different.*

Note that, if a set is independent, then any of its subsets also is.

In [CJ03], it is proved for a fixed natural number K that, given a set S with $K^n \cdot n!$ tuples, there effectively exists an independent subset $\tilde{S}$ of S with size K. This fact is used in [CJ03] to decide emptiness of tree automata with disequality constraints in exponential time. In order to produce simpler arguments in our setting, we need more than just the existence of such $\tilde{S}$. We also need to ensure that a certain tuple t in S is also in $\tilde{S}$. As a first step, we note that, since all tuples of an independent subset coincide at certain components, we can restrict our search of such $\tilde{S}$ to subsets of S whose tuples already coincide with t at some fixed components.

Definition 4.21. *Let S, t, I be such that $t \in S \subseteq T$ and $I \subseteq \{1, \dots, n\}$. We define the set of tuples $\mathsf{coincident}(S, t, I)$ as $\{t' \in S \mid \forall i \in I : t'[i] = t[i]\}$.*

Note that, if $t' \in \mathsf{coincident}(S, t, I)$, then $\mathsf{coincident}(S, t', I) = \mathsf{coincident}(S, t, I)$. For a natural number K, we define a counting property on sets of tuples, namely K-smallness, that will be useful to construct an independent set of size K containing a specific tuple t.

Definition 4.22. *Let K be a natural number. Let $S \subseteq T$ be a set of tuples. We say that S is K-*small *if the following statement holds:*

$$\forall t \in S : \forall I \subsetneq \{1, \dots, n\} : |\mathsf{coincident}(S, t, I)| < K^{n-|I|} \cdot (n - |I|)!$$

Example 4.23. *Consider tuples of $n = 3$ elements with $\mathbb{N}$ as the underlying universe. Let $S = \{t_1 = \langle 1, 1, 1\rangle,\ t_2 = \langle 1, 2, 2\rangle,\ t_3 = \langle 1, 2, 3\rangle\}$, and let $K = 3$. In order to see whether S is K-small, we need to consider every tuple $t \in S$ and every strict subset I of the indexes $\{1, 2, 3\}$. We first consider any such I with $|I| \leq 1$, and observe that the statement of K-smallness is trivially satisfied: $|\mathsf{coincident}(S, t, I)| \leq |S| = 3$ for any $t \in S$, which is strictly less than $K^{n-|I|} \cdot (n - |I|)! \geq 3^{3-1} \cdot (3 - 1)! = 18$. We now consider any such I with $|I| = 2$. In this case, the strict upper bound imposed by the definition of K-smallness is $K^{n-|I|} \cdot (n - |I|)! = 3^{3-2} \cdot (3 - 2)! = 3$. It is easy to see that the bound is again satisfied: for any $t \in S$, observe that if $2 \in I$ it follows $|\mathsf{coincident}(S, t, I)| \leq 2$ since $t_1[2] \neq t_2[2] = t_3[2]$, and if $3 \in I$ it follows $|\mathsf{coincident}(S, t, I)| = 1$ since $t_1[3]$, $t_2[3]$, $t_3[3]$ are pairwise different. Since the bound is satisfied in all the cases, S is K-small.*

The following lemma states that K-small sets are indeed "small".

Lemma 4.24. *Let K be a natural number. Let $S \subseteq T$ be a non-empty K-small set. Then, $|S| < K^n \cdot n!$.*

Proof. Note that for any tuple $t \in S$, $S = \mathsf{coincident}(S, t, \emptyset)$ holds. Thus, $|S| = |\mathsf{coincident}(S, t, \emptyset)| < K^{n-|\emptyset|} \cdot (n - |\emptyset|)! = K^n \cdot n!$. ■

The following lemma holds by Definitions 4.21 and 4.22.

Lemma 4.25. *Let K be a natural number. Let $S \subseteq T$ be a set of tuples. Then, checking whether S is K-small can be done with at most $|S|^2 \cdot 2^n \cdot n$ comparisons between elements occurring in the tuples of S.*

In order to show that detecting K-smallness is enough for our purposes, we prove in Lemma 4.31 that, given a set S and a tuple $t \in S$ such that $S \setminus \{t\}$ is K-small but S is not, there always exists an independent subset $\tilde{S} \subseteq S$ of size K and including t. To this end, as a first ingredient, we relate the existence of an independent subset of S with the existence of an edge-free subset of nodes of a graph. In the literature, edge-free subsets of nodes are simply called independent. Here, we use this other name in order to avoid confusion with our notion of independent sets of tuples.

Definition 4.26. *Let $G = \langle V, E\rangle$ be an undirected graph. Let $\tilde{V}$ be a subset of V. We say that $\tilde{V}$ is* edge-free *in G if each two nodes of $\tilde{V}$ are not connected, i.e., if $\{(u, v) \mid u, v \in \tilde{V}\} \cap E = \emptyset$.*

The graph where we want to find edge-free subsets of nodes is defined to have $\mathsf{coincident}(S, t, I)$ as its set of nodes, for a fixed I, and to have an edge between each two different tuples t_1, t_2 if and only if t_1 and t_2 coincide at some component not in I. This is defined formally as follows.

Definition 4.27. *Let S, t, I be such that $t \in S \subseteq T$ and $I \subseteq \{1, \ldots, n\}$. We define $\mathsf{graph}(S, t, I)$ as the undirected graph $G = \langle V, E\rangle$ with $V = \mathsf{coincident}(S, t, I)$ and $E = \{(t_1, t_2) \in V^2 \mid t_1 \neq t_2 \wedge \exists i \in \{1, \ldots, n\} \setminus I : t_1[i] = t_2[i]\}$.*

Example 4.28. *Following Example 4.23, consider $\mathsf{graph}(S, t_2, I = \{2\})$. Note that its set of nodes is $\{t_2, t_3\}$ since $t_1[2] \neq t_2[2] = t_3[2]$. Also note that the graph has the edge (t_2, t_3) since these tuples coincide at their first component, which is an index not included in I. It follows that the graph has no edge-free subset of nodes with size greater than 1. Consider now $\mathsf{graph}(S, t_2, I' = \{1, 2\})$, and note that its set of nodes is $\{t_2, t_3\}$ since $t_2[1] = t_3[1]$ and $t_1[2] \neq t_2[2] = t_3[2]$, and its set of edges is empty since $t_2[3] \neq t_3[3]$, where 3 is the only index not included in I'. Thus, the whole graph is edge-free. It is easy to conclude that all the nodes that conform this graph correspond to an independent subset of S: the nodes/tuples coincide at their respective components $1, 2$ since the graph has been defined with $I' = \{1, 2\}$, and moreover, the tuples are pairwise different at all other components, i.e., at component 3, since the whole graph is edge-free.*

The following trivial lemma formally establishes the relation between independent sets of tuples and edge-free sets of nodes of a graph.

Lemma 4.29. *Let S, t, I be such that $t \in S \subseteq T$ and $I \subseteq \{1, \ldots, n\}$. Let $\tilde{S}$ be a subset of $\mathsf{coincident}(S, t, I)$ which is edge-free in $\mathsf{graph}(S, t, I)$.*

Then, $\tilde{S}$ is independent.

In the proof of Lemma 4.31, the existence of an edge-free subset of nodes is concluded using, as a last ingredient, the following simple and well-known statement from graph theory, where $\mathsf{maxdegree}(G)$ denotes the maximum among all degrees of nodes of G.

Lemma 4.30. *Let $G = \langle V, E\rangle$ be an undirected graph. Let u be a node of G.*

Then, there exists a subset $\tilde{V}$ of V which is edge-free in G, includes u, and satisfies $|\tilde{V}| = \left\lceil \frac{|V|}{\mathsf{maxdegree}(G)+1} \right\rceil$.

Lemma 4.31. *Let K be a natural number. Let $S \subseteq T$ be a set of tuples which is not K-small. Let $t \in S$ be a tuple satisfying that $S \setminus \{t\}$ is K-small.*

Then, there exists an independent set $\tilde{S}$ of tuples such that $t \in \tilde{S} \subseteq S$ and $|\tilde{S}| \geq K$.

Proof. Since S is not K-small, there is a set $I \subsetneq \{1, \ldots, n\}$ and a tuple $t' \in S$ satisfying $|\mathsf{coincident}(S, t', I)| \geq K^{n-|I|} \cdot (n-|I|)!$. Among all the possible I's satisfying such a condition, we choose one maximal in size.

Note that t belongs to $\mathsf{coincident}(S, t', I)$, since otherwise, $\mathsf{coincident}(S, t', I) = \mathsf{coincident}(S \setminus \{t\}, t', I)$, and hence, $|\mathsf{coincident}(S \setminus \{t\}, t', I)| \geq K^{n-|I|} \cdot (n-|I|)!$, which implies that $S \setminus \{t\}$ is not K-small, contradicting the assumptions of the lemma. Therefore, $\mathsf{coincident}(S, t', I) = \mathsf{coincident}(S, t, I)$. In other words, the mentioned tuple t' can be assumed to be t.

In the case $|I| = n-1$, we have $|\mathsf{coincident}(S, t, I)| \geq K^{n-(n-1)} \cdot (n-(n-1))! = K$. In this case, note that $\mathsf{coincident}(S, t, I)$ itself is necessarily an independent set because its tuples coincide in all components but one, and hence they must be all pairwise different at such component. Thus, we conclude by defining $\tilde{S}$ as $\mathsf{coincident}(S, t, I)$.

At this point, we assume $|I| < n-1$. Under this assumption, by the maximality selection of I, the following condition holds:

$$\forall t' \in S : \forall I' \subsetneq \{1, \ldots, n\}, |I'| = |I|+1 : |\mathsf{coincident}(S, t', I')| < K^{n-|I|-1} \cdot (n-|I|-1)!$$

Now, we analyse some properties of $G = \langle V, E \rangle = \mathsf{graph}(S, t, I)$. First, note that $|V| = |\mathsf{coincident}(S, t, I)| \geq K^{n-|I|} \cdot (n-|I|)!$, since $\mathsf{coincident}(S, t, I)$ is the set of nodes of G. Second, we bound $\mathsf{maxdegree}(G)$ by bounding the degree of each node t' of G as follows:

$$\begin{aligned}
\mathsf{degree}(G, t') &\leq \textstyle\sum_{i \in \{1,\ldots,n\} \setminus I} \left(\left| \{ t'' \in \mathsf{coincident}(S, t, I) \mid t'[i] = t''[i] \} \right| - 1 \right) \\
&= \textstyle\sum_{i \in \{1,\ldots,n\} \setminus I} (|\mathsf{coincident}(S, t', I \uplus \{i\})| - 1) \\
&< (n-|I|) \cdot K^{n-|I|-1} \cdot (n-|I|-1)! - 1 \\
&= K^{n-|I|-1} \cdot (n-|I|)! - 1
\end{aligned}$$

Therefore, $\mathsf{maxdegree}(G) < K^{n-|I|-1} \cdot (n-|I|)! - 1$. By Lemma 4.30, there exists a subset $\tilde{S}$ of $\mathsf{coincident}(S, t, I)$ which is edge-free in G, includes t, and satisfies:

$$|\tilde{S}| = \left\lceil \frac{|V|}{\mathsf{maxdegree}(G)+1} \right\rceil \geq \left\lceil \frac{K^{n-|I|} \cdot (n-|I|)!}{K^{n-|I|-1} \cdot (n-|I|)!} \right\rceil = K$$

By Lemma 4.29, it follows that $\tilde{S}$ is an independent set. ∎

Example 4.32. *Consider again the definitions of Example 4.23, and let $t_4 = \langle 1, 2, 4 \rangle$. Clearly, $S \uplus \{t_4\}$ is not K-small: $\mathsf{coincident}(S \uplus \{t_4\}, t_4, I = \{1,2\}) = \{t_2, t_3, t_4\}$, which has cardinal 3, and thus, does not satisfy that it is strictly less than $K^{n-|I|} \cdot (n-|I|)! = 3^{3-2} \cdot (3-2)! = 3$. Since S was K-small, by Lemma 4.31 there exists an independent subset $\tilde{S}$ of $S \uplus \{t_4\}$ with size at least K and including t_4: $\{t_2, t_3, t_4\}$ is such a subset (in this example there is no other possible definition of $\tilde{S}$).*

In the following corollary we restate the result from [CJ03] as a particular consequence of Lemmas 4.24 and 4.31. This result is useful when it is not necessary to have a distinguished tuple t in the independent subset.

Corollary 4.33. *Let K be a natural number. Let $S \subseteq T$ be a set of tuples such that $|S| \geq K^n \cdot n!$.*

Then, there exists an independent subset of tuples $\tilde{S} \subseteq S$ satisfying $|\tilde{S}| \geq K$.

4.4.2 Independent sets of terms

In this section we translate the previous definitions and results from tuples to terms and positions.

Definition 4.34. *Let P be a set of positions. Let $p_1, \ldots, p_n$ be the positions in P, ordered lexicographically*[1]*. Let $t \in \mathcal{T}(\Sigma)$ be a term. We define $\mathsf{Tuple}_P(t)$ as the tuple $\langle s_1, \ldots, s_n \rangle$, where each s_i is $t|_{p_i}$ when p_i is in $\mathsf{Pos}(t)$, and a special symbol $\perp$ not in Σ, otherwise.*

*Let $S \subseteq \mathcal{T}(\Sigma)$ be a set of terms. We define $\mathsf{Tuples}_P(S)$ as $\{\mathsf{Tuple}_P(t) \mid t \in S\}$. We say that S is P-*independent *if $\mathsf{Tuples}_P(S)$ is independent. Let K be a natural number. We say that S is (K, P)-*small *if $\mathsf{Tuples}_P(S)$ is K-small.*

In order to adapt the previous results we need to guarantee that there is a bijection between the set of terms S and the set of tuples $\mathsf{Tuples}_P(S)$. In our concrete setting, this holds thanks to the fact that the considered set of positions P always contains λ. Hence, consider a set of positions P including λ and a set of terms $\{t_1, \ldots, t_m\} \subseteq \mathcal{T}(\Sigma)$ such that $P \cap \mathsf{Pos}(t_1) = \ldots = P \cap \mathsf{Pos}(t_m)$. Note that in this case, $\{t_1, \ldots, t_m\}$ is P-independent if and only if for each $p \in P$, either p is not in any of the sets $\mathsf{Pos}(t_1), \ldots, \mathsf{Pos}(t_m)$, or it is in all of them and either $t_1|_p = \ldots = t_m|_p$ or the subterms $t_1|_p, \ldots, t_m|_p$ are pairwise different.

The following facts are straightforwardly implied by Lemmas 4.24, 4.25, and 4.31, and Corollary 4.33, respectively.

Lemma 4.35. *Let P be a set of positions including λ and let K be a natural number. Let $S \subseteq \mathcal{T}(\Sigma)$ be a non-empty (K, P)-small set of terms.*

Then, $|S| < K^{|P|} \cdot |P|!$.

Lemma 4.36. *Let P be a set of positions including λ and let K be a natural number. Let $S \subseteq \mathcal{T}(\Sigma)$ be a set of terms.*

Then, checking whether S is (K, P)-small can be done with at most $|S|^2 \cdot 2^{|P|} \cdot |P|$ comparisons between elements of $\{t|_p \mid t \in S \ \wedge \ p \in \mathsf{Pos}(t) \cap P\} \uplus \{\perp\}$.

Lemma 4.37. *Let P be a set of positions including λ, let K be a natural number, and let $S \subseteq \mathcal{T}(\Sigma)$ be a set of terms which is not (K, P)-small. Let $t \in S$ be a term satisfying that $S \setminus \{t\}$ is (K, P)-small.*

Then, there exists a P-independent set $\tilde{S}$ of terms such that $t \in \tilde{S} \subseteq S$ and $|\tilde{S}| \geq K$.

Corollary 4.38. *Let P be a set of positions including λ, let K be a natural number, and let $S \subseteq \mathcal{T}(\Sigma)$ be a set of terms such that $|S| \geq K^{|P|} \cdot |P|!$.*

Then, there exists a P-independent set of terms $\tilde{S} \subseteq S$ satisfying $|\tilde{S}| \geq K$.

[1] The concrete selected order for positions is not important at all, but we choose this one in order to fix a precise definition.

4.5 Constraint-satisfying replacements

In this section we study how to perform replacements on runs of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ in a way that guarantees that all constraints are satisfied, i.e., that the weak run resulting from the replacement is actually a run. We start in Section 4.5.1 focusing on the implicit equality constraints. This is the simplest case since, in order to satisfy the equalities tested by the run, it can be proved that it suffices to perform the replacement simultaneously at several parallel positions. Moreover, these positions for the replacement can be easily defined by considering the positions of the run involved in equality tests. The remaining sections are devoted to disequality constraints. In Section 4.5.2 we formalize a criterion to distinguish two different kinds of disequality constraints. Intuitively, this distinction depends on how "close" are the positions tested by the disequality constraint to the positions where the replacement is performed. The "closest" ones are studied in Section 4.5.3 and the "furthest" ones in Section 4.5.4.

Recall that the reason to study replacements on runs of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ is to be able to reason about the emptiness of the recognized language: if any "big enough" run can be reduced by means of a decreasing replacement, then emptiness can be decided by checking only "small" runs. In order to formalize the notion of such decreasing replacements, we assume a given well-founded ordering $\ll$, total on terms, fulfilling the strict size relation (if $|t| < |t'|$, then $t \ll t'$) and monotonic (if $s \ll t|_p$, then $t[s]_p \ll t$). Note that a Knuth-Bendix ordering [KB70] with the standard term size comparison as first component satisfies these conditions. We consider this ordering extended to runs r, r' in a way such that $r \ll r'$ if $\mathsf{term}(r) \ll \mathsf{term}(r')$.

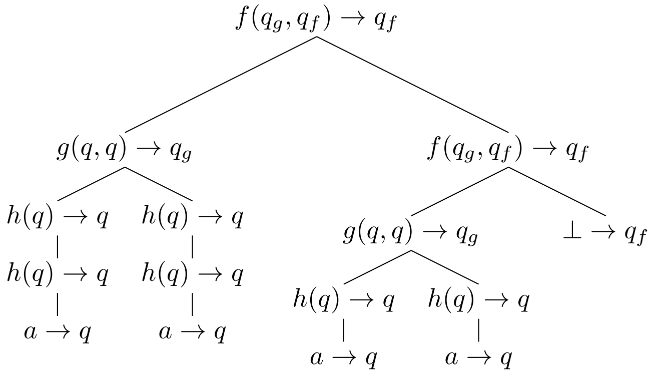
4.5.1 Equality constraints

We start by defining a kind of replacement that satisfies all the implicit equality constraints occurring in the rules applied in a run. Note that, in general, a simple replacement $r[r']_p$ is not enough, since equality tests checked at positions above p may become falsified after the replacement. To satisfy these equality tests it is necessary that such a replacement is done at the same time at all the subruns involved in an equality test, i.e., a replacement needs to be performed simultaneously at multiple parallel positions. In order to simplify the definition of these positions, we reason over uniform weak runs. Recall that uniform weak runs satisfy the implicit equality constraints occurring in the rules applied, and moreover, an implicit equality constraint asks for equality not only of subterms but also of weak subruns. Using these properties of uniform weak runs and the fact that equality constraints of $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ are implicitly defined by duplication of states, we can easily define the positions for the replacement with the following notion of abstract positions. Given a uniform weak run r and a position $p \in \mathsf{Pos}(r)$, we describe the abstract position of p in r as a sequence of the form $q_1.q_2 \ldots q_n.\bar{p}$, where $q_1, \ldots, q_n$ are states and $\bar{p}$ is a position. Intuitively, $q_1, \ldots, q_n$ are the states found while traversing r from the root to p, and $\bar{p}$ is the residual suffix of p after the last state.

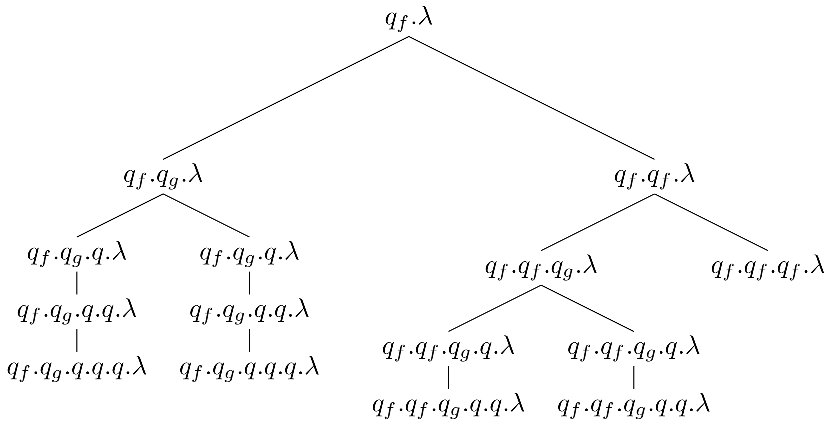
Definition 4.39. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let r be a uniform weak run of A, and let p be a position in $\mathsf{Pos}(r)$. We define the* abstract position *of p in r,* *denoted* $\mathsf{abstract}_r(p)$*, recursively as follows, where we explicitly write* $r(\lambda)$ *as a rule* $l \xrightarrow{c} q$*. For the case where* p *is in* $\mathsf{Pos}_\Sigma(l)$*,* $\mathsf{abstract}_r(p)$ *is defined as* $q.p$*. For the case where* p *is of the form* $p_1.p_2$*, where* p_1 *is a position in* $\mathsf{Pos}_Q(l)$*,* $\mathsf{abstract}_r(p)$ *is defined as* $q.\mathsf{abstract}_{r|_{p_1}}(p_2)$*. We write* $\mathsf{abstract}(p)$ *when* r *is clear from the context.*

We denote abstract positions as P*, with possible subscripts, and say that* P *is a* pure *abstract position when it is of the form* $q_1 \dots q_n.\lambda$*, for* $q_1, \dots, q_n \in Q$*.*

Example 4.40. *Consider the following uniform run* r *of some unspecified* $\mathtt{TA}_{\mathsf{ihom},\not\approx}$*:*



We depict the abstract position corresponding to each of the positions in $\mathsf{Pos}(r)$ *using the same tree structure of* $\mathsf{term}(r)$*:*



Note that all the abstract positions are pure: this is because all the rules applied in r *have a single alphabet symbol. Also note that the sibling positions* 1.1 *and* 1.2 *have the same abstract position* $q_f.q_g.q.\lambda$*: this is related to the fact that the rule* $g(q,q) \to q_g$ *applied at position* 1 *has two occurrences of the state* q *at its left-hand side, i.e., it has an implicit equality constraint between its two children. Moreover, since* r *is uniform, the respective children of* 1.1 *and* 1.2 *also share common abstract positions:* $\mathsf{abstract}(1.1.1) = \mathsf{abstract}(1.2.1) = q_f.q_g.q.q.\lambda$ *and* $\mathsf{abstract}(1.1.1.1) = \mathsf{abstract}(1.2.1.1) = q_f.q_g.q.q.q.\lambda$*. Since* $g(q,q) \to q_g$ *is also applied at* 2.1*, an analogous analysis can be made for the abstract positions corresponding to positions of the form* $2.1.1.p$ *and* $2.1.2.p$*.*

Note that, for a uniform weak run r and a position $p \in \mathsf{Pos}(r)$, $r(p)$ is defined if and only if $\mathsf{abstract}(p)$ is a pure abstract position. Furthermore, if two positions $p_1, p_2 \in \mathsf{Pos}(r)$ satisfy that $\mathsf{abstract}(p_1), \mathsf{abstract}(p_2)$ are pure and identical, then $r(p_1)$ and $r(p_2)$ are defined and $r|_{p_1}$ and $r|_{p_2}$ are equal. Intuitively, according to the definition of uniform weak run of a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$, for positions sharing the same sequence of states $q_1 \ldots q_n$ from the root, the corresponding uniform weak subruns must coincide. For this reason, with $r|_{q_1 \ldots q_n.\lambda}$ we denote such a uniform weak subrun, and with $r(q_1 \ldots q_n.\lambda)$ the rule applied at the root position of $r|_{q_1 \ldots q_n.\lambda}$. In addition, given a uniform weak run r' reaching the same state as $r|_{q_1 \ldots q_n.\lambda}$, we denote as $r[r']_{q_1 \ldots q_n.\lambda}$ the result of replacing by r' the uniform weak subrun at each position p holding $\mathsf{abstract}_r(p) = q_1 \ldots q_n.\lambda$. It is straightforward that such $r[r']_{q_1 \ldots q_n.\lambda}$ is also a uniform weak run. The following lemma formally states this property.

Lemma 4.41. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let r, r' be uniform weak runs of A. Let P be a pure abstract position of r such that $r|_P$ and r' reach the same state.*

Then, $r[r']_P$ is a uniform weak run.

The previous fact is equivalent to saying that a replacement $r[r']_P$ defined by means of a pure abstract position P necessarily satisfies the equality tests. However, note that nothing is guaranteed about the disequality constraints occurring in the rules applied in $r[r']_P$, even in the case where r and r' are uniform runs. Dealing with disequality constraints requires more complex arguments, and we present them in the following sections.

Before concluding this section, we give some additional definitions on abstract positions. This formalism, besides simplifying the previous Lemma 4.41, also helps in making the remaining reasonings of our work simpler and more accessible. For this reason, we are interested in adapting some typical operations on positions to the setting of abstract positions. In particular, we need to relax the conditions on abstract positions by allowing concatenations of the form $P.p$, where P is a pure abstract position and p is a position. We also need to compare abstract positions between them by means of a prefix relation. Such a relation is more complex for abstract positions than for positions, since an abstract position implicitly represents a set of positions.

Definition 4.42. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let r be a uniform weak run of A, and let $P, \bar{P}$ be abstract positions of r more explicitly written of the form $q_1 \ldots q_n.p$ and $\bar{q}_1 \ldots \bar{q}_m.\bar{p}$, respectively. We say that P is a* prefix *of $\bar{P}$, denoted $P \leq \bar{P}$, if $n \leq m$, $q_1 \ldots q_n = \bar{q}_1 \ldots \bar{q}_n$, and the following conditions hold:*

- *if $n = m$, then $p \leq \bar{p}$,*
- *if $n < m$, then the left-hand side l of the rule $r(q_1 \ldots q_n.\lambda)$ has an occurrence of state $\bar{q}_{n+1}$ in the subterm $l|_p$.*

Moreover, we say that P is $m - n$ steps above *$\bar{P}$. We say that P is a* strict prefix *of $\bar{P}$, denoted $P < \bar{P}$, if $P \leq \bar{P}$ and $P \neq \bar{P}$. We say that P and $\bar{P}$ are parallel, denoted $P \parallel \bar{P}$, if neither $P \leq \bar{P}$ nor $\bar{P} \leq P$ hold.*

Let P be a pure abstract position of r more explicitly written of the form $q_1 \ldots q_n.\lambda$. Let p be a position in $\mathsf{Pos}(r|_P)$. By the concatenation *$P.p$ we denote the abstract position $q_1 \ldots q_{n-1}.\mathsf{abstract}_{r|_P}(p)$.*

It is important to remark that the previous definitions on abstract positions contradict a usual intuition on positions. Consider two parallel abstract positions P_1 and P_2 of a uniform weak run r and note that, even though they are parallel, it is possible that both of them are prefix of a common abstract position P of r, i.e., that $P_1, P_2 \leq P$. It is easy to see that this is only possible in the case where P_1, P_2 and P can be written of the form $q_1 \dots q_n.p_1$, $q_1 \dots q_n.p_2$ and $q_1 \dots q_n.q_{n+1} \dots q_m.p$, respectively, with p_1 and p_2 being parallel positions and $n < m$, and moreover, the left-hand side of the rule $r(q_1 \dots q_n.\lambda)$ has an occurrence of the state q_{n+1} below the positions p_1 and p_2. The following technical lemma proves that, in the particular case where one of P_1, P_2 is a pure abstract position, then it is not possible that both of them are prefix of P.

Lemma 4.43. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let r be a uniform weak run of A. Let P_1, P_2 be parallel abstract positions of r such that at least one of them is pure.*

Then, there is no abstract position P of r such that $P_1 \leq P$ and $P_2 \leq P$.

Proof. Assume without loss of generality that P_1 is pure, and consider any abstract position P of r such that $P_1 \leq P$. In order to conclude, it suffices to prove $P_2 \not\leq P$. Let P_1 and P_2 be more explicitly written of the form $q_1 \dots q_n.\lambda$ and $q_1 \dots q_i.\bar{q}_{i+1} \dots \bar{q}_m.p_2$, respectively, where $q_1 \dots q_i$ is the maximal common prefix of P_1 and P_2. Note that since $P_1 \not\leq P_2$ and P_1 is pure, necessarily $i < n$. In the case where $i = m$, the left-hand side of the rule $r(q_1 \dots q_i.\lambda)$ does not have any occurrence of the state q_{i+1} below position p_2: otherwise $P_2 \leq P_1$ contradicting the assumption that P_1 and P_2 are parallel. Hence, in this case $P_2 \not\leq P$. In the case where $i < m$, we have that $q_{i+1} \neq \bar{q}_{i+1}$, and hence, $P_2 \not\leq P$ follows again. This concludes the proof. ■

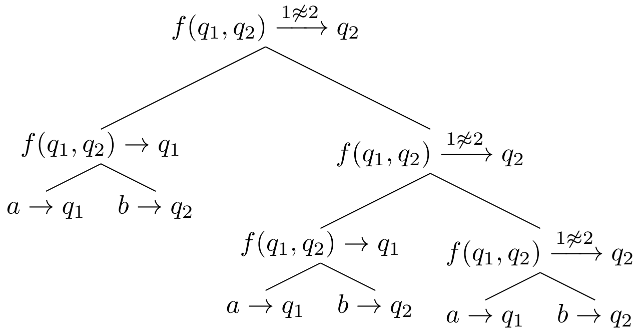
As a final remark, note that an abstract position is defined with respect to a concrete uniform weak run, which leads to some counterintuitive cases when comparing abstract positions of different uniform weak runs. For example, consider two uniform weak runs r_1 and r_2, and positions $p_1 \in \mathsf{Pos}(r_1)$ and $p_2 \in \mathsf{Pos}(r_2)$. Clearly, it is possible for p_1 and p_2 to be equal and yet $\mathsf{abstract}_{r_1}(p_1) \neq \mathsf{abstract}_{r_2}(p_2)$, and it is also possible that $P = \mathsf{abstract}_{r_1}(p_1) = \mathsf{abstract}_{r_2}(p_2)$ and yet $\mathsf{abstract}_{r_1}^{-1}(P) \neq \mathsf{abstract}_{r_2}^{-1}(P)$. For these reasons, comparing abstract positions of r_1 and r_2 can only be done when r_1 and r_2 are "similar". In our setting, we are interested in the case where r_1 and r_2 can be written of the form $r[r'_1]_P$ and $r[r'_2]_P$, respectively, for some uniform weak runs r, r'_1, r'_2 and pure abstract position P of r. Note that in such case, an abstract position P_1 of r_1 and an abstract position P_2 of r_2 can be compared if P is not a strict prefix of P_1 or P_2.

4.5.2 Classifying disequality constraints

We now consider the disequality constraints of the rules applied in $r[r']_P$, where r and r' are uniform runs and P is a pure abstract position of r such that $r|_P$ and r' reach the same state. Recall that $r[r']_P$ is necessarily a uniform weak run as stated in Lemma 4.41. Moreover, since r and r' are uniform runs, for $r[r']_P$ to satisfy all the constraints—and thus be a run—it only remains to prove that the disequality constraints of rules applied at prefixes of P are satisfied. That is, we have to show that $\mathsf{term}(r[r']_P)|_{\bar{p}.\bar{p}_1} \neq \mathsf{term}(r[r']_P)|_{\bar{p}.\bar{p}_2}$ holds for each triplet of positions $\langle \bar{p}, \bar{p}_1, \bar{p}_2 \rangle$ satisfying the following conditions: $r(\bar{p})$ is defined, the atom $\bar{p}_1 \not\approx \bar{p}_2$ occurs in the disequality constraint of the rule $r(\bar{p})$, $\mathsf{abstract}_r(\bar{p}) < P$, and $\bar{p}.\bar{p}_1, \bar{p}.\bar{p}_2 \in \mathsf{Pos}(r[r']_P)$. We can generalize this idea to abstract positions in order to simplify further reasonings. Consider any two such triplets $\langle \bar{p}, \bar{p}_1, \bar{p}_2 \rangle$ and $\langle \bar{p}', \bar{p}_1, \bar{p}_2 \rangle$ such that $\mathsf{abstract}_r(\bar{p}) = \mathsf{abstract}_r(\bar{p}')$. Note that, since r is a uniform run, it follows that $r|_{\bar{p}} = r|_{\bar{p}'}$. Therefore, a replacement at P satisfies $\mathsf{term}(r[r']_P)|_{\bar{p}.\bar{p}_1} \neq \mathsf{term}(r[r']_P)|_{\bar{p}.\bar{p}_2}$ if and only if it also satisfies $\mathsf{term}(r[r']_P)|_{\bar{p}'.\bar{p}_1} \neq \mathsf{term}(r[r']_P)|_{\bar{p}'.\bar{p}_2}$. In other words, different triplets with the same abstract positions are actually equivalent and we only need to reason about one of them. The following definition formalizes these triplets with abstract positions. Moreover, it also distinguishes the case where the positions are “close” to P, i.e., the test involves subterms of $\mathsf{term}(r')$, from the one where the positions are “far” from P.

Definition 4.44. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let r be a uniform weak run of A, and let P be a pure abstract position of r. Let $\bar{P}$ be a pure abstract position of r such that $\bar{P} < P$, and let $\bar{p}_1, \bar{p}_2$ be positions such that the atom $\bar{p}_1 \not\approx \bar{p}_2$ occurs in the disequality constraint of the rule $r(\bar{P})$. We say that a* disequality is tested *(by r) at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$. Moreover, we say that it is a* close disequality *(of r) with respect to P if $P \leq \bar{P}.\bar{p}_1$ or $P \leq \bar{P}.\bar{p}_2$, and otherwise, we say that it is a* far disequality *(of r) with respect to P. We say that it is* falsified *if $\bar{p}_1, \bar{p}_2 \in \mathsf{Pos}(r|_{\bar{P}})$ and $\mathsf{term}(r|_{\bar{P}})|_{\bar{p}_1} = \mathsf{term}(r|_{\bar{P}})|_{\bar{p}_2}$.*

Example 4.45. *Consider the following uniform weak run of an unspecified $\mathtt{TA}_{\mathsf{ihom},\not\approx}$:*


$f(q_1,q_2) \xrightarrow{1\not\approx 2} q_2$
$f(q_1,q_2) \rightarrow q_1$
$f(q_1,q_2) \xrightarrow{1\not\approx 2} q_2$
$a \rightarrow q_1$
$b \rightarrow q_2$
$f(q_1,q_2) \rightarrow q_1$
$f(q_1,q_2) \xrightarrow{1\not\approx 2} q_2$
$a \rightarrow q_1$
$b \rightarrow q_2$
$a \rightarrow q_1$
$b \rightarrow q_2$

Note that none of the applied rules involves an implicit equality constraint, and that the rules applied at positions $p_1 = \lambda$, $p_2 = 2$, $p_3 = 2.2$ have the same disequality constraint $1 \not\approx 2$. Let $P_1 = \mathsf{abstract}(p_1)$, $P_2 = \mathsf{abstract}(p_2)$, $P_3 = \mathsf{abstract}(p_3)$. With respect to P_3, note that $\langle P_1, 1, 2 \rangle$ is a far disequality since $P_1 < P_3$ but $P_3 \not\leq P_1.1$ and $P_3 \not\leq P_1.2$, whereas $\langle P_2, 1, 2 \rangle$ is a close disequality since $P_2 < P_3$ and $P_3 \leq P_2.2$ (actually, $P_3 = P_2.2$). Also, the disequality tested at $\langle P_3, 1, 2 \rangle$ classifies neither as close nor far with respect to P_3 since $P_3 \not< P_3$. Finally, note that the disequality tested at $\langle P_2, 1, 2 \rangle$ is falsified since the subterms being compared are both $f(a, b)$.

When r and P are clear from the context, we just say that a disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is close/far, and omit that the distinction is done with respect to P. In our setting, since we want to reason about the disequality tests falsified when performing a replacement, such an implicit P corresponds to where the replacement takes place.

In some cases, when $\bar{p}_1, \bar{p}_2$ are clear from the context or not relevant, we just say that a disequality is tested at $\bar{P}$. Finally, we say that a disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is tested d steps above P if $\bar{P}$ is d steps above P. We deal with close and far disequalities separately in the following sections.

4.5.3 Close disequalities

We first tackle the falsified close disequalities in the replacement $r[r']_P$. Recall that, in this case, such disequalities are necessarily tested at triplets $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ where $\bar{P} < P$ and $P \leq \bar{P}.\bar{p}_1 \vee P \leq \bar{P}.\bar{p}_2$. The following example illustrates some of the challenges of dealing with close disequalities.

Example 4.46. *Let $\Sigma = \{\bot{:}0, a{:}0, h{:}1, g{:}2, f{:}2\}$, and consider the language L of terms of the form $f(e_1, f(e_2, \ldots f(e_m, \bot) \ldots))$, where $m \geq 0$ and each e_i is a term of the form $g(h^{\alpha_i}(a), h^{\beta_i}(a))$ with distinct $\alpha_i, \beta_i \geq 0$, and if $i < m$, then $e_i|_1 \neq e_{i+1}|_1$ and $e_i|_2 \neq e_{i+1}|_2$ (i.e., $\alpha_i \neq \alpha_{i+1}$ and $\beta_i \neq \beta_{i+1}$). Such language can be recognized by $\mathsf{TA}_{\mathsf{ihom},\not\approx}$ using only disequality constraints. In particular, to ensure that α_i, β_i are distinct for each subterm e_i, we could use a disequality constraint of the form $1 \not\approx 2$ at the root position of each e_i. Nevertheless, to better illustrate different cases of close disequalities, we perform such test from the direct parent of e_i instead. Let A be the $\mathsf{TA}_{\mathsf{ihom},\not\approx}$ $\langle \{q, q', q_g, q_f\}, \Sigma, \{q_f\}, \Delta \rangle$, where Δ is the set with the unconstrained rules $a \to q$, $a \to q'$, $h(q) \to q$, $h(q) \to q'$, $g(q, q') \to q_g$, $\bot \to q_f$, and the constrained rule:*

$$f(q_g, q_f) \xrightarrow{\substack{1.1 \not\approx 1.2 \,\wedge \\ 1.1 \not\approx 2.1.1 \,\wedge \\ 1.2 \not\approx 2.1.2}} q_f$$

Note that the first disequality atom corresponds to the test $\alpha_i \neq \beta_i$, whereas the next two atoms to $\alpha_i \neq \alpha_{i+1}$ and $\beta_i \neq \beta_{i+1}$, respectively. It is easy to see that $\mathcal{L}(A) = L$. As an additional remark, note that since all the rules have a single alphabet symbol and have no implicit equality constraints, there is a bijection between positions and abstract positions for all runs of A. Thus, in this example both notions are interchangeable.

Assume a given accepting uniform run r of A where we want to replace one of its subruns $r|_P$ reaching q_g (i.e., recognizing one of the subterms e_i) by a new run also reaching q_g chosen among some candidate runs $r_1, \ldots, r_n$ of A, such that the close disequalities are satisfied after the replacement. Assume also that P is neither the shortest nor the longest abstract position of r of the form $q_f \ldots q_f.q_g.\lambda$ (i.e., that $0 < i < m$). Under these conditions, each of the replacements $r[r_j]_P$ might falsify at most the following 5 distinct close disequalities with respect to P:

- *Let $\bar{P}_1$ be the abstract position one step above P (i.e., the direct parent). The disequalities $\langle \bar{P}_1, 1.1, 1.2 \rangle$, $\langle \bar{P}_1, 1.1, 2.1.1 \rangle$, $\langle \bar{P}_1, 1.2, 2.1.2 \rangle$ tested by the rule $r(\bar{P}_1)$ are close with respect to P since $\bar{P}_1 < P \leq \bar{P}_1.1.1$, $\bar{P}_1.1.2$ (actually, $P = \bar{P}_1.1$).*

- *Let $\bar{P}_2$ be the abstract position two steps above P (i.e., the grandparent). The disequalities $\langle \bar{P}_2, 1.1, 2.1.1 \rangle$, $\langle \bar{P}_2, 1.2, 2.1.2 \rangle$ tested by the rule $r(\bar{P}_2)$ are close with respect to P since $\bar{P}_2 < P \leq \bar{P}_2.2.1.1$, $\bar{P}_2.2.1.2$ (actually, $P = \bar{P}_2.2.1$). Note that $r(\bar{P}_2)$ also tests $\langle \bar{P}_2, 1.1, 1.2 \rangle$, but it is a far disequality with respect to P since $\bar{P}_2 < P$ and $P \not\leq \bar{P}_2.1.1$, $P \not\leq \bar{P}_2.1.2$.*

Hence, each of the close disequalities to consider involves the subterm $\mathsf{term}(r_j|_1)$ *or* $\mathsf{term}(r_j|_2)$*: in the case of* $\langle \bar{P}_1, 1.1, 1.2\rangle$*, both subterms are compared against each other, whereas in the remaining close disequalities the tests compare one of those subterms against another subterm pending at a position parallel to* P*. These two situations require different techniques to find an* r_j *satisfying all close disequalities:*

- *To satisfy* $\langle \bar{P}_1, 1.1, 1.2\rangle$*, we need to guarantee* $\mathsf{term}(r_j|_1) \neq \mathsf{term}(r_j|_2)$*. To this end, in general it is convenient that each of the runs in the list of candidates* $r_1, \ldots, r_n$ *satisfies that its subruns pending at* 1 *and* 2 *recognize distinct terms if, and only if, the subterms* $\mathsf{term}(r|_P)|_1$ *and* $\mathsf{term}(r|_P)|_2$ *that they replace are distinct. Such condition is captured by the notion of* $\sim^A$*-equivalence between terms that will be introduced in Definition 4.49. We will prove in Lemma 4.50 that, when* $r_1, \ldots, r_n$ *recognize terms* $\sim^A$*-equivalent to the subterm* $\mathsf{term}(r|_P)$ *being replaced, then all the replacements* $r[r_j]_P$ *satisfy the close disequality* $\langle \bar{P}_1, 1.1, 1.2\rangle$*.*

- *To satisfy the remaining close disequalities, we need to guarantee that* $\mathsf{term}(r_j|_1)$ *is different from* $\mathsf{term}(r|_{\bar{P}_1.2.1.1})$ *and* $\mathsf{term}(r|_{\bar{P}_2.1.1})$*, and that* $\mathsf{term}(r_j|_2)$ *is different from* $\mathsf{term}(r|_{\bar{P}_1.2.1.2})$ *and* $\mathsf{term}(r|_{\bar{P}_2.1.2})$*. Since these cases are analogous, we just focus on guaranteeing* $\mathsf{term}(r_j|_1) \neq \mathsf{term}(r|_{\bar{P}_1.2.1.1})$*. To this end, we use the notion of independence from Section 4.4.2: if* $\mathsf{term}(r|_P), \mathsf{term}(r_1), \ldots, \mathsf{term}(r_n)$ *form a* $\{1,2\}$*-independent set, then either* $\mathsf{term}(r|_P)|_k, \mathsf{term}(r_1)|_k, \ldots, \mathsf{term}(r_n)|_k$ *are equal or they are pairwise different, for* $k \in \{1,2\}$*. We will prove in Lemma 4.50 that, when* $\{1,2\}$*-independence holds and the candidates* $r_1, \ldots, r_n$ *recognize distinct terms, then* $\mathsf{term}(r_j|_1) \neq \mathsf{term}(r|_{\bar{P}_1.2.1.1})$ *is falsified by at most one of the candidates* r_j *among* $r_1, \ldots, r_n$*. Thus, since we have* 4 *such close disequalities to consider, we need* $n > 4$ *candidates to ensure the existence of an* r_j *satisfying all close disequalities.*

From the previous example, it is easy to see that a close disequality $\langle \bar{P}, \bar{p}_1, \bar{p}_2\rangle$ in a replacement $r[r']_P$ necessarily involves a subterm of $\mathsf{term}(r')$ pending at some position p', where p' is suffix of $\bar{p}_1$ or $\bar{p}_2$. We define the set of such suffixes as follows.

Definition 4.47. *Let* A *be a* $\mathsf{TA}_{\mathsf{ihom},\not\approx}$*. We define the set of positions* $\mathsf{suff}_{\not\approx}(A)$ *as the set of suffixes of the positions occurring in the disequality constraints of the rules of* A*, i.e.,* $\{p \mid \exists (l \xrightarrow{c} q) \in \Delta, \exists p_1, p_2 : (p_1.p \not\approx p_2) \in c\}$ *where* Δ *is the set of rules of* A *(recall that disequality atoms are unordered pairs). We just write* $\mathsf{suff}_{\not\approx}$ *when* A *is clear from the context.*

Example 4.48. *Consider the* $\mathsf{TA}_{\mathsf{ihom},\not\approx}$ A *from Example 4.46, and note that* $\mathsf{suff}_{\not\approx}$ *is* $\{\lambda,\ 1,\ 1.1,\ 2.1.1,\ 2,\ 1.2,\ 2.1.2\}$*. This is a strict superset of the positions of the candidate runs* $r_1, \ldots, r_n$ *identified in Example 4.46 as being involved in some close disequality (with respect to* P *in the replacements* $r[r_j]_P$*), i.e., positions* 1 *and* 2*.*

In order to define replacements that do not falsify any close disequality, we first introduce an equivalence relation $\sim^A$ on terms, induced by a $\mathsf{TA}_{\mathsf{ihom},\not\approx}$ A. Intuitively, two terms are equivalent if they share the same set of positions among the positions in $\mathsf{suff}_{\not\approx}$, and moreover, they satisfy the same equality and disequality relations among subterms at such positions.

Definition 4.49. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. We define the equivalence relation $\sim^A$ on $\mathcal{T}(\Sigma)$ as $t \sim^A t'$ if and only if the following conditions hold:*

- $\mathsf{Pos}(t) \cap \mathsf{suff}_{\not\approx} = \mathsf{Pos}(t') \cap \mathsf{suff}_{\not\approx}$,
- $\forall p_1, p_2 \in \mathsf{Pos}(t) \cap \mathsf{suff}_{\not\approx} : (t|_{p_1} = t|_{p_2} \Leftrightarrow t'|_{p_1} = t'|_{p_2})$.

Now, consider a specific disequality tested by the uniform run r at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$, and assume that it is a close disequality with respect to a pure abstract position P of r. Given some candidate uniform runs $r_1, \ldots, r_n$ for replacements of the form $r[r_i]_P$, we prove that at most one of those replacements can falsify the close disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ if $r|_P, r_1, \ldots, r_n$ recognize distinct terms that are $\sim^A$-equivalent and form a $\mathsf{suff}_{\not\approx}$-independent set.

Lemma 4.50. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let r be a uniform run of A, and let P be a pure abstract position of r. Let a close disequality with respect to P be tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$. Let $r_1, \ldots, r_n$ be uniform runs of A reaching the same state as $r|_P$ and such that the terms $\mathsf{term}(r|_P), \mathsf{term}(r_1), \ldots, \mathsf{term}(r_n)$ are pairwise different, $\sim^A$-equivalent, and form a $\mathsf{suff}_{\not\approx}$-independent set.*

Then, for at most one $i \in \{1, \ldots, n\}$, the replacement $r[r_i]_P$ falsifies the close disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$.

Proof. We start considering the case $\{\bar{p}_1, \bar{p}_2\} \not\subseteq \mathsf{Pos}(r|_{\bar{P}})$. This is straightforward, since the assumptions that $\mathsf{term}(r|_P), \mathsf{term}(r_1), \ldots, \mathsf{term}(r_n)$ are $\sim^A$-equivalent and $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is close with respect to P guarantee that $\{\bar{p}_1, \bar{p}_2\} \not\subseteq \mathsf{Pos}(r[r_i]_P|_{\bar{P}})$ for each $i \in \{1, \ldots, n\}$, and thus, in all the replacements the close disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is trivially satisfied. Hence, from now on we assume $\{\bar{p}_1, \bar{p}_2\} \subseteq \mathsf{Pos}(r|_{\bar{P}})$, and note that $\{\bar{p}_1, \bar{p}_2\} \subseteq \mathsf{Pos}(r[r_i]_P|_{\bar{P}})$ follows for each $i \in \{1, \ldots, n\}$.

We reason on the underlying terms. Let $t = \mathsf{term}(r|_{\bar{P}})$, and let $s_i = \mathsf{term}(r_i)$ for each $i \in \{1, \ldots, n\}$. Note that $t|_{\bar{p}_1} \neq t|_{\bar{p}_2}$ holds. Let S be the set of positions of t where the replacements take place, i.e., $S = \{p \in \mathsf{Pos}(t) \mid \exists \bar{p} \in \mathsf{abstract}_r^{-1}(\bar{P}) : \mathsf{abstract}_r(\bar{p}.p) = P\}$ (note that the set could be equivalently defined changing the existential quantifier in the condition by a universal quantifier). By definition, S is a non-empty set of parallel positions, and moreover, the subterms of t pending at the positions in S are all identical. To ease the presentation, we denote by $t|_S$ the subterm of t pending at any of the positions in S, and by $t[s_i]_S$ the simultaneous replacement in t of all the subterms pending at positions in S by s_i. Note that, by the assumptions of the lemma, $t|_S, s_1, \ldots, s_n$ are distinct terms, $\sim^A$-equivalent, and form a $\mathsf{suff}_{\not\approx}$-independent set. In order to conclude, it suffices to show that at most one $i \in \{1, \ldots, n\}$ satisfies $t[s_i]_S|_{\bar{p}_1} = t[s_i]_S|_{\bar{p}_2}$.

By the assumption that $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is close with respect to P, it follows that there exists $p \in S$ such that $p \leq \bar{p}_1$ or $p \leq \bar{p}_2$. Since both cases are symmetric, without loss of generality we assume $p \leq \bar{p}_1$. Let $\bar{p}'_1$ be $\bar{p}_1 - p$ and note that $\bar{p}'_1 \in \mathsf{suff}_{\not\approx}$. Now, we distinguish cases depending on $\bar{p}_2$ and the positions in S. First, assume that there exists $p' \in S$ such that $p' \leq \bar{p}_2$. Let $\bar{p}'_2$ be $\bar{p}_2 - p'$ and note that $\bar{p}'_2 \in \mathsf{suff}_{\not\approx}$. In this case, it suffices to prove that at most one $i \in \{1, \ldots, n\}$ satisfies $s_i|_{\bar{p}'_1} = s_i|_{\bar{p}'_2}$. But, since $t|_S \sim^A s_i$ and $t|_S|_{\bar{p}'_1} = t|_p|_{\bar{p}'_1} = t|_{\bar{p}_1} \neq t|_{\bar{p}_2} = t|_{p'}|_{\bar{p}'_2} = t|_S|_{\bar{p}'_2}$, it follows $s_i|_{\bar{p}'_1} \neq s_i|_{\bar{p}'_2}$, for all $i \in \{1, \ldots, n\}$. Second, assume that $\bar{p}_2$ is parallel to all positions in S, and consider any $i, j \in \{1, \dots, n\}$ such that $t[s_i]_S|_{\bar{p}_1} = t[s_i]_S|_{\bar{p}_2}$ and $t[s_j]_S|_{\bar{p}_1} = t[s_j]_S|_{\bar{p}_2}$. Note that, in this case, this condition is equivalent to saying $s_i|_{\bar{p}'_1} = t|_{\bar{p}_2}$ and $s_j|_{\bar{p}'_1} = t|_{\bar{p}_2}$. It suffices to note that necessarily $i = j$: otherwise, $t|_S|_{\bar{p}'_1} = s_i|_{\bar{p}'_1} = s_j|_{\bar{p}'_1}$ since $\{t|_S, s_1, \dots, s_n\}$ is $\mathsf{suff}_{\not\approx}$-independent, and thus, $t|_{\bar{p}_1} = t|_{\bar{p}_2}$ since $t|_S|_{\bar{p}'_1} = t|_p|_{\bar{p}'_1} = t|_{\bar{p}_1}$, contradicting $t|_{\bar{p}_1} \neq t|_{\bar{p}_2}$. Third, assume that there exists $p' \in S$ satisfying $\bar{p}_2 < p'$. Then, for all $i \in \{1, \dots, n\}$ it follows $\mathsf{height}(t[s_i]_S|_{\bar{p}_1}) < \mathsf{height}(t[s_i]_S|_{\bar{p}_2})$, and thus, $t[s_i]_S|_{\bar{p}_1} \neq t[s_i]_S|_{\bar{p}_2}$. ∎

Now we are ready to construct a replacement $r[r']_P$ that does not falsify any close disequality. From the previous result, it is clear that a single candidate r' for the replacement at P might not suffice, and instead, we require some uniform runs $r_1, \dots, r_n$ such that n is greater than the number of different close disequalities in r. Moreover, these $r_1, \dots, r_n$ must reach the same state as $r|_P$ and satisfy that $\mathsf{term}(r|_P), \mathsf{term}(r_1), \dots, \mathsf{term}(r_n)$ are pairwise different, $\sim^A$-equivalent, and form a $\mathsf{suff}_{\not\approx}$-independent set. These conditions are enough to guarantee that if $r[r_i]_P$ falsifies a close disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$, then no other $r[r_j]_P$, with $i \neq j$, can falsify the close disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$. The following lemma states the number n of needed candidates to construct m replacements that do not falsify any close disequality.

Lemma 4.51. *Let A be a $\mathsf{TA}_{\mathsf{ihom},\not\approx}$. Let m be a natural number, and let $n = \mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx} + m$. Let r be a uniform run of A, and let P be a pure abstract position of r. Let $r_1, \dots, r_n$ be uniform runs of A reaching the same state as $r|_P$ and such that the terms $\mathsf{term}(r|_P), \mathsf{term}(r_1), \dots, \mathsf{term}(r_n)$ are pairwise different, $\sim^A$-equivalent, and form a $\mathsf{suff}_{\not\approx}$-independent set.*

Then, there exists a subset $\{i_1, \dots, i_m\}$ of $\{1, \dots, n\}$ such that the replacements $r[r_{i_1}]_P, \dots, r[r_{i_m}]_P$ do not falsify any close disequality.

Proof. Note that by Lemma 4.50, each close disequality can be falsified in at most one of the replacements $r[r_1]_P, \dots, r[r_n]_P$. Also note that, since a close disequality can be tested at most $\mathsf{h}_{\not\approx}$ steps above P and there are $\mathsf{n}_{\not\approx}$ different disequality atoms in the rules of A, it follows that there are at most $\mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx}$ different close disequalities that we need to consider. Therefore, $n - \mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx} = m$ of the replacements $r[r_1]_P, \dots, r[r_n]_P$, say, $r[r_{i_1}]_P, \dots, r[r_{i_m}]_P$, do not falsify any close disequality. ∎

The previous result is not enough for our purposes, since the arguments in Section 4.5.4 need a bound for the case where the candidate terms are not assumed to be $\sim^A$-equivalent or forming a $\mathsf{suff}_{\not\approx}$-independent set. These assumptions are necessary when the replacement must be performed at a fixed pure abstract position P of the uniform run r, but not when such P can be chosen among several possibilities. Hence, consider some pure abstract positions $P_1, \dots, P_n$ of r such that $r|_{P_1}, \dots, r|_{P_n}$ reach the same state and recognize distinct terms. We prove that, when n is "big enough", there exists a subset $\{i_1, \dots, i_m\}$ of $\{1, \dots, n\}$ such that the replacements $r[r|_{P_{i_1}}]_{P_{i_m}}, \dots, r[r|_{P_{i_{m-1}}}]_{P_{i_m}}$ do not falsify any close disequality with respect to P_{i_m}. The value of such n is given by means of the function $\mathcal{B}_{\text{close}}$ of Definition 4.54, which uses as intermediate result the following bound for the number of equivalence classes induced by the relation $\sim^A$.

Definition 4.52. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. We define $\mathcal{B}_{\text{eq}}(A)$ as $2^{|\mathsf{suff}_{\not\approx}|} \cdot |\mathsf{suff}_{\not\approx}|^{|\mathsf{suff}_{\not\approx}|}$.*

Lemma 4.53. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$.*

Then, the number of different equivalence classes induced by $\sim^A$ is bounded by $\mathcal{B}_{\text{eq}}(A)$.

Proof. The first condition of the definition of $\sim^A$ induces as many equivalence classes as subsets of $\mathsf{suff}_{\not\approx}$ are, and this is bounded by $2^{|\mathsf{suff}_{\not\approx}|}$. The second condition of the definition of $\sim^A$ depends on which subterms pending at positions in $\mathsf{suff}_{\not\approx}$ are equal or different. This condition induces as many equivalence classes as the number of partitions of the set $\mathsf{suff}_{\not\approx}$, and this is bounded by $|\mathsf{suff}_{\not\approx}|^{|\mathsf{suff}_{\not\approx}|}$. The statement follows by combining both bounds. ■

We now give the concrete definition of $\mathcal{B}_{\text{close}}$ for the number of needed candidates.

Definition 4.54. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let m be a natural number. We define $\mathcal{B}_{\text{close}}(A, m)$ as $(\mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx} + m)^{|\mathsf{suff}_{\not\approx}|} \cdot |\mathsf{suff}_{\not\approx}|! \cdot \mathcal{B}_{\text{eq}}(A)$.*

Lemma 4.55. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let m be a natural number, and let $n = \mathcal{B}_{\text{close}}(A, m)$. Let r be a uniform run of A. Let $P_1, \ldots, P_n$ be pure abstract positions of r such that the subruns $r|_{P_1}, \ldots, r|_{P_n}$ reach the same state and the terms $\mathsf{term}(r|_{P_1}), \ldots, \mathsf{term}(r|_{P_n})$ are pairwise different.*

Then, there exists a subset $\{i_1, \ldots, i_m\}$ of $\{1, \ldots, n\}$ such that $r|_{P_{i_1}} \ll \ldots \ll r|_{P_{i_m}}$ and the replacements $r[r|_{P_{i_1}}]_{P_{i_m}}, \ldots, r[r|_{P_{i_{m-1}}}]_{P_{i_m}}$ do not falsify any close disequality.

Proof. By Lemma 4.53, there are $n' := n/\mathcal{B}_{\text{eq}}(A)$ pure abstract positions among $P_1, \ldots, P_n$ satisfying that the terms recognized by the subruns at such positions are $\sim^A$-equivalent. Without loss of generality, we assume that these n' pure abstract positions are the first ones, i.e., that the terms $\mathsf{term}(r|_{P_1}), \ldots, \mathsf{term}(r|_{P_{n'}})$ are $\sim^A$-equivalent. Since n' is $(\mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx} + m)^{|\mathsf{suff}_{\not\approx}|} \cdot |\mathsf{suff}_{\not\approx}|!$, by Corollary 4.38 there exists a $\mathsf{suff}_{\not\approx}$-independent subset of $\{\mathsf{term}(r|_{P_1}), \ldots, \mathsf{term}(r|_{P_{n'}})\}$ with size $n'' := \mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx} + m$. Without loss of generality, we assume that this subset is formed by the terms recognized by the subruns at the n'' first pure abstract positions, i.e., that $\{\mathsf{term}(r|_{P_1}), \ldots, \mathsf{term}(r|_{P_{n''}})\}$ is $\mathsf{suff}_{\not\approx}$-independent. We also assume without loss of generality that $\mathsf{term}(r|_{P_1}) \ll \ldots \ll \mathsf{term}(r|_{P_{n''}})$. Let the i_m of the statement be defined as n''. By Lemma 4.51 applied on r, P_{i_m} and $r|_{P_1}, \ldots, r|_{P_{n''-1}}$, it follows that there exists a subset $\{i_1, \ldots, i_{m-1}\}$ of $\{1, \ldots, n'' - 1\}$ such that the replacements $r[r|_{P_{i_1}}]_{P_{i_m}}, \ldots, r[r|_{P_{i_{m-1}}}]_{P_{i_m}}$ do not falsify any close disequality. We conclude the proof by assuming without loss of generality that $i_1 < \ldots < i_{m-1}$, and thus, $r|_{P_{i_1}} \ll \ldots \ll r|_{P_{i_{m-1}}} \ll r|_{P_{i_m}}$. ■

4.5.4 Far disequalities

We start with a result on far disequalities analogous to the statement on close disequalities of Lemma 4.50. More precisely, consider a specific disequality tested by the uniform run r at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$, and assume that it is a far disequality with respect to a pure abstract position P of r. Given some candidate uniform runs $r_1, \ldots, r_n$ for replacements of the form $r[r_i]_P$, we prove that at most one of those replacements can falsify the far disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$. In contrast with Lemma 4.50, in this case the only needed assumptions on $r_1, \ldots, r_n$ are that they reach the same state as $r|_P$ and recognize pairwise different terms.

Lemma 4.56. *Let A be a $\mathsf{TA}_{\mathsf{ihom},\not\approx}$. Let r be a uniform run of A, and let P be a pure abstract position of r. Let a far disequality with respect to P be tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$. Let $r_1, \ldots, r_n$ be uniform runs of A on distinct terms and reaching the same state as $r|_P$.*

Then, for at most one $i \in \{1, \ldots, n\}$, the replacement $r[r_i]_P$ falsifies the far disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$.

Proof. We start considering the case $\{\bar{p}_1, \bar{p}_2\} \not\subseteq \mathsf{Pos}(r|_{\bar{P}})$. This is straightforward, since the assumption that $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is far with respect to P guarantees that $\{\bar{p}_1, \bar{p}_2\} \not\subseteq \mathsf{Pos}(r[r_i]_P|_{\bar{P}})$ for each $i \in \{1, \ldots, n\}$, and thus, in all the replacements the far disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is trivially satisfied. Hence, from now on we assume $\{\bar{p}_1, \bar{p}_2\} \subseteq \mathsf{Pos}(r|_{\bar{P}})$, and note that $\{\bar{p}_1, \bar{p}_2\} \subseteq \mathsf{Pos}(r[r_i]_P|_{\bar{P}})$ follows for each $i \in \{1, \ldots, n\}$.

We reason on the underlying terms. Let $t_i = \mathsf{term}(r|_{\bar{P}.\bar{p}_i})$ for $i \in \{1, 2\}$, and note that $t_1 \neq t_2$ holds. For $i \in \{1, 2\}$, let S_i be the sets of positions of t_i where the replacements take place, i.e., $S_i = \{p \in \mathsf{Pos}(t_i) \mid \exists \bar{p} \in \mathsf{abstract}_r^{-1}(\bar{P}) : \mathsf{abstract}_r(\bar{p}.\bar{p}_i.p) = P\}$. By definition, S_i is a (maybe empty) set of parallel positions, and moreover, the subterms of t_i pending at the positions in S_i are all identical. As in the proof of Lemma 4.50, we denote by $t_i[s]_{S_i}$ the simultaneous replacement in t_i of all the subterms pending at positions in S_i by a term s. In order to conclude, it suffices to show that at most one term s satisfies $t_1[s]_{S_1} = t_2[s]_{S_2}$.

We assume that there exists a term s satisfying $t_1[s]_{S_1} = t_2[s]_{S_2}$, and prove that it is unique. Note that, for each position $p_1 \in S_1$, there is no position $p_2 \in S_2$ such that $p_1 < p_2$ or $p_2 < p_1$: otherwise, the condition $t_1[s]_{S_1} = t_2[s]_{S_2}$ would be false for any s. Also, note that $S_1 \neq S_2$: otherwise, the condition $t_1[s]_{S_1} = t_2[s]_{S_2}$ would imply $t_1 = t_2$ since all the replaced subterms of t_1 and t_2 are identical by definition. Hence, there exists a position $p \in (S_1 - S_2) \cup (S_2 - S_1)$. Without loss of generality, assume that such a p is in $S_1 - S_2$. The condition $t_1[s]_{S_1} = t_2[s]_{S_2}$ and the fact that any position in S_2 is parallel with p implies $s = t_2|_p$, and we are done. ■

It is clear from the previous result that a single candidate r' for the replacement $r[r']_P$ is not enough to guarantee that no far disequality is falsified. In particular, given some candidates $r_1, \ldots, r_n$ recognizing distinct terms, each specific far disequality can only be falsified by one of the r_i's when performing the replacement at P. In contrast to our arguments in Section 4.5.3 dealing with close disequalities, note that the number of far disequalities is not bounded. For this reason, the definition of the number n of needed candidates is more complex. We start with the following intermediate lemma where we only consider far disequalities that are "near" P, i.e., far disequalities tested at a bounded distance from the pure abstract position where the replacement is performed.

Lemma 4.57. *Let A be a $\mathsf{TA}_{\mathsf{ihom},\not\approx}$. Let m, k be natural numbers, and let $n = k \cdot \mathsf{n}_{\not\approx} + m$. Let r be a uniform run of A, and let P be a pure abstract position of r. Let $r_1, \ldots, r_n$ be uniform runs of A on distinct terms reaching the same state as $r|_P$, and such that,* *for a given natural number d, the replacements $r[r_1]_P, \ldots, r[r_n]_P$ do not falsify any far disequality tested at most d steps above P.*

Then, there exists a subset $\{i_1, \ldots, i_m\}$ of $\{1, \ldots, n\}$ such that the replacements $r[r_{i_1}]_P, \ldots, r[r_{i_m}]_P$ do not falsify any far disequality tested at most $d + k$ steps above P.

Proof. Note that by Lemma 4.56, each far disequality can be falsified in at most one of the replacements $r[r_1]_P, \ldots, r[r_n]_P$. Also note that, among the far disequalities that are tested at most $d+k$ steps above P, we do not need to consider the ones that are tested at most d steps above P since they are already satisfied by assumption. Hence, since there are $\mathsf{n}_{\not\approx}$ different disequality atoms in the rules of A, it follows that there are at most $k \cdot \mathsf{n}_{\not\approx}$ different far disequalities that we need to consider. Therefore, $n - k \cdot \mathsf{n}_{\not\approx} = m$ of the replacements $r[r_1]_P, \ldots, r[r_n]_P$, say, $r[r_{i_1}]_P, \ldots, r[r_{i_m}]_P$, do not falsify any far disequality tested at most $d + k$ steps above P. ■

Now we are ready to tackle the far disequalities that are not "near" P. Consider that we have candidates $r_1, \ldots, r_n$ such that the replacements $r[r_1]_P, \ldots, r[r_n]_P$ do not falsify any close disequality. We assume that all of them falsify some far disequality, since otherwise, no further arguments would be needed. We define an n "big enough" to guarantee that we are able to construct from subruns of r and from $r_1, \ldots, r_n$ new candidates $r'_1, \ldots, r'_n$ for replacements at a pure abstract position $P' < P$ such that, again, $r[r'_1]_{P'}, \ldots, r[r'_n]_{P'}$ do not falsify any close disequality. Note that in the case where all of them falsify some far disequality, this argument can be iterated to obtain new candidates $r''_1, \ldots, r''_n$ to perform replacements at a $P'' < P' < P$, i.e., at a pure abstract position closer to the root. Hence, we are guaranteed to eventually find a replacement that does not falsify any far disequality. The number n of needed candidates is given by means of the function $\mathcal{B}$ of Definition 4.59 and the proof of this fact is given in Lemma 4.60. The function $\mathcal{B}$ takes two natural numbers M and N for which we do not give a concrete definition until Lemma 4.62. At this point it suffices to assume that they satisfy $M \cdot N \geq \mathcal{B}(A, M, N) = n$. In order to illustrate the definition of $\mathcal{B}$, we sketch the steps that we perform in the proof of Lemma 4.60 to construct the new candidates $r'_1, \ldots, r'_n$ and to find the new pure abstract position P' for the replacement. We start by noting that, since all the replacements $r[r_1]_P, \ldots, r[r_n]_P$ falsify far disequalities, we can consider the maximal pure abstract positions $\bar{P}_1, \ldots, \bar{P}_n$ such that, for $i \in \{1, \ldots, n\}$, the replacement $r[r_i]_P$ falsifies a far disequality tested at $\bar{P}_i$. We also assume without loss of generality that $\bar{P}_1 \geq \bar{P}_2 \geq \ldots \geq \bar{P}_n$ by reordering the runs r_i if necessary (see Figure 4.58). Now, the proof proceeds as follows:

- As an initial step, we need the pure abstract positions $\bar{P}_1, \ldots, \bar{P}_n$ to be spaced between them. In particular, we want any two $\bar{P}_i, \bar{P}_j$ to be more than $\mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}$ steps away from each other. To this end, we first remove from $\bar{P}_1, \ldots, \bar{P}_n$ any repetition of pure abstract positions. Recall that, by Lemma 4.56, the far disequalities falsified by a candidate r_i are necessarily different from the far disequalities falsified by any other candidate r_j. However, since a rule may have several different disequality atoms, it follows that we can have several occurrences of identical $\bar{P}_i$'s. But there are at most $\mathsf{n}_{\not\approx}$ occurrences of the

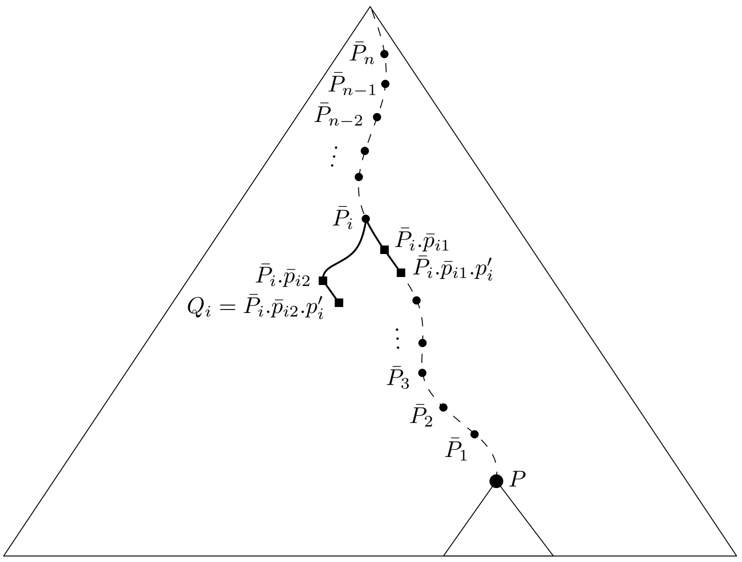


Figure 4.58: Abstract positions where a far disequality is falsified when performing the replacements $r[r_1]_P, \ldots, r[r_n]_P$ of Lemma 4.60: for each replacement $r[r_i]_P$, the pure abstract position $\bar{P}_i$ corresponds to the deepest pure abstract position where a far disequality becomes falsified in $r[r_i]_P$. The figure also representes for one of such $\bar{P}_i$ the two positions $\bar{p}_{i1}$ and $\bar{p}_{i2}$ involved in the falsified disequality atom of the rule $r(\bar{P}_i)$, and also the extension p'_i that guarantees that $\bar{P}_i.\bar{p}_{i2}.p'_i$ is pure and parallel to P, and $\bar{P}_i.\bar{p}_{i1}.p'_i$ is a prefix of P. Note that, since P is an abstract position, each replacement $r[r_i]_P$ actually involves simultaneous replacements at several parallel positions of r, and hence, the single path depicted in the figure should be understood as all paths in r with abstract position P.

same element, and hence, it is possible to take from $\bar{P}_1, \ldots, \bar{P}_n$ a selection with $n_1 := n/\mathsf{n}_{\not\approx}$ distinct pure abstract positions, say $\bar{P}_1, \ldots, \bar{P}_{n_1}$. Finally, we can take a new selection with $n_2 := n_1/(1 + \mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}})$ pure abstract positions from $\bar{P}_1, \ldots, \bar{P}_{n_1}$, say $\bar{P}_1, \ldots, \bar{P}_{n_2}$, that are more than $\mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}$ steps away from each other.

- We now consider each selected $\bar{P}_i$ and their corresponding positions $\bar{p}_{i1}, \bar{p}_{i2}$ of the far disequality tested at $\bar{P}_i$ and falsified by $r[r_i]_P$, and consider common extensions of $\bar{P}_i.\bar{p}_{i1}, \bar{P}_i.\bar{p}_{i2}$ defined by positions p'_i satisfying that either $\bar{P}_i.\bar{p}_{i1}.p'_i$ or $\bar{P}_i.\bar{p}_{i2}.p'_i$ is pure in r and the terms pending at such abstract positions in r are still different (and note that the terms pending at such abstract positions in $r[r_i]_P$ must coincide). We prove that for each of such p'_i, one of the extensions, say $\bar{P}_i.\bar{p}_{i1}.p'_i$, is a prefix of P, and that the other one, i.e., $\bar{P}_i.\bar{p}_{i2}.p'_i$, is parallel to P. Among all the possible p'_i, we choose a minimal one in size such that the extension $\bar{P}_i.\bar{p}_{i2}.p'_i$ parallel to P is pure. Thanks to the fact that the selected $\bar{P}_1, \ldots, \bar{P}_{n_2}$ are spaced between them by more than $\mathsf{h}_{\not\approx}+\mathsf{h}_{\mathrm{lhs}}$ steps, this extension can be done for each $\bar{P}_i$ without reaching any larger $\bar{P}_j$. Let Q_i be the extension $\bar{P}_i.\bar{p}_{i2}.p'_i$. We prove that the subruns $r|_{Q_1}, \ldots, r|_{Q_{n_2}}$ recognize distinct terms.

- At this point, we split $\{1, \ldots, n_2\}$ into two subsets depending on how close the corresponding $\bar{P}_i$'s are to P: one subset with the n_3 closest ones, and the other with the $n_4 := n_2 - n_3$ remaining ones. Say they are $\{1, \ldots, n_3\}$ and $\{n_3 + 1, \ldots, n_2\}$, respectively. Recall that each $\bar{P}_i$ is the maximal pure abstract position where a far disequality is falsified by the replacement $r[r_i]_P$. This means that the $\bar{P}_i$'s that are furthest from P necessarily correspond to the replacements where all the falsified far disequalities are "very far" from P. We now extract $M + 1$ indexes from $\{1, \ldots, n_3\}$, say $\{1, \ldots, M + 1\}$, such that, for each $i \in \{1, \ldots, M\}$, the replacement $r[r|_{Q_i}]_{Q_{M+1}}$ does not falsify any close disequality with respect to $\bar{P}_{M+1}$. Moreover, for each of such i, we extract a set of N indexes from $\{n_3 + 1, \ldots, n_2\}$ such that, for each of such indexes j, the simultaneous replacement $r[r|_{Q_i}]_{Q_{M+1}}[r_j]_P$ does not falsify any close disequality with respect to $\bar{P}_{M+1}$. We prove that $n_3 := |Q| \cdot \mathcal{B}_{\text{close}}(A, M{+}1{+}(2{\cdot}\mathsf{h}_{\not\approx}{+}\mathsf{h}_{\text{lhs}}){\cdot}\mathsf{n}_{\not\approx})$ suffices to guarantee the existence of such $M + 1$ indexes, where the factor $|Q|$ is required in order to guarantee that $r|_{Q_1}, \ldots, r|_{Q_M}$ reach the same state as $r|_{Q_{M+1}}$. For each of such $i \in \{1, \ldots, M\}$ the generation of the subset of size N from $\{n_3 + 1, \ldots, n_2\}$ must be done depending on i since, even though for all $j \in \{n_3{+}1, \ldots, n_2\}$ the replacement $r[r_j]_P$ does not falsify any disequality below $\bar{P}_{M+1}$, it might be the case that it falsifies some such disequality when combined with the replacement at Q_{M+1}. We prove that $n_4 := N + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) \cdot \mathsf{n}_{\not\approx}$ suffices to guarantee the existence of such N indexes.

  To summarize, it is possible to combine each of the M replacements at Q_{M+1} with the N corresponding replacements at P, and thus we can define the $M{\cdot}N \geq \mathcal{B}(A, M, N) = n$ needed candidates as runs of the form $r[r|_{Q_i}]_{Q_{M+1}}[r_j]_P|_{\bar{P}_{M+1}}$ and the abstract position P' as $\bar{P}_{M+1}$.

By considering the values given to n_1, n_2, n_3, and n_4 in the previous explanation, we can finally define the global bound $\mathcal{B}(A, M, N)$ and prove the main result of this section.

Definition 4.59. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let M, N be natural numbers. We define:*

$$\begin{aligned}\mathcal{B}(A, M, N) = \mathsf{n}_{\not\approx} \cdot (1 + \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) \cdot \big(&|Q| \cdot \mathcal{B}_{\text{close}}(A, M + 1 + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) \cdot \mathsf{n}_{\not\approx}) \\ &+ N + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) \cdot \mathsf{n}_{\not\approx}\big)\end{aligned}$$

Lemma 4.60. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let M, N be natural numbers satisfying $M \cdot N \geq \mathcal{B}(A, M, N)$. Let $n = \mathcal{B}(A, M, N)$. Let r be a uniform run of A, and let P be a pure abstract position of r. Let $r_1, \ldots, r_n$ be uniform runs of A on distinct terms reaching the same state as $r|_P$, and such that $r_1, \ldots, r_n \ll r|_P$ and each one of the replacements $r[r_1]_P, \ldots, r[r_n]_P$ falsifies at least one far disequality but does not falsify any close disequality.*

Then, there exists a pure abstract position $P' < P$ of r and uniform runs $r'_1, \ldots, r'_n$ of A on distinct terms reaching the same state as $r|_{P'}$, and such that $r'_1, \ldots, r'_n \ll r|_{P'}$ and each one of the replacements $r[r'_1]_{P'}, \ldots, r[r'_n]_{P'}$ does not falsify any close disequality.

Proof. For each $i \in \{1, \dots, n\}$, let $\bar{P}_i$ be the maximal pure abstract position such that the replacement $r[r_i]_P$ falsifies a far disequality tested at $\bar{P}_i$. Without loss of generality, we assume that $\bar{P}_1 \geq \bar{P}_2 \geq \dots \geq \bar{P}_n$ by reordering the runs r_i if necessary. Note that $|\mathsf{term}(r|_{\bar{P}_1})| \leq |\mathsf{term}(r|_{\bar{P}_2})| \leq \dots \leq |\mathsf{term}(r|_{\bar{P}_n})|$. By Lemma 4.56 and the fact that there are $\mathsf{n}_{\not\approx}$ different disequality atoms in the rules of A, it follows that for each pure abstract position $P' < P$ at most $\mathsf{n}_{\not\approx}$ of the pure abstract positions $\bar{P}_i$ coincide with P'. Thus, we can choose a subset S of $\{1, \dots, n\}$ with size

$$\begin{aligned} n' := n/(\mathsf{n}_{\not\approx} \cdot (1 + \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}})) = |Q| \cdot \mathcal{B}_{\text{close}}(A, M + 1 + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) \cdot \mathsf{n}_{\not\approx}) \\ + N + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) \cdot \mathsf{n}_{\not\approx} \end{aligned}$$

satisfying that $\bar{P}_j$ is more than $\mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}$ steps above $\bar{P}_i$ for each $i, j \in S$ with $i < j$, and that $\bar{P}_i$ is more than $\mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}$ steps above P for each $i \in S$. Without loss of generality, we assume that S is $\{1, \dots, n'\}$. Note that $\bar{P}_1 > \bar{P}_2 > \dots > \bar{P}_{n'}$ and $|\mathsf{term}(r|_{\bar{P}_1})| < |\mathsf{term}(r|_{\bar{P}_2})| < \dots < |\mathsf{term}(r|_{\bar{P}_{n'}})|$.

Consider any k in S. Since the replacement $r[r_k]_P$ falsifies a disequality tested at $\bar{P}_k$, it follows that there exist positions $\bar{p}_{k1}, \bar{p}_{k2}$ such that the atom $\bar{p}_{k1} \not\approx \bar{p}_{k2}$ occurs in the disequality constraint of the rule $r(\bar{P}_k)$, and moreover, $\mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k1}} \neq \mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k2}}$ and $\mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k1}} = \mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k2}}$. Let p_k be the shortest position such that at least one of $\bar{P}_k.\bar{p}_{k1}.p_k, \bar{P}_k.\bar{p}_{k2}.p_k$ is defined in r (i.e., corresponds to a pure abstract position of r), and moreover, $\mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k1}.p_k} \neq \mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k2}.p_k}$ and $\mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k1}.p_k} = \mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k2}.p_k}$. Note that $|\bar{p}_{k1}|, |\bar{p}_{k2}| \leq \mathsf{h}_{\not\approx}$ and $|p_k| \leq \mathsf{h}_{\text{lhs}}$, which implies that $\bar{P}_i$ is not a prefix of $\bar{P}_k.\bar{p}_{k1}.p_k$ or $\bar{P}_k.\bar{p}_{k2}.p_k$ for each $i \in S$ with $i < k$, and neither P is prefix of $\bar{P}_k.\bar{p}_{k1}.p_k$ or $\bar{P}_k.\bar{p}_{k2}.p_k$. Also note that $\bar{P}_k.\bar{p}_{k1}.p_k$ and $\bar{P}_k.\bar{p}_{k2}.p_k$ are necessarily parallel and at least one of them is a prefix of P, since otherwise, either $\mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k1}.p_k} \neq \mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k2}.p_k}$ or $\mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k1}.p_k} = \mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k2}.p_k}$ would be impossible. By Lemma 4.43, without loss of generality, we can assume that $\bar{P}_k.\bar{p}_{k1}.p_k$ is a prefix of P, and that $\bar{P}_k.\bar{p}_{k2}.p_k$ is not a prefix of P. Let p'_k be the shortest extension of p_k such that $\bar{P}_k.\bar{p}_{k2}.p'_k$ is defined in r (i.e., corresponds to a pure abstract position of r), and moreover, $\mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k1}.p'_k} \neq \mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k2}.p'_k}$ and $\mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k1}.p'_k} = \mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k2}.p'_k}$. Note that in the case where $\bar{P}_k.\bar{p}_{k2}.p_k$ already corresponds to a pure abstract position of r, then p'_k is just p_k. In any case, we have $|p'_k| \leq \mathsf{h}_{\text{lhs}}$. Let Q_k be the pure abstract position $\bar{P}_k.\bar{p}_{k2}.p'_k$. Observe that all of such Q_k are parallel with P.

We prove that the terms $\mathsf{term}(r|_{Q_1}), \dots, \mathsf{term}(r|_{Q_{n'}})$ are distinct by showing that $\mathsf{term}(r|_{Q_i})$ is a strict subterm of $\mathsf{term}(r|_{Q_j})$ for each $1 \leq i < j \leq n'$. From the fact that $\bar{P}_j$ is more than $\mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}$ steps above $\bar{P}_i$, it follows that $\bar{P}_j.\bar{p}_{j1}.p'_j$ is a strict prefix of $\bar{P}_i$, and thus also of Q_i. Hence, $\mathsf{term}(r|_{Q_i})$ is a strict subterm of $\mathsf{term}(r|_{\bar{P}_j})|_{\bar{p}_{j1}.p'_j}$. Moreover, since Q_i is parallel with P, $\mathsf{term}(r|_{Q_i})$ is also a strict subterm of $\mathsf{term}(r[r_j]_P|_{\bar{P}_j})|_{\bar{p}_{j1}.p'_j}$. Since $\mathsf{term}(r[r_j]_P|_{\bar{P}_j})|_{\bar{p}_{j1}.p'_j} = \mathsf{term}(r[r_j]_P|_{\bar{P}_j})|_{\bar{p}_{j2}.p'_j}$, it follows that $\mathsf{term}(r|_{Q_i})$ is also a strict subterm of $\mathsf{term}(r[r_j]_P|_{\bar{P}_j})|_{\bar{p}_{j2}.p'_j}$, i.e., of $\mathsf{term}(r[r_j]_P|_{Q_j})$. Finally, since Q_j is parallel with P, $\mathsf{term}(r|_{Q_i})$ is also a strict subterm of $\mathsf{term}(r|_{Q_j})$.

Let $m = M + 1 + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) \cdot \mathsf{n}_{\not\approx}$, and consider the first $|Q| \cdot \mathcal{B}_{\text{close}}(A, m)$ elements of S, i.e., $\{1, \dots, |Q| \cdot \mathcal{B}_{\text{close}}(A, m)\}$. Necessarily, there exists $\mathcal{B}_{\text{close}}(A, m)$ elements among them, say $\{1, \ldots, \mathcal{B}_{\text{close}}(A, m)\}$ without loss of generality, such that the subruns of r at the pure abstract positions $Q_1, \ldots, Q_{\mathcal{B}_{\text{close}}(A,m)}$ reach the same state. By Lemma 4.55, there exists a subset $\{i_1, \ldots, i_m\}$ of $\{1, \ldots, \mathcal{B}_{\text{close}}(A, m)\}$ such that $r|_{Q_{i_1}} \lll \ldots \lll r|_{Q_{i_m}}$ and the replacements $r[r|_{Q_{i_1}}]_{Q_{i_m}}, \ldots, r[r|_{Q_{i_{m-1}}}]_{Q_{i_m}}$ do not falsify any close disequality. Moreover, by Lemma 4.57, there exists a subset $\{j_1, \ldots, j_M\}$ of $\{i_1, \ldots, i_{m-1}\}$ such that the replacements $r[r|_{Q_{j_1}}]_{Q_{i_m}}, \ldots, r[r|_{Q_{j_M}}]_{Q_{i_m}}$ do not falsify any far disequality tested at most $2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}$ steps above Q_{i_m}. Note that, since $\bar{P}_{i_m}$ is at most $\mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}$ steps above Q_{i_m}, the replacements do not falsify any disequality tested at most $\mathsf{h}_{\not\approx}$ steps above $\bar{P}_{i_m}$, and hence, they do not falsify any close disequality with respect to $\bar{P}_{i_m}$.

Now, consider the last $|S| - |Q| \cdot \mathcal{B}_{\text{close}}(A, m)$ remaining elements of S. Observe that $S' := S \setminus \{1, \ldots, |Q| \cdot \mathcal{B}_{\text{close}}(A, m)\} = \{|Q| \cdot \mathcal{B}_{\text{close}}(A, m) + 1, \ldots, n'\} = \{n' - (N + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) \cdot \mathsf{n}_{\not\approx}) + 1, \ldots, n'\}$. Also, note that, for each $i \in S'$, the replacement $r[r_i]_P$ does not falsify any disequality tested below or at $\bar{P}_{i_m}$. Thus, for each $i \in S'$ and each $k \in \{1, \ldots, M\}$, since $\bar{P}_{i_m}.\bar{p}_{i_m 1}.p'_{i_m}$ and $\bar{P}_{i_m}.\bar{p}_{i_m 2}.p'_{i_m}$ are parallel, it follows that the replacement $r[r|_{Q_{j_k}}]_{Q_{i_m}}[r_i]_P$ does not falsify any disequality tested below or at $\bar{P}_{i_m}.\bar{p}_{i_m 1}.p'_{i_m}$. Recall that $|\bar{p}_{i_m 1}| \leq \mathsf{h}_{\not\approx}$ and $|p'_{i_m}| \leq \mathsf{h}_{\text{lhs}}$. By Lemma 4.57, for each fixed $k \in \{1, \ldots, M\}$, we can choose a subset S_k of S' with size N such that, for each $i \in S_k$, the replacement $r[r|_{Q_{j_k}}]_{Q_{i_m}}[r_i]_P$ does not falsify any disequality tested below or at $\bar{P}_{i_m}$, and moreover, it does not falsify any disequality tested at most $\mathsf{h}_{\not\approx}$ steps above $\bar{P}_{i_m}$, i.e., it does not falsify any close disequality with respect to $\bar{P}_{i_m}$. Let r_{ki} be $r[r|_{Q_{j_k}}]_{Q_{i_m}}[r_i]_P|_{\bar{P}_{i_m}}$ for each $k \in \{1, \ldots, M\}$ and each $i \in S_k$. Note that all such r_{ki}'s are uniform runs of A on distinct terms reaching the same state as $r|_{\bar{P}_{i_m}}$. Moreover, each of such r_{ki} satisfies $r_{ki} \lll r|_{\bar{P}_{i_m}}$, and the replacement $r[r_{ki}]_{\bar{P}_{i_m}}$, which in fact produces $r[r|_{Q_{j_k}}]_{Q_{i_m}}[r_i]_P$, does not falsify any close disequality with respect to $\bar{P}_{i_m}$. Observe that there are $M \cdot N \geq n$ of such r_{ki}'s. Thus, by defining P' as $\bar{P}_{i_m}$ and $r'_1, \ldots, r'_n$ as n of such r_{ki}'s, the lemma follows. ■

At this point it is clear that, by iterative applications of Lemma 4.60, we can construct a replacement that does not falsify any disequality or implicit equality constraint—i.e., a replacement that produces a run—whenever we have $\mathcal{B}(A, M, N)$ candidates for the replacement that do not falsify any close disequality. Moreover, note that since the candidates considered are smaller than the subrun being replaced, such replacement necessarily decreases the size of the starting run. The following corollary is an immediate consequence of this fact stating that, when the starting run is accepting, then it is not a minimum accepting run since we can decrease its size by performing such a replacement.

Corollary 4.61. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let M, N be natural numbers satisfying $M \cdot N \geq \mathcal{B}(A, M, N)$. Let $n = \mathcal{B}(A, M, N)$. Let r be an accepting uniform run of A, and let P be a pure abstract position of r. Let $r_1, \ldots, r_n$ be uniform runs of A on distinct terms reaching the same state as $r|_P$, and such that $r_1, \ldots, r_n \lll r|_P$ and each one of the replacements $r[r_1]_P, \ldots, r[r_n]_P$ does not falsify any close disequality.*

Then, r is not a minimum accepting run.

In order to conclude, it only remains to prove that there exist M and N satisfying $M \cdot N \geq \mathcal{B}(A, M, N)$. In the following lemma we give concrete values for M and N

that satisfy that condition. Note that there exist alternative definitions, but the ones we use are rather straightforward and lead to a simple proof.

Lemma 4.62. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let M, N be natural numbers defined as:*

$$\begin{aligned} M &= \mathsf{n}_{\not\approx} \cdot (1 + \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) + 1 \\ N &= \mathsf{n}_{\not\approx} \cdot (1 + \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) \cdot \big(|Q| \cdot \mathcal{B}_{\text{close}}(A, M + 1 + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) \cdot \mathsf{n}_{\not\approx}) \\ &\qquad\qquad + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) \cdot \mathsf{n}_{\not\approx}\big) \end{aligned}$$

Then, $M \cdot N = \mathcal{B}(A, M, N)$.

Proof. It follows by replacing the N in the definition of $\mathcal{B}$ by the definition of N in the statement, and factoring the result. More precisely, let $X = \mathsf{n}_{\not\approx} \cdot (1 + \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}})$ and $Y = \big(|Q| \cdot \mathcal{B}_{\text{close}}(A, M+1+(2\cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) \cdot \mathsf{n}_{\not\approx}) + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\text{lhs}}) \cdot \mathsf{n}_{\not\approx}\big)$, and note that $M = X + 1$ and $N = X \cdot Y$. Then, $\mathcal{B}(A, M, N) = X \cdot (Y + N) = X \cdot (Y + X \cdot Y) = X \cdot ((1 + X) \cdot Y) = (1 + X) \cdot X \cdot Y = M \cdot N$. ■

4.6 Emptiness decision algorithm

In this section we introduce an algorithm that decides emptiness of the language recognized by $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ in exponential time. In contrast to the previous section, we can now refrain from reasoning on runs since most of the information they provide is superfluous in our current setting. In particular, the only relevant data that the algorithm needs from a run is the term it recognizes and the state it reaches. For this reason, we focus on a formalism simpler than runs, namely, (term,state)-pairs defined as follows.

Definition 4.63. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. A (term,state)-pair $\langle t, q \rangle \in \mathcal{T}(\Sigma) \times Q$ is called a* feasible pair *of A if there exists a run of A on t reaching q, and moreover, it is called* accepting *if such run is accepting, i.e., if $q \in F$.*

We compare (term,state)-pairs by the lexicographic extension of $\ll$ and an arbitrary total ordering on states. We also use $\ll$ to denote the ordering on (term,state)-pairs.

Two sets of feasible pairs, called $\mathsf{Definitive}$ and $\mathsf{Candidates}$, are maintained as data structures of the algorithm. Initially, $\mathsf{Definitive}$ is empty, and $\mathsf{Candidates}$ has all the feasible pairs $\langle t, q \rangle$ such that there exists a run with only one applied rule on t and reaching q. At each iteration, the minimum pair $\langle t, q \rangle$ with respect to $\ll$ in $\mathsf{Candidates}$ is considered. The pair $\langle t, q \rangle$ is added to the set $\mathsf{Definitive}$ unless it is realized that it cannot be used to construct the minimum term with respect to $\ll$ of $\mathcal{L}(A)$. This fact can be detected using the results of the previous section. The following Corollary 4.66 is a direct consequence of combining Corollary 4.61 with Lemma 4.51, and translating the reasoning to the setting of (term,state)-pairs. We also provide Definitions 4.64 and 4.65 to simplify the notation.

Definition 4.64. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let $\langle t, q \rangle, \langle t', q' \rangle \in \mathcal{T}(\Sigma) \times Q$ be feasible (term,state)-pairs. We say that $\langle t', q' \rangle$ is a* piece of *$\langle t, q \rangle$ if there exists a run r of A on t reaching q such that there is a position $p \in \mathsf{Pos}(t)$ satisfying that $r(p)$ is defined, $r|_p$ reaches q', and $t|_p = t'$.*

Definition 4.65. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let M, N be natural numbers defined as in Lemma 4.62. We define $K(A)$ as $\mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx} + \mathcal{B}(A, M, N)$.*

Corollary 4.66. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let $t, s_1, \ldots, s_{K(A)}$ be distinct terms in $\mathcal{T}(\Sigma)$ with runs of A on them reaching a state $q \in Q$, and such that $\{t, s_1, \ldots, s_{K(A)}\}$ is $\mathsf{suff}_{\not\approx}$-independent, $t \sim^A s_1, \ldots, s_{K(A)}$, and $s_1, \ldots, s_{K(A)} \ll t$.*

Then, $\langle t, q\rangle$ is not a piece of the minimum accepting pair of A.

According to the previous corollary, in order to discard the addition of $\langle t, q\rangle$ to $\mathsf{Definitive}$, we should consider the set $\{s \mid \langle s, q\rangle \in \mathsf{Definitive} \;\wedge\; s \sim^A t\}$ and check whether it has a subset $\{s_1, \ldots, s_{K(A)}\}$ such that $\{t, s_1, \ldots, s_{K(A)}\}$ is $\mathsf{suff}_{\not\approx}$-independent. The time complexity of searching for such subset is too high for our goals. Fortunately, Section 4.4.2 gives us an alternative criterion to determine, in some cases, that such a subset exists. Along the execution of the algorithm we preserve an invariant stating that each of such sets $\{s \mid \langle s, q\rangle \in \mathsf{Definitive} \;\wedge\; s \sim^A t\}$ is a $(K(A){+}1, \mathsf{suff}_{\not\approx})$-small set of terms. If the addition of t to this set makes it non-$(K(A){+}1, \mathsf{suff}_{\not\approx})$-small, then, by Lemma 4.37, it follows the existence of the subset $\{s_1, \ldots, s_{K(A)}\}$ mentioned above. Thus, in this case we must discard the pair $\langle t, q\rangle$, since it is not a piece of the minimum accepting pair of A.

In the case where the pair $\langle t, q\rangle$ is not discarded, it is added to the set $\mathsf{Definitive}$ and used to generate new feasible (term,state)-pairs, which are added to $\mathsf{Candidates}$. This generation is performed (i) using the left-hand sides of rules in Δ to determine the symbols in the top-most positions of the new terms, (ii) using the feasible (term,state)-pairs in $\mathsf{Definitive}$ to instantiate the states appearing in such left-hand sides, and also (iii) guaranteeing that the specific pair $\langle t, q\rangle$ is used for the instantiation. This last condition ensures that all the pairs added to $\mathsf{Candidates}$ are new, i.e., that the algorithm has still not considered them to be added to $\mathsf{Definitive}$ (although they may be already in $\mathsf{Candidates}$ due to a previous generation). This generation is defined formally as follows.

Definition 4.67. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let $S \subseteq \mathcal{T}(\Sigma) \times Q$ be a set of feasible (term,state)-pairs. Let $\langle t, q\rangle \in S$ be a feasible (term,state)-pair. We define the* set of instantiations *of Δ with S and $\langle t, q\rangle$ as the set of feasible pairs:*

$$\begin{aligned}\big\{\langle t' = l[t_1]_{p_1} \ldots [t_n]_{p_n}, q'\rangle \mid\ & \exists (l \xrightarrow{c} q') \in \Delta,\ \{p_1, \ldots, p_n\} = \mathsf{Pos}_Q(l),\\ & \forall i \in \{1, \ldots, n\} : \langle t_i, l(p_i)\rangle \in S,\\ & \exists i \in \{1, \ldots, n\} : \langle t_i, l(p_i)\rangle = \langle t, q\rangle,\\ & \forall i, j \in \{1, \ldots, n\} : (l(p_i) = l(p_j) \Rightarrow t_i = t_j),\\ & \forall (\bar{p}_1 \not\approx \bar{p}_2) \in c : (\bar{p}_1, \bar{p}_2 \in \mathsf{Pos}(t') \Rightarrow t'|_{\bar{p}_1} \neq t'|_{\bar{p}_2})\big\}\end{aligned}$$

Example 4.68. *Consider a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ A over $\{a{:}0, f{:}2\}$ with set of states $\{q_1, q_2, q_3\}$ and set of rules $\Delta = \{a \to q_1|q_2,\ f(q_1, q_2) \to q_1|q_2,\ f(q_1, q_1) \to q_3\}$, where we use the notation $l \to q_1|q_2$ to simultaneously denote the rules $l \to q_1$ and $l \to q_2$. Consider the set of feasible (term,state)-pairs $S = \{\langle a, q_1\rangle, \langle a, q_2\rangle\}$. The set of instantiations of Δ with S and $\langle a, q_1\rangle \in S$ is obtained as follows, were we take into account the conditions of Definition 4.67 progressively. First, we need to consider all the rules in Δ and replace, in all possible ways, each of the states occurring at their left-hand sides by the term part of a pair in S, ensuring that the state part of such pair* *matches the state being replaced. Thus, we would obtain $\langle a, q_1\rangle$, $\langle a, q_2\rangle$ from the rules $a \to q_1|q_2$ (where no state replacement is performed), $\langle f(a,a), q_1\rangle$, $\langle f(a,a), q_2\rangle$ from the rules $f(q_1,q_2) \to q_1|q_2$ (obtained by replacing q_1 and q_2 by the terms of the pairs $\langle a, q_1\rangle, \langle a, q_2\rangle \in S$, respectively), and $\langle f(a,a), q_3\rangle$ from the rule $f(q_1,q_1) \to q_3$ (obtained by replacing twice q_1 by the term of the pair $\langle a, q_1\rangle \in S$). Nevertheless, since we are doing the instantiation of Δ with S and $\langle a, q_1\rangle \in S$, Definition 4.67 requires that $\langle a, q_1\rangle \in S$ is used to replace at least one of the states occurring at the left-hand side of the rules. Thus, rules $a \to q_1|q_2$ cannot actually be used in the instantiation of this example, and the pairs obtained are just $\langle f(a,a), q_1\rangle$, $\langle f(a,a), q_2\rangle$, $\langle f(a,a), q_3\rangle$ since all of them have used $\langle a, q_1\rangle$ to replace (at least) an occurrence of q_1. Finally, the last two conditions of Definition 4.67 ensure that the pairs generated are feasible, meaning that the (dis)equality constraints of the rules used in the instantiation are respected. In this example there is no disequality constraint, and the single implicit equality constraint of A occurs in the rule $f(q_1,q_1) \to q_3$ between positions 1, 2, and has been respected when instantiating the pair $\langle f(a,a), q_3\rangle$. In summary, the set of instantiations of Δ with S and $\langle a, q_1\rangle \in S$ is $\{\langle f(a,a), q_1\rangle, \langle f(a,a), q_2\rangle, \langle f(a,a), q_3\rangle\}$.*

When there are no more pairs in Candidates to be considered, the algorithm stops and states non-emptiness if there is a (term,state)-pair in Definitive where the state is final. We present in Algorithm 4.69 a formalization of the previous explanations.

Algorithm 4.69 Emptiness decision for the language recognized by a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ A.

```
Input: a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ $A = \langle Q, \Sigma, F, \Delta\rangle$.
Data structures: Definitive, Candidates sets of elements in $\mathcal{T}(\Sigma) \times Q$.
  (1) Insert in Candidates the pairs $\langle l, q\rangle$ such that there exists a rule $(l \xrightarrow{c} q) \in \Delta$
      with $l \in \mathcal{T}(\Sigma)$ and satisfying $\forall(\bar{p}_1 \not\approx \bar{p}_2) \in c : (\bar{p}_1, \bar{p}_2 \in \mathsf{Pos}(l) \Rightarrow l|_{\bar{p}_1} \neq l|_{\bar{p}_2})$.
  (2) While Candidates is not empty:
      (a) Let $\langle t, q\rangle$ be the smallest pair in Candidates with respect to $\ll$.
      (b) Remove $\langle t, q\rangle$ from Candidates.
      (c) If $\{s \mid \langle s, q\rangle \in \mathsf{Definitive} \wedge s \sim^A t\} \uplus \{t\}$ is $(K(A)+1, \mathsf{suff}_{\not\approx})$-small:
          (i) Insert $\langle t, q\rangle$ in Definitive.
          (ii) Insert in Candidates all the elements in the set of instantiations of $\Delta$
               with Definitive and $\langle t, q\rangle$.
  (3) If there is a pair $\langle t, q\rangle \in$ Definitive with $q \in F$, then output 'NON-EMPTY', else
      output 'EMPTY'.
```

Example 4.70. *Consider the signature Σ with binary symbols h, g and a nullary symbol a, and the language of terms over Σ of the form $h(t_1, t_2)$ satisfying that t_1, t_2 are different complete trees over g and a. Such language is recognized by the* $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ $A = \langle\{q, q', q_{\mathsf{accept}}\}, \Sigma, \{q_{\mathsf{accept}}\}, \{a \to q|q',\ g(q,q) \to q|q',\ h(q,q') \xrightarrow{1\not\approx 2} q_{\mathsf{accept}}\}\rangle$, *where again we use $l \to q|q'$ to simultaneously denote the rules $l \to q$ and $l \to q'$.*

In order to apply Algorithm 4.69, we first need to fix an ordering $\ll$ for terms. We choose a natural recursive definition: for distinct terms t, t', if $|t| < |t'|$, then $t \ll t'$, and in the case $|t| = |t'|$ (which implies that the sizes are at least 2 since $t \neq t'$), $t \ll t'$

if $(t(\lambda) = g \wedge t'(\lambda) = h) \vee (t(\lambda) = t'(\lambda) \wedge t|_1 \ll t'|_1) \vee (t(\lambda) = t'(\lambda) \wedge t|_1 = t'|_1 \wedge t|_2 \ll t'|_2)$*. To lexicographically extend the ordering to (term,state)-pairs we simply assume* $\langle t, q\rangle \ll \langle t, q'\rangle \ll \langle t, q_{\mathsf{accept}}\rangle$*.*

We execute Algorithm 4.69 step by step. First, $\mathsf{Candidates} := \{\langle a, q\rangle,\ \langle a, q'\rangle\}$ *is set at Step 1. Second, Step 2 is executed repeatedly, where the first four iterations proceed as follows (to ease the presentation, we do not discuss* $(K(A){+}1, \mathsf{suff}_{\not\approx})$*-smallness in detail):*

1. *The* $\ll$*-minimum pair extracted from* Candidates *is* $\langle a, q\rangle$*. Since* Definitive *is still empty, Step 2.c is satisfied with a trivial* $(K(A){+}1, \mathsf{suff}_{\not\approx})$*-smallness test. Hence,* $\mathsf{Definitive} := \{\langle a, q\rangle\}$*, and the pair* $\langle a, q\rangle$ *and the current* Definitive *are used to instantiate* 2 *new feasible pairs for* Candidates*:*

$$\mathsf{Candidates} := \{\langle a, q'\rangle\} \uplus \{\langle g(a,a), q\rangle,\ \langle g(a,a), q'\rangle\}$$

2. *The* $\ll$*-minimum pair extracted from* Candidates *is* $\langle a, q'\rangle$*. Now,* Definitive *is non-empty, but its subset* $\{s \mid \langle s, q'\rangle \in \mathsf{Definitive}\}$ *is, and thus, Step 2.c is satisfied. Hence,* $\mathsf{Definitive} := \{\langle a, q\rangle,\ \langle a, q'\rangle\}$*. In this case, the instantiation does not produce any feasible pair: since it is required that the current pair* $\langle a, q'\rangle$ *is used in the instantiation, the only transition rule that we can use is* $h(q, q') \xrightarrow{1 \not\approx 2} q_{\mathsf{accept}}$*, but it is easy to see that the disequality constraint* $1 \not\approx 2$ *cannot be satisfied with the current feasible pairs in* Definitive*. Hence:*

$$\mathsf{Candidates} = \{\langle g(a,a), q\rangle,\ \langle g(a,a), q'\rangle\}$$

3. *The* $\ll$*-minimum pair extracted from* Candidates *is* $\langle g(a,a), q\rangle$*. Now, note that* $\{s \mid \langle s, q\rangle \in \mathsf{Definitive}\ \wedge\ s \sim^A g(a,a)\}$ *is empty since* $\mathsf{suff}_{\not\approx} = \{\lambda, 1, 2\}$*. Thus, again Step 2.c is satisfied. Hence,* $\mathsf{Definitive} := \{\langle a, q\rangle,\ \langle a, q'\rangle,\ \langle g(a,a), q\rangle\}$*, and the instantiation produces* 3 *new feasible pairs:*

$$\begin{aligned}\mathsf{Candidates} := \{\langle g(a,a), q'\rangle\} \uplus \{&\langle g(g(a,a), g(a,a)), q\rangle,\ \langle g(g(a,a), g(a,a)), q'\rangle,\\ &\langle h(g(a,a), a), q_{\mathsf{accept}}\rangle\}\end{aligned}$$

   At this point it would be possible to conclude $\mathcal{L}(A) \neq \emptyset$ *due to the presence of the feasible pair* $\langle h(g(a,a), a), q_{\mathsf{accept}}\rangle$ *in* Candidates*. Nevertheless, the algorithm does not stop its execution until* Candidates *is empty.*

4. *The* $\ll$*-minimum pair extracted from* Candidates *is* $\langle g(a,a), q'\rangle$*. As before, Step 2.c is satisfied since* $\{s \mid \langle s, q'\rangle \in \mathsf{Definitive}\ \wedge\ s \sim^A g(a,a)\}$ *is empty. Hence,* $\mathsf{Definitive} := \{\langle a, q\rangle,\ \langle a, q'\rangle,\ \langle g(a,a), q\rangle,\ \langle g(a,a), q'\rangle\}$*, and the instantiation produces* 1 *new feasible pair:*

$$\begin{aligned}\mathsf{Candidates} := \{&\langle g(g(a,a), g(a,a)), q\rangle,\ \langle g(g(a,a), g(a,a)), q'\rangle,\\ &\langle h(g(a,a), a), q_{\mathsf{accept}}\rangle\} \uplus \{\langle h(a, g(a,a)), q_{\mathsf{accept}}\rangle\}\end{aligned}$$

In the next iteration of Step 2, the algorithm will extract from Candidates *the pair* $\langle h(a, g(a,a)), q_{\mathsf{accept}}\rangle$ *generated in the 4th iteration (which is, in fact, the* $\ll$*-minimum* *accepting pair of A), and insert it into* Definitive*. Hence, once* Candidates *is completely emptied, the algorithm will halt with output* `'NON-EMPTY'`*, as expected. Note that, in the detailed iterations, the feasible pairs instantiated are all new, even though this is in general not the case: the instantiation might produce feasible pairs that are already in* Candidates *(the new pairs are only guaranteed to be $\ll$-greater than all the feasible pairs in* Definitive*).*

Now it remains to prove that our algorithm is correct and terminates in the desired time. We start stating its time complexity.

Lemma 4.71. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$.*
Then, Algorithm 4.69 on input A takes time in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2\cdot|\mathsf{Pos}_{\mathrm{lhs}}|\cdot\log|A|)}$.

Proof. Let A be $\langle Q,\Sigma,F,\Delta\rangle$ more explicitly written. First note that, by Definitions 4.52, 4.54, 4.59, and 4.65, and Lemma 4.62, $K(A)$ is in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|\cdot\log|A|)}$. Now, consider any maximal subset of Definitive with (term,state)-pairs having the same state and with all the terms belonging to the same equivalence class of $\sim^A$, and note that, since Algorithm 4.69 guarantees that such subset is $(K(A){+}1,\mathsf{suff}_{\not\approx})$-small, by Lemma 4.35 it follows that its size is in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2\cdot\log|A|)}$. By Lemma 4.53, there are $|Q|\cdot\mathcal{B}_{\mathrm{eq}}(A)$ of such subsets, and thus, we have that the size of Definitive is in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2\cdot\log|A|)}$. Each time a pair is added to Definitive, new pairs are generated and added to Candidates. When this happens, the maximum number of new pairs that can be generated in the instantiation is bounded by $|\Delta|\cdot|\mathsf{Definitive}|^{|\mathsf{Pos}_{\mathrm{lhs}}|}$. Hence, the number of pairs inserted in Candidates during the whole execution is in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2\cdot|\mathsf{Pos}_{\mathrm{lhs}}|\cdot\log|A|)}$. Since each iteration of the algorithm removes a pair from Candidates, it follows that the number of iterations is in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2\cdot|\mathsf{Pos}_{\mathrm{lhs}}|\cdot\log|A|)}$.

It remains to prove that each iteration takes time in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2\cdot|\mathsf{Pos}_{\mathrm{lhs}}|\cdot\log|A|)}$. To avoid double exponential blowup, we consider a directed acyclic graph (DAG) representation for terms. More precisely, the algorithm uses a DAG as an internal data structure, where each node is labeled by a symbol $f\in\Sigma$ and has $\mathsf{arity}(f)$ ordered outedges. In this way, each node of the DAG implicitly represents a term over Σ, and thus, each of the terms considered by the algorithm is simply a reference to the appropriate node in the DAG. Additionally, we consider that the DAG is minimum, meaning that each two distinct nodes of the DAG represent different terms (note that each time a new node has to be inserted into the DAG, it can be checked in linear time with respect to the size of the DAG whether the DAG already contains a node representing the desired term). This implies that the total size of the representation, i.e., the size of the DAG, equals the number of distinct terms generated by the algorithm, which is in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2\cdot|\mathsf{Pos}_{\mathrm{lhs}}|\cdot\log|A|)}$. Now, we analyse the cost of performing each of the steps in the loop when using such an internal data structure. For Step 2.a, it is necessary to select the minimum pair in Candidates with respect to $\ll$. To this end, it suffices to compute, for each two distinct nodes of the DAG representing terms t,t', whether $t\ll t'$. The cost of this computation is polynomial with respect to the size of the DAG when using a dynamic programming scheme, and hence, this step takes time in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2\cdot|\mathsf{Pos}_{\mathrm{lhs}}|\cdot\log|A|)}$. For Step 2.c, $\sim^A$-equivalence with a term t must be checked: this consists in testing, for each node of the DAG representing a term t' and each two positions $p_1,p_2\in\mathsf{suff}_{\not\approx}$, whether the subterms of t,t' pending at positions p_1,p_2 exist, and when they do, whether $t|_{p_1}, t|_{p_2}$ are represented by the same node of the DAG (i.e., are equal terms) if and only if $t'|_{p_1}, t'|_{p_2}$ are represented by the same node of the DAG (i.e., are equal terms). Also for Step 2.c, it is necessary to test $(K(A)+1, \mathsf{suff}_{\not\approx})$-smallness: by Lemma 4.36, this requires at most $|\mathsf{Definitive}|^2 \cdot 2^{|\mathsf{suff}_{\not\approx}|} \cdot |\mathsf{suff}_{\not\approx}|$ equality comparisons between subterms, and such comparisons consist simply in checking whether the nodes of the DAG representing the involved subterms coincide. Thus, this step also takes time in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot |\mathsf{Pos}_{\mathrm{lhs}}| \cdot \log |A|)}$. Finally, for Step 2.c.ii, we need to insert into the DAG the newly instantiated terms, and also avoid the insertion of repeated elements in $\mathsf{Candidates}$. As justified before, inserting a node into the DAG takes linear time with respect to the size of the DAG, as we need to keep the DAG minimum. Avoiding repeated elements in $\mathsf{Candidates}$ is straightforward: it suffices to check whether $\mathsf{Candidates}$ already contains a (term,state)-pair with the same state and referencing the same node of the DAG as the (term,state)-pair to be inserted. Hence, this also takes time in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot |\mathsf{Pos}_{\mathrm{lhs}}| \cdot \log |A|)}$, and we are done. ■

We now prove Algorithm 4.69 to be correct. We start stating the following trivial property of our algorithm: whenever a feasible pair is not generated, it is due to having previously discarded another pair needed for its construction.

Lemma 4.72. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Let $\langle t, q \rangle \in \mathcal{T}(\Sigma) \times Q$ be a feasible (term,state)-pair of A satisfying that it is not generated by Algorithm 4.69 on input A.*

Then, there exists a feasible pair $\langle t', q' \rangle \in \mathcal{T}(\Sigma) \times Q$ of A such that it is a piece of $\langle t, q \rangle$ and is discarded by Algorithm 4.69 on input A.

Proof. We proceed by induction on $\mathsf{height}(t)$. Let r be a run of A on t reaching q and let $l \xrightarrow{c} q$ be the rule applied at root position in r. Let $p_1, \ldots, p_m$ be the positions in $\mathsf{Pos}_Q(l)$. Note that necessarily $m > 0$, since otherwise the pair $\langle t, q \rangle$ is generated at Step 1 of Algorithm 4.69. Let $t_1, \ldots, t_m$ be the terms $t|_{p_1}, \ldots, t|_{p_m}$ and $q_1, \ldots, q_m$ be the states $l(p_1), \ldots, l(p_m)$, respectively. Note that $\mathsf{height}(t_i) < \mathsf{height}(t)$ since $p_i \neq \lambda$, and that q_i is the state reached by the subrun $r|_{p_i}$, for all $i \in \{1, \ldots, m\}$. Since $\langle t, q \rangle$ is not generated by Algorithm 4.69 on input A, then there exists $i \in \{1, \ldots, m\}$ satisfying that $\langle t_i, q_i \rangle$ is either discarded or not generated. Note that such i must exist, since otherwise, $\langle t, q \rangle$ is generated at Step 2.c.ii of Algorithm 4.69, contradicting the definition of $\langle t, q \rangle$.

In order to conclude, first assume that $\langle t_i, q_i \rangle$ is discarded. In this case, the statement holds by defining the pair $\langle t', q' \rangle$ of the lemma as $\langle t_i, q_i \rangle$. Next, assume that $\langle t_i, q_i \rangle$ is not generated. In this case, by induction hypothesis, there exists a feasible pair $\langle \hat{t}, \hat{q} \rangle \in \mathcal{T}(\Sigma) \times Q$ of A such that it is a piece of $\langle t_i, q_i \rangle$ and is discarded by Algorithm 4.69 on input A. Hence, the statement holds by defining the pair $\langle t', q' \rangle$ of the lemma as $\langle \hat{t}, \hat{q} \rangle$. ■

We are finally ready to prove the soundness and completeness of Algorithm 4.69.

Lemma 4.73. *Let A be a $\mathtt{TA}_{\mathsf{ihom},\not\approx}$.*

Then, $\mathcal{L}(A)$ is empty if and only if Algorithm 4.69 on input A outputs `'EMPTY'`.

Proof. Let A be $\langle Q, \Sigma, F, \Delta\rangle$ more explicitly written. The left-to-right direction follows trivially, since in this case it is not possible to generate a feasible pair $\langle t, q\rangle \in \mathcal{T}(\Sigma) \times Q$ of A satisfying that $q \in F$, and hence, the algorithm necessarily outputs `'EMPTY'`. For the other direction, we proceed by contradiction by assuming that there exists an accepted term, but the algorithm cannot find any accepting pair of A and outputs `'EMPTY'`. Let $\langle t, q\rangle \in \mathcal{T}(\Sigma) \times Q$ be the minimum accepting pair of A with respect to $\ll$. By assumption, $\langle t, q\rangle$ is either discarded or not generated by the algorithm. The former case is not possible, since it implies that $\mathsf{Definitive}$ contains at least one pair $\langle t', q\rangle$ such that $t' \sim^A t$, and therefore $\langle t', q\rangle$ is an accepting pair of A, contradicting the fact that the algorithm finds no accepting pairs of A. Hence, assume that $\langle t, q\rangle$ is not generated by the algorithm. By Lemma 4.72, it follows that there exists a feasible pair $\langle t', q'\rangle$ of A such that it is a piece of $\langle t, q\rangle$ and is discarded by the algorithm. Consider that the execution of the algorithm is at the iteration when the pair $\langle t', q'\rangle$ is discarded and let S be $\{s \mid \langle s, q'\rangle \in \mathsf{Definitive} \wedge s \sim^A t'\} \uplus \{t'\}$. We know that $S \setminus \{t'\}$ is $(K(A)+1, \mathsf{suff}_{\not\approx})$-small, but S is not. Hence, by Lemma 4.37, it follows that there exists a $\mathsf{suff}_{\not\approx}$-independent set of terms $\tilde{S} \subseteq S$ including t' and satisfying $|\tilde{S}| \geq K(A) + 1$. By definition, all the terms in $\tilde{S} \setminus \{t'\}$ also have runs of A on them reaching the state q', are $\sim^A$-equivalent to t', and smaller than t' with respect to $\ll$. By Corollary 4.66, it follows that $\langle t', q'\rangle$ is not a piece of the minimum accepting pair of A, contradicting the selection of $\langle t, q\rangle$. ■

The next corollary follows from Lemmas 4.71 and 4.73.

Corollary 4.74. *Emptiness of the language recognized by a* $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ *A can be decided with time in* $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot |\mathsf{Pos}_{\mathrm{lhs}}| \cdot \log |A|)}$.

4.7 Consequences

In this section we state the consequences that follow from the decidability of the emptiness problem for $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ in exponential time. Since the constructions from [GG13] summarized in Section 4.2 deal with the class $\mathtt{TA}_{\mathsf{hom},\not\approx}$, we begin by translating Corollary 4.74 to the setting of $\mathtt{TA}_{\mathsf{hom},\not\approx}$.

Corollary 4.75. *Emptiness of the language recognized by a* $\mathtt{TA}_{\mathsf{hom},\not\approx}$ *A can be decided with time in* $2^{\mathcal{O}((\mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx})^2 \cdot |\mathsf{Pos}_{\mathrm{lhs}}| \cdot \log |A|)}$.

Proof. Follows from Lemma 4.16, Corollary 4.74, and the definition of $\mathsf{suff}_{\not\approx}$ for $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. ■

Theorem 4.76. *Deciding emptiness and finiteness of the language recognized by a* $\mathtt{TA}_{\mathsf{hom},\not\approx}$ *A is in EXPTIME.*

Proof. Decidability of emptiness in exponential time follows directly from Corollary 4.75. In the case of finiteness, by Proposition 4.10, a $\mathtt{TA}_{\mathsf{hom},\not\approx}$ A' such that $\mathcal{L}(A')$ is empty if and only if $\mathcal{L}(A)$ is finite can be computed in exponential time with respect to $|A|$. Moreover, the bounds on $\mathsf{h}_{\not\approx}(A')$, $\mathsf{n}_{\not\approx}(A')$, $|\mathsf{Pos}_{\mathrm{lhs}}(A')|$, and $|A'|$ are such that, by Corollary 4.75, emptiness of $\mathcal{L}(A')$ can be decided in exponential time with respect to $|A|$. ■

Note that, since $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ can be transformed into $\mathtt{TA}_{\mathsf{hom},\not\approx}$ in polynomial time as detailed in Lemma 4.19, from the previous result it follows that the finiteness problem for $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ is also decidable in exponential time.

Now we give decidability results on the images of regular tree languages under tree homomorphisms.

Theorem 4.77. *The inclusion problem for images of regular tree languages under tree homomorphisms, i.e., deciding $H_1(\mathcal{L}(A_1)) \subseteq H_2(\mathcal{L}(A_2))$ for* TA *A_1, A_2 and tree homomorphisms H_1, H_2 given as input, is EXPTIME-complete.*

Proof. By Proposition 4.4, two $\mathtt{TA}_{\mathsf{hom}}$ A_1', A_2' recognizing $H_1(\mathcal{L}(A_1))$ and $H_2(\mathcal{L}(A_2))$, respectively, can be computed in polynomial time with respect to the size of the input. By Propositions 4.8 and 4.9, a $\mathtt{TA}_{\mathsf{hom},\not\approx}$ A recognizing $\mathcal{L}(A_1') \cap \overline{\mathcal{L}(A_2')}$ can be computed in exponential time with respect to the size of the input. Moreover, the bounds on $\mathsf{h}_{\not\approx}(A)$, $\mathsf{n}_{\not\approx}(A)$, $|\mathsf{Pos}_{\text{lhs}}(A)|$, and $|A|$ are such that, by Corollary 4.75, emptiness of $\mathcal{L}(A)$ can be decided in exponential time with respect to the size of the input. Thus, we conclude by noting that emptiness of $\mathcal{L}(A_1') \cap \overline{\mathcal{L}(A_2')}$ is equivalent to $H_1(\mathcal{L}(A_1)) \subseteq H_2(\mathcal{L}(A_2))$, and that the problem is EXPTIME-hard by Proposition 4.1. ■

Corollary 4.78. *The equivalence problem for images of regular tree languages under tree homomorphisms, i.e., deciding $H_1(\mathcal{L}(A_1)) = H_2(\mathcal{L}(A_2))$ for* TA *A_1, A_2 and tree homomorphisms H_1, H_2 given as input, is EXPTIME-complete.*

Corollary 4.79. *The inclusion and equivalence problems for ranges of bottom-up tree transducers are EXPTIME-complete.*

Theorem 4.80. *The finite difference problem for images of regular tree languages under tree homomorphisms, i.e., deciding finiteness of $H_1(\mathcal{L}(A_1)) \setminus H_2(\mathcal{L}(A_2))$ for* TA *A_1, A_2 and tree homomorphisms H_1, H_2 given as input, is EXPTIME-complete.*

Proof. By Propositions 4.4, 4.8, 4.9 and 4.10, a $\mathtt{TA}_{\mathsf{hom},\not\approx}$ A such that $\mathcal{L}(A)$ is empty if and only if $H_1(\mathcal{L}(A_1)) \cap \overline{H_2(\mathcal{L}(A_2))}$ is finite can be computed in exponential time with respect to the size of the input. Moreover, the bounds on $\mathsf{h}_{\not\approx}(A)$, $\mathsf{n}_{\not\approx}(A)$, $|\mathsf{Pos}_{\text{lhs}}(A)|$, and $|A|$ are such that, by Corollary 4.75, emptiness of $\mathcal{L}(A)$ can be decided in exponential time with respect to the size of the input. Thus, we conclude by noting that finiteness of $H_1(\mathcal{L}(A_1)) \cap \overline{H_2(\mathcal{L}(A_2))}$ is equivalent to finiteness of $H_1(\mathcal{L}(A_1)) \setminus H_2(\mathcal{L}(A_2))$, and that the problem is EXPTIME-hard by Proposition 4.1. ■

Our results have also implications in the context of term rewriting. The set of reducible terms of a term rewrite system can be described as the image of a regular tree language under a tree homomorphism, and the set of normal forms, i.e., the set of terms for which no rule can be applied, is just its complement. Thus, we can decide inclusion and equality of such sets with respect to two given term rewrite systems in exponential time. Since ground reducibility is a particular case of such problems and it is shown EXPTIME-hard in [CJ03], we conclude that these problems are EXPTIME-complete.

Corollary 4.81. *Deciding $\mathsf{Red}(R_1) = \mathsf{Red}(R_2)$, $\mathsf{Red}(R_1) \subseteq \mathsf{Red}(R_2)$, $\mathsf{NF}(R_1) = \mathsf{NF}(R_2)$, and $\mathsf{NF}(R_1) \subseteq \mathsf{NF}(R_2)$ for given term rewrite systems R_1, R_2 is EXPTIME-complete, where $\mathsf{Red}(R)$ and $\mathsf{NF}(R)$ denote the set of reducible terms and the set of normal forms, respectively, with respect to R.*

In [GT95], the question $\mathsf{Rel}(L_1) \subseteq L_2$ is shown decidable for given regular tree languages L_1, L_2 and where the relation Rel is defined in several ways according to a given term rewrite system R. Tree homomorphisms are used to describe the image of L_1 through this relation: two tree homomorphisms H_l and H_r, and a tree language R_c are defined satisfying $\mathsf{Rel}(L_1) = H_r(H_l^{-1}(L_1) \cap R_c)$, so that deciding $\mathsf{Rel}(L_1) \subseteq L_2$ is done by testing $H_r(H_l^{-1}(L_1) \cap R_c) \subseteq L_2$. The tree homomorphisms H_l, H_r depend only on the rewrite system R. The tree language R_c depends also on the relation Rel. Our results allow to improve the results in [GT95] where R_c is a regular tree language. These are when Rel is one of the following relations: the one rewriting step, the one parallel rewriting step, the one-pass innermost-outermost step for left-linear term rewrite systems, and the one-pass outermost-innermost step for right-linear term rewrite systems (see [GT95] for details). In those cases, we are able to extend the results to decide the question $\mathsf{Rel}_1(L_1) \subseteq \mathsf{Rel}_2(L_2)$: analogously, tree homomorphisms $H_{1,l}, H_{1,r}, H_{2,l}, H_{2,r}$ and regular tree languages $R_{1,c}, R_{2,c}$ can be defined such that $\mathsf{Rel}_1(L_1) = H_{1,r}(H_{1,l}^{-1}(L_1) \cap R_{1,c})$ and $\mathsf{Rel}_2(L_2) = H_{2,r}(H_{2,l}^{-1}(L_2) \cap R_{2,c})$, so that deciding $\mathsf{Rel}_1(L_1) \subseteq \mathsf{Rel}_2(L_2)$ is done by testing $H_{1,r}(H_{1,l}^{-1}(L_1) \cap R_{1,c}) \subseteq H_{2,r}(H_{2,l}^{-1}(L_2) \cap R_{2,c})$. Under the given assumptions, $H_{1,l}^{-1}(L_1) \cap R_{1,c}$ and $H_{2,l}^{-1}(L_2) \cap R_{2,c}$ are regular languages. Thus, the above inclusion relates two images of regular tree languages under tree homomorphisms.

Corollary 4.82. *Deciding the inclusion $H_{1,r}(H_{1,l}^{-1}(L_1) \cap R_{1,c}) \subseteq H_{2,r}(H_{2,l}^{-1}(L_2) \cap R_{2,c})$ is EXPTIME-complete for given tree homomorphisms $H_{1,l}, H_{1,r}, H_{2,l}, H_{2,r}$ and given regular tree languages $L_1, L_2, R_{1,c}, R_{2,c}$.*

Corollary 4.83. *Deciding the equality $\mathsf{Rel}_1(L_1) = \mathsf{Rel}_2(L_2)$ and inclusion $\mathsf{Rel}_1(L_1) \subseteq \mathsf{Rel}_2(L_2)$ is EXPTIME-complete for given regular tree languages L_1, L_2 and a given term rewrite system R, where $\mathsf{Rel}_1, \mathsf{Rel}_2$ are defined as either the one rewriting step, the one parallel rewriting step, the one-pass innermost-outermost step if R is left-linear, and the one-pass outermost-innermost step if R is right-linear.*

Now it only remains to tackle the HOM problem. Recall that, by Lemma 4.6, given a $\mathtt{TA}_{\mathsf{hom}}$ A recognizing the image of a regular tree language under a tree homomorphism, $\mathcal{L}(A)$ is regular if and only if $\mathcal{L}(A) \cap \overline{\mathcal{L}(\mathsf{linearize}(A, \check{h}))}$ is empty. Unfortunately, the cost of computing a $\mathtt{TA}_{\mathsf{hom},\not\approx}$ recognizing $\mathcal{L}(A) \cap \overline{\mathcal{L}(\mathsf{linearize}(A, \check{h}))}$ is triple exponential when using the constructions from [GG13], as summarized in Definition 4.5 and Propositions 4.8 and 4.9. In order to lower it to a single exponential, we extend several ideas and transformations from [GG13]. In particular, we refine Definition 4.5 and Proposition 4.8 to obtain in exponential time a $\mathtt{TA}_{\not\approx}$ recognizing $\overline{\mathcal{L}(\mathsf{linearize}(A, \check{h}))}$. The desired $\mathtt{TA}_{\mathsf{hom},\not\approx}$ can then be obtained by intersecting with Proposition 4.9 such $\mathtt{TA}_{\not\approx}$ with the given $\mathtt{TA}_{\mathsf{hom}}$.

In Definition 4.84 we propose a construction that, for a given $\mathtt{TA}_{\mathsf{hom}}$ A and natural number $\mathtt{h}$, directly computes in exponential time a $\mathtt{TA}_{\not\approx}$ A' that recognizes $\overline{\mathcal{L}(\mathsf{linearize}(A, \mathtt{h}))}$. Intuitively, the idea of the construction is to obtain an automaton A' whose runs compute sets of unreachable states of the linearized A. More precisely, the states of A' are sets of states of A, and the rules of A' ensure that there exists a run of A' on a term t reaching a state S if and only if, for each state q in S, there is no run of the linearized A on t reaching q. This is the typical idea for the construction of a complement automaton, but here we must also deal with the fact that (i) since we want to recognize the complement of the linearization, we must consider that the equality constraints of A are falsified whenever the involved subterms have height greater than the given $\mathtt{h}$ (even if the subterms are indeed equal), and (ii) the left-hand sides of the rules of A are not necessarily flat. For (i), it suffices that the states record the height of the recognized subterms (up to $\mathtt{h}+1$) and then consider that a rule of A is not applicable if an equality constraint has to be tested between subterms having height greater than $\mathtt{h}$. For (ii), we proceed as in Proposition 4.8 by introducing new states of the form $q_{r,p}$, where r is a rule of A and p is a position of the left-hand side of r (except λ and positions labeled by states). The goal of these new states is to be able to deal with each non-flat rule as if it was decomposed into several flat rules.

Hence, the states of A' are the pairs of the form $\langle S, h\rangle$, where h is a number in $\{0, \ldots, \mathtt{h}+1\}$ and S is a set containing states of A and new states of the form $q_{r,p}$, and the rules of A' are of the form $f(\langle S_1, h_1\rangle, \ldots, \langle S_m, h_m\rangle) \xrightarrow{D} \langle S, h\rangle$, where D is a disequality constraint constructed by negating equality atoms occurring in the rules of A. The actual definition of the set of rules of A' is quite involved. Intuitively, the rules are defined considering any possible combination of alphabet symbol f of arity m, states $\langle S_1, h_1\rangle, \ldots, \langle S_m, h_m\rangle$, and disequality constraint D, and then, for each of such combinations, considering all the possible states $\langle S, h\rangle$ that are valid as right-hand side. The notion of valid in this case can be interpreted as follows. On the one hand, the height h must be exactly one more than the maximum among the heights $h_1, \ldots, h_m$ of the left-hand side, unless one of them is already $\mathtt{h}+1$, in which case $h = \mathtt{h}+1$. On the other hand, the set S must contain states of A that are guaranteed to be unreachable with runs of the linearized A given the information provided by f and D, and inductively by $\langle S_1, h_1\rangle, \ldots, \langle S_m, h_m\rangle$. More precisely, for each term t where the rule $f(\langle S_1, h_1\rangle, \ldots, \langle S_m, h_m\rangle) \xrightarrow{D} \langle S, h\rangle$ can be applied at the root position, and for each rule r of A reaching a state occurring in S, we must guarantee that r is not applicable at the root position of t. This is equivalent to the union of the following cases:

- The alphabet symbols occurring in the left-hand side of the rule r do not match with the symbols occurring at the root position of t at the respective positions. Note that this forces the states to keep information about the symbols occurring below in some way.

- A state occurring in the left-hand side of r at a certain position p is unreachable by the linearized A on $t|_p$. Thus, we also need to keep information about the unreachable states below.

- The equality constraint of r cannot be satisfied by the linearized A, either because two subterms involved in an equality test are different, one of them does not exist, or their height is greater than $\mathtt{h}$.

If r is a flat rule with only the root position labeled by an alphabet symbol, then the previous conditions can be checked as follows. The first item is satisfied if the alphabet symbol of r does not match with the alphabet symbol f of the rule that is being generated. The second item is satisfied if one of the sets S_i occurring in the left-hand side of the constructed rule contains the state occurring in the left-hand side of r at position i. And finally, the third item is satisfied if r has an equality atom $p_1 \approx p_2$ and the recorded heights at those two positions of the generated rule are $\mathtt{h}+1$, or if D includes the negation of such atom, i.e., $p_1 \not\approx p_2$. In the case where r is not a flat rule, these conditions can be checked analogously using the states of the form $q_{r,p}$ as intermediate steps of the computation.

Definition 4.84. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{TA}_{\mathsf{hom}}$. Let $\mathtt{h}$ be a natural number. The* $\mathtt{h}$-complement *of A is the $\mathtt{TA}_{\not\approx}$ $\langle \bar{Q}, \Sigma, \bar{F}, \bar{\Delta}\rangle$ where:*

- *$\bar{Q} = 2^{Q \uplus Q'} \times \{0, \dots, \mathtt{h}+1\}$, where the set of new states Q' is defined as $\{q_{r,p} \mid r = (l \xrightarrow{c} q) \in \Delta \ \wedge \ p \in \mathsf{Pos}_\Sigma(l) \setminus \{\lambda\}\}$,*
- *$\bar{F} = \{\langle S, h\rangle \in \bar{Q} \mid F \subseteq S\}$,*
- *$\bar{\Delta}$ is the set of all rules of the form $f(\langle S_1, h_1\rangle, \dots, \langle S_m, h_m\rangle) \xrightarrow{D} \langle S, h\rangle$ satisfying the following conditions:*
    - *f is a symbol in Σ with arity m,*
    - *$\langle S_1, h_1\rangle, \dots, \langle S_m, h_m\rangle, \langle S, h\rangle \in \bar{Q}$,*
    - *$h = \min\{\mathtt{h}+1, \max\{1+h_1, \dots, 1+h_m\}\}$ where $\max \emptyset = 0$,*
    - *D is a conjunction of disequality atoms, and each $p_1 \not\approx p_2$ occurring in D satisfies that the atom $p_1 \approx p_2$ occurs in the constraint of some rule in Δ,*
    - *each state $q \in Q$ occurring in S satisfies the following condition for each rule $r \in \Delta$ of the form $f(l_1, \dots, l_m) \xrightarrow{c} q$: there exists $i \in \{1, \dots, m\}$ such that $(l_i \in Q \cap S_i) \vee (q_{r,i} \in Q' \cap S_i)$, or there exists $j \in \{1, \dots, m\}$ such that the position j occurs in c and $h_j = \mathtt{h}+1$, or there exist positions p_1, p_2 such that $p_1 \approx p_2$ occurs in c and $p_1 \not\approx p_2$ occurs in D,*
    - *each state $q_{r,p} \in Q'$ occurring in S, with r being $l \xrightarrow{c} q$ more explicitly written, satisfies the following condition: $l(p) \neq f$, or there exists $i \in \{1, \dots, \mathsf{arity}(l(p))\}$ such that $(l(p.i) \in Q \cap S_i) \vee (q_{r,p.i} \in Q' \cap S_i)$, or there exists $j \in \{1, \dots, \mathsf{arity}(l(p))\}$ such that the position $p.j$ occurs in c and $h_j = \mathtt{h}+1$.*

Example 4.85. *Consider a signature Σ consisting of a binary symbol f and a nullary symbol a. Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{TA}_{\mathsf{hom}}$, where $Q = F = \{q\}$ and $\Delta = \{a \to q,\ f(f(q,q), f(q,q)) \xrightarrow{c} q\}$ with c forcing equality between the positions 1.1, 1.2, 2.1 and 2.2. It is clear that A recognizes the language of complete trees over Σ having an even height. Now we assume a given natural number $\mathtt{h}$ and show how to construct the $\mathtt{h}$-complement $\mathtt{TA}_{\not\approx}$ $A' = \langle \bar{Q}, \Sigma, \bar{F}, \bar{\Delta}\rangle$ of A. First we focus on the set Q' of Definition 4.84. Recall that Q' contains new states whose goal is to identify positions of the left-hand sides of the rules of A that are unreachable. Only the rule $r = (f(f(q,q), f(q,q)) \xrightarrow{c} q)$ requires to introduce such new states, and we have $Q' = \{q_{r,1}, q_{r,2}\}$. It follows that the set of states $\bar{Q}$ are the pairs of the form $\langle S, h\rangle$, where*

$S \subseteq \{q, q_{r,1}, q_{r,2}\}$ and $h \in \{0, \dots, \mathtt{h}+1\}$, and that the set of final states $\bar{F}$ has the pairs $\langle S, h\rangle$ satisfying that $q \in S$. Now it only remains to define $\bar{\Delta}$. Each of the rules of $\bar{\Delta}$ has a constraint D composed of disequality atoms obtained by negating some of the equality atoms occurring in the rules of A. This implies that, in our case, D can be any conjunction of atoms of the from $\bar{p}_1 \not\approx \bar{p}_2$, where $\bar{p}_1$ and $\bar{p}_2$ are different positions in $\{1.1, 1.2, 2.1, 2.2\}$. To simplify the explanation, we denote by D any of such possible constraints. Then, the rules of $\bar{\Delta}$ have two possible forms. The first option is $a \xrightarrow{D} \langle S, 0\rangle$, where $S \subseteq \{q_{r,1}, q_{r,2}\}$. Note that the height of the recognized term is necessarily 0, and that we do not allow S to contain the state q (since runs of A on the term a can reach q). The other option is $f(\langle S_1, h_1\rangle, \langle S_2, h_2\rangle) \xrightarrow{D} \langle S_3, h_3\rangle$, where $\langle S_1, h_1\rangle$ and $\langle S_2, h_2\rangle$ are any state of $\bar{Q}$, and $\langle S_3, h_3\rangle$ satisfies the following conditions. First, h_3 is the minimum between $\mathtt{h}+1$ and $\max\{1+h_1, 1+h_2\}$. And second, S_3 only contains states of Q and Q' that are guaranteed to be unreachable given the information provided by $\langle S_1, h_1\rangle$, $\langle S_2, h_2\rangle$, and the constraint D: $q_{r,1}$ and $q_{r,2}$ are guaranteed to be unreachable if either $q \in S_1$, $q \in S_2$, $h_1 > \mathtt{h}$, or $h_2 > \mathtt{h}$, and q is guaranteed to be unreachable if either $q_{r,1} \in S_1$, $q_{r,2} \in S_2$, or D is not empty. Note how the linearization is subtly simulated in this construction by considering $q_{r,1}$ and $q_{r,2}$ unreachable when any of the recognized subterms has height greater than $\mathtt{h}$.

Lemma 4.86. *Let A be a $\mathtt{TA}_{\mathsf{hom}}$ with signature Σ, and let $\mathtt{h}$ be a natural number. Let A' be the $\mathtt{h}$-complement of A.*

Then, $\mathcal{L}(A') = \overline{\mathcal{L}(\mathsf{linearize}(A, \mathtt{h}))}$. Moreover, A' can be computed with time and space in $2^{\mathcal{O}((|A|+\log(\mathtt{h}))\cdot\mathsf{maxar}(\Sigma)+\log(|\Sigma|))}$, and such that the following bounds hold:

- $\mathsf{h}_{\not\approx}(A') = \mathsf{h}_{\approx}(A)$,
- $\mathsf{n}_{\not\approx}(A') = \mathsf{n}_{\approx}(A)$.

Proof (Sketch). The fact $\mathcal{L}(A') = \overline{\mathcal{L}(\mathsf{linearize}(A, \mathtt{h}))}$ can be proved by induction on the height of the recognized terms. However, since the proof is quite technical but not conceptually difficult, we just give the needed intuition. Consider a run r of A' on a term t reaching a state $\langle S, h\rangle$. Recall that the goal of S is to compute states of A that cannot be reached by $\mathsf{linearize}(A, \mathtt{h})$ when recognizing t. In order to reason about the definition of S, consider a state $q \in S$ and a rule $l \xrightarrow{c} q$ of A, and note that we must guarantee that such rule cannot be applied at the root position of t. This condition is equivalent to the union of the following particular cases:

- The alphabet symbols occurring in l do not match with the symbols occurring in t at the respective positions.
- For a position $p \in \mathsf{Pos}(l)$ labeled by a state q of A, the state q cannot be reached by $\mathsf{linearize}(A, \mathtt{h})$ when recognizing $t|_p$.
- The constraint c is unsatisfiable since it contains an atom $p_1 \approx p_2$ and either the subterms $t|_{p_1}$ and $t|_{p_2}$ do not exist, or are different, or their height is greater than $\mathtt{h}$.

If each rule of A with right-hand side q satisfies any of the previous conditions, then q cannot be reached by $\mathsf{linearize}(A, \mathtt{h})$ when recognizing t. Note that the definition of the rules of A' checks precisely these conditions, and that the only difficulty stems from the fact that the left-hand sides of the rules of A are not necessarily flat. The $\mathtt{h}$-complement construction deals with the non-flat rules by introducing new states that identify each of the positions of their left-hand sides (except λ and the positions labeled by states), and then using these new states to propagate upwards the relevant information.

The previous justifications are enough to see that the definition of A' is sound. In order to see that it is also complete, note that the states $\langle S, h\rangle$ of A' are constructed considering any possible combination of S and h, and moreover, for the rules of A' any possible left-hand side $f(\langle S_1, h_1\rangle, \ldots, \langle S_m, h_m\rangle)$ and disequality constraint D is considered. This is enough to guarantee that, among all the possible runs of A' on a term t, there are runs computing maximal sets of unreachable states. This maximality implies completeness.

It only remains to analyse the cost of the construction. Note that the number k of states of A' is bounded by $2^{|A|} \cdot (\mathtt{h}+2)$. Moreover, note that the number of different rules of A' is at most $k^{\mathsf{maxar}(\Sigma)+1} \cdot |\Sigma| \cdot 2^{\mathsf{n}_{\approx}(A)}$, and that the size of each one of them is bounded by $\mathsf{maxar}(\Sigma) + 2 + |A|$, where $|A|$ bounds the size of the disequality constraint. Hence, $|A'|$ is in $2^{\mathcal{O}((|A|+\log(\mathtt{h}))\cdot \mathsf{maxar}(\Sigma)+\log(|\Sigma|)+\mathsf{n}_{\approx}(A))}$. It is easy to see that the cost of computing the transformation is polynomial with respect to $|A'|$, and that $\mathsf{h}_{\not\approx}(A') = \mathsf{h}_{\approx}(A)$ and $\mathsf{n}_{\not\approx}(A') = \mathsf{n}_{\approx}(A)$, and hence, the statement holds. ■

As a technical detail, in order to apply Proposition 4.9 to the $\mathtt{h}$-complement of a $\mathtt{TA}_{\mathsf{hom}}$, we also need to show that it admits deterministic accepting runs (see Definition 4.2 for the notion of determinism). The proof is not technically difficult, but we include it to provide more intuition on the transformation.

Lemma 4.87. *Let A be a $\mathtt{TA}_{\mathsf{hom}}$, and let $\mathtt{h}$ be a natural number. Let A' be the $\mathtt{h}$-complement of A.*

Then, A' admits deterministic accepting runs.

Proof. We first note that, for any term t, there exists a run r of A' on t that computes maximal sets of unreachable states. In other words, an r satisfying that, for each position $p \in \mathsf{Pos}(t)$, the subrun $r|_p$ reaches the state of the form $\langle S, h\rangle$ where $h = \min\{\mathtt{h}+1, \mathsf{height}(t|_p)\}$ and S is such that there exists no run of A' on $t|_p$ reaching a state of the form $\langle S', h\rangle$ with $S' \not\subseteq S$. We call such r a maximal run of A' on t. The previous fact is not enough, since a maximal run needs not be deterministic: this is because although two occurrences of the same subterm are evaluated to the same state in a maximal run, it might happen that this state is reached using rules whose constraint is different. However, it is easy to see that, among all the maximal runs of A' on t, there necessarily exists a maximal run r satisfying $r(p_1) = r(p_2)$ if $t|_{p_1} = t|_{p_2}$, for each two positions $p_1, p_2 \in \mathsf{Pos}(t)$. Such r is necessarily deterministic. In order to conclude, we just note that any term in $\mathcal{L}(A')$ admits a deterministic maximal run of A' on it, and such run is necessarily accepting by the definition of the final states of an $\mathtt{h}$-complement. ■

Note that Lemmas 4.86 and 4.87, and Proposition 4.9 guarantee that we can compute a $\mathtt{TA}_{\mathsf{hom},\not\approx}$ recognizing $\mathcal{L}(A) \cap \overline{\mathcal{L}(\mathsf{linearize}(A, \check{h}))}$ with size exponentially bounded by $|A|$, even though $\check{h}$ itself is already exponential with respect to $|A|$. We can finally state the complexity of deciding HOM.

Theorem 4.88. *The HOM problem is EXPTIME-complete.*

Proof. Assume a given $\mathtt{TA}$ A and a tree homomorphism H. By Propositions 4.4 and 4.9 and Lemmas 4.6, 4.86, and 4.87, a $\mathtt{TA}_{\mathsf{hom},\not\approx}$ A' such that $\mathcal{L}(A')$ is empty if and only if $H(\mathcal{L}(A))$ is regular can be computed in exponential time with respect to the size of A and H. Moreover, the bounds on $\mathsf{h}_{\not\approx}(A')$, $\mathsf{n}_{\not\approx}(A')$, $|\mathsf{Pos}_{\mathrm{lhs}}(A')|$, and $|A'|$ are such that, by Corollary 4.75, emptiness of $\mathcal{L}(A')$ can be decided in exponential time with respect to the size of A and H. Thus, we conclude by noting that the problem is EXPTIME-hard by Proposition 4.1. ■

Chapter 5

Decidability of global reflexive disequality constraints

In this chapter we focus on automata with global disequality constraints. In this setting, the constraints that must be satisfied by the runs are associated with the automaton itself and not with its transitions. The atomic predicates occurring in such constraints are disequality expressions of the form $q_1 \not\approx q_2$, for states q_1 and q_2 of the automaton, and are satisfied by a run r on a term t if the following property holds: for each two distinct positions p_1 and p_2 of t such that the subrun $r|_{p_1}$ reaches q_1 and the subrun $r|_{p_2}$ reaches q_2, the subterms $t|_{p_1}$ and $t|_{p_2}$ are different. We tackle the emptiness and finiteness problems for the subclass $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ of tree automata with global *reflexive* disequality constraints. More precisely, this subclass is defined by imposing the following additional requirements on the form of the global disequality constraint: (i) the global constraint is a conjunction of positive atoms, instead of an arbitrary Boolean expression, and (ii) the global constraint defines a reflexive relation on the states occurring in it, that is, whenever the constraint has an atom of the form $q_1 \not\approx q_2$, then it also contains the reflexive atoms $q_1 \not\approx q_1$ and $q_2 \not\approx q_2$. We obtain triple exponential time algorithms for both decision problems. The outline of the proof for emptiness proceeds as follows. In a first step, a transformation of the given $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ is defined in order to obtain a simpler setting for the remaining reasonings. Then, an inference system is used to construct runs on this new setting. Such inference system guarantees along the construction of the runs that the global disequality constraint is satisfied. Finally, we show that the inference takes triple exponential time, and that it constructs an accepting run if and only if the language recognized by the given $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ is not empty. Finiteness of the recognized language is tackled similarly. In particular, we prove that the number of inference steps performed to construct a run has a direct correspondence with the height of such run. Hence, if we detect that it is possible to pump some of the inference steps performed while constructing an accepting run, then we can conclude that there are arbitrarily high terms in the language.

The organization of this chapter is as follows. In Section 5.1 we give a general definition for tree automata with global (dis)equality constraints, introducing a uniform notation to easily refer to variants on the form of the global constraint. We also summarize how those variants compare with each other with respect to expressiveness, giving especial emphasis to the expressive power of classes closely related to the model $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ studied in this chapter. In Section 5.2 we give an overview of the original insight that led to the proof for the decidability of the emptiness and finiteness problems for $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$. In Sections 5.3 and 5.4 we analyse properties of $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ and define the transformation that simplifies the setting, respectively, and in Section 5.5 we present the inference system and tackle the decision of emptiness and finiteness for $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$. Finally, in Section 5.6 we show how to adapt the previous decision results to automata with global reflexive disequality constraints that run on unranked ordered terms.

5.1 TA with global constraints

We define a class of tree automata with global constraints that generalizes the original notion of automata with global constraints TAGED from [FTT08, FTT10]. Our definition is parametric in order to easily restrict which type of atoms may occur in the global constraint. Intuitively, with the constraint type $\approx$ as parameter we allow atoms of the form $q_1 \approx q_2$, and with the constraint type $\not\approx$ we allow atoms of the form $q_1 \not\approx q_2$. To obtain a unified notation for the automaton classes studied here and in the literature, we also define several particularizations of those two constraint types.

Definition 5.1. *A* tree automaton with global constraints *over the constraint types* $\tau_1, \dots, \tau_n$ *is denoted* $\mathtt{TAG}_{\tau_1,\dots,\tau_n}$ *and defined as a tuple* $A = \langle Q, \Sigma, F, \Delta, C\rangle$*, where* $\langle Q, \Sigma, F, \Delta\rangle$ *is a* TA*, denoted* $\mathtt{ta}(A)$*, and* C*, called the* (global) constraint*, is a Boolean combination of atomic constraints of types* $\tau_1, \dots, \tau_n$*. A* $\mathtt{TAG}_{\tau_1,\dots,\tau_n}$ *is called* positive conjunctive*, denoted* $\mathtt{TAG}^{\wedge}_{\tau_1,\dots,\tau_n}$*, when its global constraint is a conjunction of atomic constraints. For the fragment* $\mathtt{TAG}^{\wedge}_{\tau_1,\dots,\tau_n}$*, the global constraint* C *is indistinguishably treated as a set of atoms, and moreover, we denote that a state* q *is involved in some atom of* C *as* $q \in C$*, and say that* q *is a* constrained state*.*

We define the constraint types $\approx$ *and* $\not\approx$ *as follows: an atom of type* $\approx$ *(respectively,* $\not\approx$*) is an unordered pair of the form* $q_1 \approx q_2$ *(respectively,* $q_1 \not\approx q_2$*), where* $q_1, q_2 \in Q$*. Additionally, for the positive conjunctive fragment of automata with global constraints we introduce several particular cases of both constraint types. These new types are defined by restricting which kind of relation may be induced on the states occurring in the global constraint:*

- *The type* $\approx_{\mathcal{I}}$ *is the particular case of* $\approx$ *where the atoms of type* $\approx$ *only relate identical states, that is: the constraint has no atom of the form* $q_1 \approx q_2$ *where* q_1 *and* q_2 *are distinct states. Analogously for the type* $\not\approx_{\mathcal{I}}$*.*

- *The type* $\approx_{\mathcal{R}}$ *is the particular case of* $\approx$ *where the induced relation on the states occurring in atoms of type* $\approx$ *is reflexive, that is: whenever the constraint has an atom of the form* $q_1 \approx q_2$*, then it also contains the reflexive atoms* $q_1 \approx q_1$ *and* $q_2 \approx q_2$*. Analogously for the type* $\not\approx_{\mathcal{R}}$*.*

- *The type $\approx_{\mathcal{A}}$ is the particular case of $\approx$ where the induced relation on the states occurring in atoms of type $\approx$ is anti-reflexive, that is: the constraint has no atom of the form $q \approx q$. Analogously for the type $\not\approx_{\mathcal{A}}$.*

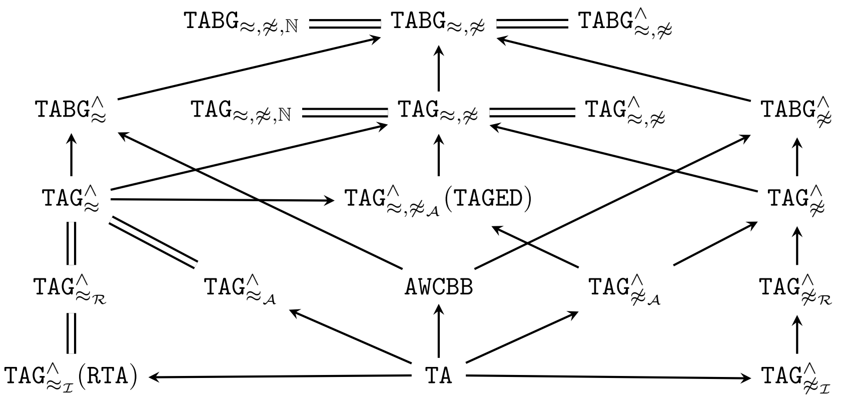
A run *of A on a term $t \in \mathcal{T}(\Sigma)$ is a run of $\mathtt{ta}(A)$ on t satisfying the global constraint C, where the satisfiability of Boolean expressions is as usual, and the satisfiability of the atomic predicates of types $\approx$ and $\not\approx$ is defined as follows: an atom of the form $q_1 \approx q_2$ (respectively, $q_1 \not\approx q_2$) is satisfied if and only if, for each distinct $p_1, p_2 \in \mathsf{Pos}(t)$ such that the right-hand sides of the rules $r(p_1)$ and $r(p_2)$ are q_1 and q_2, respectively, $t|_{p_1} = t|_{p_2}$ (respectively, $t|_{p_1} \neq t|_{p_2}$). We adapt the usual definitions on runs:* $\mathsf{term}(r) = t$*,* $\mathsf{Pos}(r) = \mathsf{Pos}(t)$*,* $\mathsf{height}(r) = \mathsf{height}(t)$*, and the notion of* subrun $r|_p$ *and* replacement $r[r']_p$ *for $p \in \mathsf{Pos}(r)$ and some other run r' of A. The* state reached *by r is the right-hand side of $r(\lambda)$. The run r is called* accepting *if it reaches a state in F. A term t is* accepted/recognized *by A if there exists an accepting run of A on t. The* language recognized *by A, denoted $\mathcal{L}(A)$, is the set of terms accepted by A. By $\mathcal{L}(A, q)$ we denote the set of terms for which there exists a run of A on them reaching q.*

We remark that a constraint of the form $\neg(q_1 \approx q_2)$ is not equivalent to $q_1 \not\approx q_2$ since a universal quantifier is involved in the interpretation of the atoms. Similarly for $\neg(q_1 \not\approx q_2)$ and $q_1 \approx q_2$.

It is easy to see that the class RTA from [JKV11] is equivalent to $\mathtt{TAG}^{\wedge}_{\approx_{\mathcal{I}}}$, and that the class TAGED from [FTT10] is equivalent to $\mathtt{TAG}^{\wedge}_{\approx,\not\approx_{\mathcal{A}}}$. We informally generalize Definition 5.1 in order to capture the automaton models studied in [Vac10] and [BCG$^+$13]. First, we adopt from [Vac10] the constraint type $\mathbb{N}$, which allows to impose restrictions on the number $|q|$ of occurrences of a given state q in a run, or the number $\|q\|$ of distinct subterms reaching a given state q in a run. The atoms of type $\mathbb{N}$ are expressions of the form $a_1|q_1| + \ldots + a_n|q_n| \otimes k$ or of the form $a_1\|q_1\| + \ldots + a_n\|q_n\| \otimes k$, where $\otimes$ is any operator in $\{\geq, \leq, =\}$ and $a_1, \ldots, a_n, k$ are natural numbers, with straightforward interpretations. In [BCG$^+$13], a class $\mathtt{TABG}_{\approx,\not\approx,\mathbb{N}}$ is introduced merging the model with global constraints $\mathtt{TAG}_{\approx,\not\approx,\mathbb{N}}$ and the model with local constraints AWCBB. Moreover, all constraints in $\mathtt{TABG}_{\approx,\not\approx,\mathbb{N}}$ are interpreted modulo a flat equational theory: local and global (dis)equality tests are performed modulo the given theory, and the number $\|q\|$ is reinterpreted to be the number of distinct equivalent classes (modulo the given theory) of subterms reaching q in a run. For a formal definition of flat equational theories, see [BCG$^+$13].

5.1.1 Analysis of the expressive power

Here we summarize how the distinct classes of automata with global constraints relate to each other in terms of expressiveness (see Figure 5.2). We start with known results from the literature. First, in [Vac10] it is shown effective equivalence between the classes $\mathtt{TAG}_{\approx,\not\approx,\mathbb{N}}$ and $\mathtt{TAG}^{\wedge}_{\approx,\not\approx}$. In [BCG$^+$13], a reworked proof is proposed for effective equivalence between $\mathtt{TABG}_{\approx,\not\approx,\mathbb{N}}$ and its positive conjunctive fragment $\mathtt{TABG}^{\wedge}_{\approx,\not\approx}$. The constructions used in this latter proof easily allow to conclude effective equivalence between the subclass of $\mathtt{TAG}_{\approx,\not\approx,\mathbb{N}}$ where the global constraint has no atom of the form $q \not\approx q$ and $\mathtt{TAG}^{\wedge}_{\approx,\not\approx_{\mathcal{A}}}$ (i.e., TAGED), which in turn is proved in [FTT10] to be effectively



$$\begin{tikzcd}
 & \mathtt{TABG}_{\approx,\not\approx,\mathbb{N}} \arrow[r, equal] & \mathtt{TABG}_{\approx,\not\approx} \arrow[r, equal] & \mathtt{TABG}^{\wedge}_{\approx,\not\approx} & \\
\mathtt{TABG}^{\wedge}_{\approx} \arrow[urr] & \mathtt{TAG}_{\approx,\not\approx,\mathbb{N}} \arrow[r, equal] & \mathtt{TAG}_{\approx,\not\approx} \arrow[u] \arrow[r, equal] & \mathtt{TAG}^{\wedge}_{\approx,\not\approx} & \mathtt{TABG}^{\wedge}_{\not\approx} \arrow[ull] \\
\mathtt{TAG}^{\wedge}_{\approx} \arrow[u] \arrow[urr] \arrow[rr] \arrow[d, equal] \arrow[dr, equal] & & \mathtt{TAG}^{\wedge}_{\approx,\not\approx_{\mathcal{A}}}(\mathtt{TAGED}) \arrow[u] & & \mathtt{TAG}^{\wedge}_{\not\approx} \arrow[u] \arrow[ull] \\
\mathtt{TAG}^{\wedge}_{\approx_{\mathcal{R}}} \arrow[d, equal] & \mathtt{TAG}^{\wedge}_{\approx_{\mathcal{A}}} & \mathtt{AWCBB} \arrow[uull] \arrow[uurr] & \mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{A}}} \arrow[ul] \arrow[ur] & \mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}} \arrow[u] \\
\mathtt{TAG}^{\wedge}_{\approx_{\mathcal{I}}}(\mathtt{RTA}) & & \mathtt{TA} \arrow[ll] \arrow[ul] \arrow[u] \arrow[ur] \arrow[rr] & & \mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{I}}} \arrow[u]
\end{tikzcd}$$

Figure 5.2: Classes of tree automata with local and global constraints. Effective strict inclusion is denoted $\longrightarrow$, effective equivalence is denoted $=\!=$, otherwise the classes are incomparable.

equivalent to $\mathtt{TAG}^{\wedge}_{\approx_{\mathcal{I}},\not\approx_{\mathcal{A}}}$ by means of an exponential time transformation (these two relations are not depicted in Figure 5.2 to keep its presentation simpler). Such exponential time transformation also leads in [FTT10] to the effective equivalence between the classes $\mathtt{TAG}^{\wedge}_{\approx}$, $\mathtt{TAG}^{\wedge}_{\approx_{\mathcal{R}}}$, and $\mathtt{TAG}^{\wedge}_{\approx_{\mathcal{I}}}$ (i.e., $\mathtt{RTA}$). To prove that these classes are also equivalent to $\mathtt{TAG}^{\wedge}_{\approx_{\mathcal{A}}}$ as depicted in Figure 5.2, it suffices to transform $\mathtt{TAG}^{\wedge}_{\approx_{\mathcal{I}}}$ into $\mathtt{TAG}^{\wedge}_{\approx_{\mathcal{A}}}$. Since this is out of scope and rather straightforward, we just do it intuitively: for each atom of the form $q \approx q$ in the given $\mathtt{TAG}^{\wedge}_{\approx_{\mathcal{I}}}$, we generate a synonym q' of q and replace the atom by $q \approx q'$, and additionally, force accepting runs to have either (i) no occurrences of q and q', (ii) exactly one occurrence of q and none of q', (iii) exactly one occurrence of q' and none of q, or (iv) some occurrences of both q and q' (this can be done by enriching the states to count, up to 2, the number of occurrences of q and q' in a run). Strict inclusion of $\mathtt{TAG}^{\wedge}_{\approx,\not\approx_{\mathcal{A}}}$ (i.e., $\mathtt{TAGED}$) in $\mathtt{TAG}_{\approx,\not\approx}$ is proved in [Vac10]. From the constructions done in that proof it can also be concluded that $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{I}}}$ is incomparable with $\mathtt{TAG}^{\wedge}_{\approx,\not\approx_{\mathcal{A}}}$ (i.e., $\mathtt{TAGED}$). Finally, the incomparability between the automaton models with global constraints and $\mathtt{AWCBB}$ is also tackled in [Vac10].

We now characterize the expressiveness of $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ by comparing it with two close variants of the disequality constraint: $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{A}}}$ and $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{I}}}$. In Lemma 5.3 we conclude that the classes of languages recognizable by $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ and $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{I}}}$ are incomparable with the class of languages recognizable by $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{A}}}$. In Lemma 5.4 we show that $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ are strictly more expressive than $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{I}}}$. To ease the presentation, we denote terms of the form $f(t_1, f(t_2, f(t_3, \ldots f(t_{n-1}, t_n)\ldots)))$ as $f_{[t_1,t_2,t_3,\ldots,t_{n-1},t_n]}$, where $n \geq 2$ and f is a binary symbol. Note that, for $i \in \{1, \ldots, n-1\}$, the term t_i occurs in such term at the position $\overbrace{2\ldots2}^{i-1}.1$, whereas t_n occurs at the position $\overbrace{2\ldots2}^{n-1}$. In order to use a uniform notation in the arguments, we will reason on the first $n-1$ terms t_i, and ignore the last term t_n.

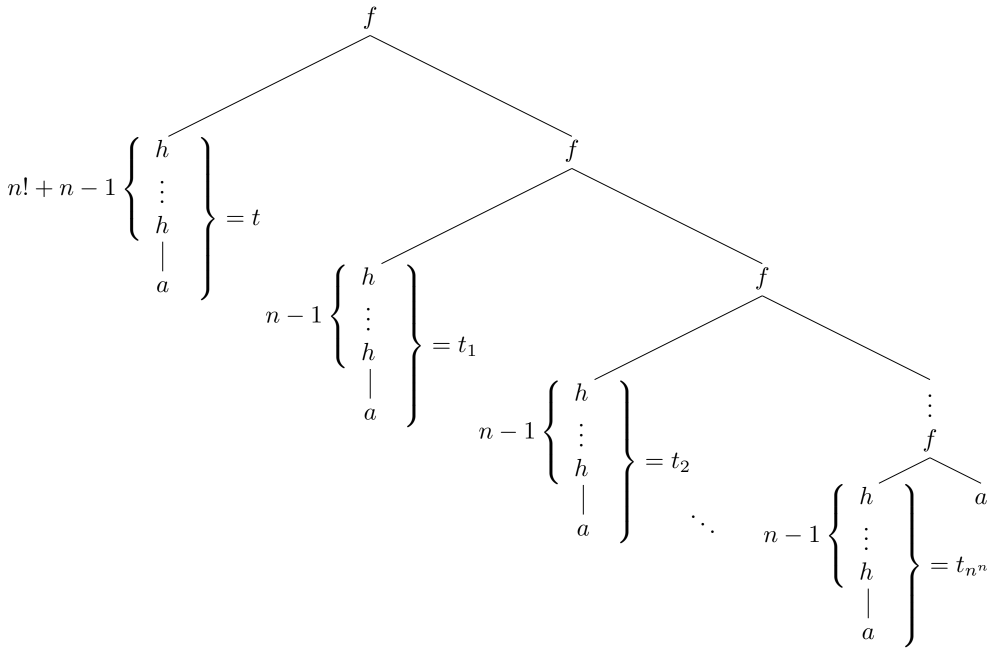
Lemma 5.3. *The class of languages recognizable by* $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{A}}}$ *is incomparable with the classes of languages recognizable by* $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ *and* $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{I}}}$ *with respect to inclusion.*

Proof. We first show that the class of languages recognizable by $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{A}}}$ does not include the classes of languages recognizable by $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ and $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{I}}}$. This can be directly concluded from [Vac10] since, as stated before, the arguments there for proving strict inclusion of $\mathtt{TAG}^{\wedge}_{\approx,\not\approx_{\mathcal{A}}}$ (i.e., $\mathtt{TAGED}$) in $\mathtt{TAG}_{\approx,\not\approx}$ also allow to conclude that $\mathtt{TAG}^{\wedge}_{\approx,\not\approx_{\mathcal{A}}}$ (i.e., $\mathtt{TAGED}$) and $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{I}}}$ are incomparable.

We now show that the class of languages recognizable by $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{A}}}$ is not included in the class of languages recognizable by $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$. Since $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{I}}}$ is a particular case of $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$, this claim holds also for $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{I}}}$. Consider the following language over the signature $\Sigma := \{a{:}0,\ h{:}1,\ f{:}2\}$:

$$L := \{f_{[h^k(a),h^{k_1}(a),\dots,h^{k_n}(a),a]} \mid n \geq 1 \ \wedge\ k,k_1,\dots,k_n \geq 0 \ \wedge\ k \neq k_1,\dots,k_n\}$$

It is straightforward that L can be recognized by a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{A}}}$. We proceed by contradiction assuming that there exists a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ A recognizing L. Let n be a natural number strictly greater than the number of states of A, and consider the following term in L of the form $f_{[t,t_1,t_2,\dots,t_{n^n},a]}$:



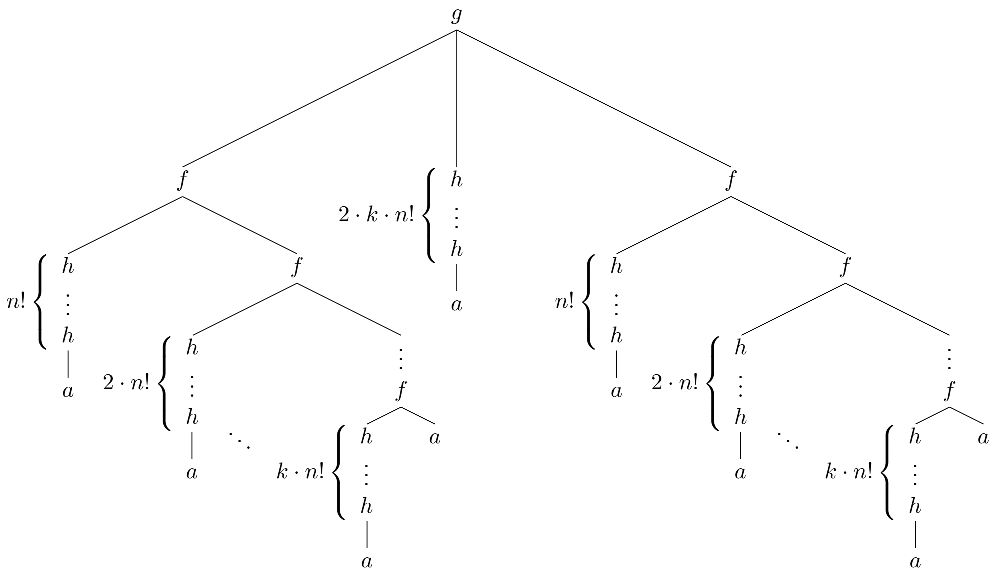
By the assumption, there exists an accepting run r of A on such term. Since n is greater than the number of states of A, and n^n is greater than the number of different sequences of states of A of length n, there exist two distinct $i,j \geq 1$ such that the positions $p_i := \overbrace{2\dots2}^{i}.1$ and $p_j := \overbrace{2\dots2}^{j}.1$ are defined in r and the subruns $r|_{p_i}$ and $r|_{p_j}$ are identical. Since the relation defined by the global constraint of A is reflexive, the identical subruns $r|_{p_i}$ and $r|_{p_j}$ cannot contain constrained states. Moreover, since $\mathsf{height}(r|_{p_i}) = n-1$ and n is greater than the number of states of A, the run $r|_{p_i}$ can be pumped to obtain another run r' of A satisfying the following conditions: the term recognized by r' is $h^{n!+n-1}(a) = t$, the states reached by r' and $r|_{p_i}$ coincide, and r' does not contain constrained states. It is easy to see that $r[r']_{p_i}$ satisfies the global constraint of A, and that it is an accepting run of A on the term $f_{[t,t_1,\ldots,t_{i-1},t,t_{i+1},\ldots,t_{n^n}]}$, which is not in L, a contradiction. ■

Lemma 5.4. *The class of languages recognizable by* $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{I}}$ *is strictly included in the class of languages recognizable by* $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$.

Proof. Since $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{I}}$ is the particular case of $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$ where only atoms of the form $q \not\approx q$ are allowed, the inclusion holds and it suffices to prove that it is strict. Consider the following language over the signature $\Sigma := \{a{:}0,\ h{:}1,\ f{:}2,\ g{:}3\}$:

$$\begin{aligned} L := \{g(f_{[h^{\alpha_1}(a),\ldots,h^{\alpha_n}(a),a]}, h^{\beta}(a), f_{[h^{\gamma_1}(a),\ldots,h^{\gamma_m}(a),a]}) \mid\ & n,m \geq 1,\\ & \alpha_1,\ldots,\alpha_n,\beta,\gamma_1,\ldots,\gamma_m \geq 0,\\ & \forall_{1\leq i<j\leq n} : \alpha_i \neq \alpha_j,\\ & \forall_{1\leq i<j\leq m} : \gamma_i \neq \gamma_j,\\ & \beta \neq \alpha_1,\ldots,\alpha_n,\gamma_1,\ldots,\gamma_m\} \end{aligned}$$

It is straightforward that L can be recognized by a $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$. We proceed by contradiction assuming that there exists a $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{I}}$ A recognizing L. Let n be a natural number strictly greater than the number of states of A, and consider the following term $t \in L$, where $k = 2 \cdot n \cdot (n+1)$:



By the assumption, there exists an accepting run r of A on t. We now define all the subruns of $r|_1$ that recognize subterms of $t|_1$ of the form $h(h(\ldots h(a)\ldots))$. Moreover, we want to refer to them by their height. Hence, let $r_{i,d}$ be the subrun of $r|_1$ at position

$\overbrace{2\ldots2}^{i-1}.1.\overbrace{1\ldots1}^{i\cdot n!-d}$, for all $i \in \{1,\ldots,k\}$ and $d \in \{0,\ldots,i\cdot n!\}$. Note that $\mathsf{height}(r_{i,d}) = d$ for all $i \in \{1,\ldots,k\}$ and $d \in \{0,\ldots,i\cdot n!\}$. Consider two different runs $r_{i,d}$ and $r_{j,d}$, and observe that $r_{i,d}$ and $r_{j,d}$ cannot reach the same state q of A if q is a constrained state, because the atom $q \not\approx q$ of the global constraint of A would be falsified since $\mathsf{term}(r_{i,d}) = \mathsf{term}(r_{j,d}) = h^d(a)$. It follows that, for a fixed d, the number of $r_{i,d}$'s reaching some constrained state is smaller than n. Hence, for a fixed d, the number of $r_{i,d}$'s containing some constrained state is smaller than $n\cdot(d+1)$. Let r_i be $r_{i,i\cdot n!}$, for all $i \in \{1,\ldots,k\}$. Note that those are the subruns of $r|_1$ at positions $\overbrace{2\ldots2}^{i-1}.1$. The number of r_i's whose subrun of height n contains a constrained state is smaller than $n\cdot(n+1)$. Let $\bar{r}_{i,d}$ be the subruns of $r|_3$ defined analogously to the definition of $r_{i,d}$ as subruns of $r|_1$, and let $\bar{r}_i$ be $\bar{r}_{i,i\cdot n!}$. By an analogous argument, the number of $\bar{r}_i$'s whose subrun of height n contains a constrained state is smaller than $n\cdot(n+1)$. By definition of k, it follows that there exists $e \in \{1,\ldots,k\}$ such that r_e and $\bar{r}_e$ satisfy that their subruns of height n do not contain any constrained state. Moreover, since $\mathsf{term}(r_e) = \mathsf{term}(\bar{r}_e) = h^{e\cdot n!}(a)$, it follows that $r_{e,d}$ and $\bar{r}_{e,d}$ cannot reach the same state q of A if q is a constrained state, for all $d \in \{0,\ldots,e\cdot n!\}$.

Note that the subruns of r_e and $\bar{r}_e$ of height n, which do not have constrained states, can be pumped to transform r_e and $\bar{r}_e$ into new runs r_e^i and $\bar{r}_e^i$, for $i \geq 0$, satisfying the following conditions:

(a) $\mathsf{term}(r_e^i) = \mathsf{term}(\bar{r}_e^i) = h^{(i+e)\cdot n!}(a)$, i.e., we can obtain runs on terms of the form $h(h(\ldots h(a)\ldots))$ with a height greater than $e\cdot n!$ and multiple of $n!$.

(b) For each position $p \in \mathsf{Pos}(r_e^i)$ such that $r_e^i|_p$ reaches a constrained state, it holds that $p \in \mathsf{Pos}(r_e)$ and $r_e|_p$ reaches a constrained state. Analogous for $\bar{r}_e^i$ and $\bar{r}_e$.

(c) For each position $p \in \mathsf{Pos}(r_e^i)$ such that $r_e^i|_p$ and $\bar{r}_e^i|_p$ reach constrained states, it holds that such states differ.

A particular consequence of condition (b) is that, for all $i \geq 0$, any position p such that $r_e^i|_p$ or $\bar{r}_e^i|_p$ reaches a constrained state necessarily satisfies $|p| < e\cdot n! - n$. By condition (a), for each $i \geq 2\cdot k - e$, the runs r_e^i and $\bar{r}_e^i$ satisfy that $\mathsf{height}(r_e^i) = \mathsf{height}(\bar{r}_e^i) \geq 2\cdot k\cdot n!$. For any such i, the height of any subrun of r_e^i or $\bar{r}_e^i$ reaching a constrained state is greater than $2\cdot k\cdot n! - (e\cdot n! - n) = (2\cdot k - e)\cdot n! + n \geq k\cdot n! + n$. Hence, since $k\cdot n!$ is the maximum height of $r_1,\ldots,r_k,\bar{r}_1,\ldots,\bar{r}_k$, constraints between subruns of $r_e^i, \bar{r}_e^i$ and of $r_1,\ldots,r_k,\bar{r}_1,\ldots,\bar{r}_k$ are satisfied. Therefore, the subrun $r|_2$ and the subrun r_e must share a constrained state at the same respective position, since otherwise, we can replace r_e by $r_e^{2\cdot k-e}$ and obtain an accepting run on a term not in L. The same applies to $\bar{r}_e$.

Let p_e be the position $\overbrace{2\ldots2}^{e-1}.1$, i.e., the position where r_e occurs in $r|_1$ and $\bar{r}_e$ occurs in $r|_3$. Let $p, \bar{p}$ be the shortest positions in $\mathsf{Pos}(r|_2)$ such that $r_e|_p$ and $r|_{2.p}$ reach the same constrained state and $\bar{r}_e|_{\bar{p}}$ and $r|_{2.\bar{p}}$ reach the same constrained state. By condition (c), it follows that $p \neq \bar{p}$. Assume without loss of generality that $p < \bar{p}$. We want to pump r_e and $\bar{r}_e$ to obtain runs of height $3\cdot k\cdot n!$, and swap the subruns of r at positions $1.p_e.p$ and $2.p$ to obtain an accepting run r' on a term not in L. First, let r'_e be $r_e^{3\cdot k-e}$, and let $\bar{r}'_e$ be $\bar{r}_e^{3\cdot k-e}$. Note that $\mathsf{height}(r'_e) = \mathsf{height}(\bar{r}'_e) = 3 \cdot k \cdot n!$ and, by the same arguments as before, the height of any subrun of r'_e or $\bar{r}'_e$ reaching a constrained state is greater than $3\cdot k\cdot n! - (e\cdot n! - n) = (3\cdot k - e)\cdot n! + n \geq 2\cdot k\cdot n! + n$. Hence, since $2 \cdot k \cdot n!$ is the maximum height of $r|_2, r_1, \ldots, r_k, \bar{r}_1, \ldots, \bar{r}_k$, constraints between subruns of $r'_e, \bar{r}'_e$ and of $r|_2, r_1, \ldots, r_k, \bar{r}_1, \ldots, \bar{r}_k$ are satisfied. Second, let r' be the replacement $r[r|_{2.p}]_{1.p_e.p}[r'_e|_p]_{2.p}[\bar{r}'_e]_{3.p_e}$. Note that $\mathsf{term}(r') \notin L$ since $\mathsf{term}(r')|_2 = \mathsf{term}(r')|_{3.p_e} = h^{3\cdot k\cdot n!}(a)$. By condition (c) in the definition of r_e^i and $\bar{r}_e^i$ and the assumption $p < \bar{p}$, constraints between the subruns $r'|_2$ and $r'|_{3.p_e}$ are satisfied. Finally, constraints between $r'|_{1.p_e}$, which has height equal to $2 \cdot k \cdot n!$, and any other subrun of r' are also satisfied by a height argument. Hence, r' is an accepting run, a contradiction. ■

5.2 Intuition on the approach

Consider a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ A. Clearly, the main difficulty to decide emptiness and finiteness of $\mathcal{L}(A)$ is related to the global disequality constraint and how it affects the language recognized by each of the states of A. Hence, an obvious first step is trying to identify which are the states where the decision of emptiness and finiteness can be done in a simple manner. To this end, we look for the states q of A satisfying the following property: there is a run r of A reaching q, the run r can be pumped without introducing new occurrences of constrained states, and moreover, every subrun of r reaching a constrained state can be pumped in such way. Note that this property means that r can be made arbitrarily high, and moreover, each of its subruns reaching a constrained state can also be made arbitrarily high. We denote as Q_A^∞ the set of states satisfying such property, and prove that it can be efficiently computed. Our interest in Q_A^∞ is twofold: on the one hand, the emptiness and finiteness decision for each state in Q_A^∞ is trivial since its recognized language is infinite by definition, and on the other hand, the states in Q_A^∞ can be exploited to ease the construction of runs of A. In order to illustrate how the construction of a run can be simplified thanks to the information provided by Q_A^∞, consider a run r of $\mathtt{ta}(A)$ that is not a valid run of A since the global constraint is falsified. Moreover, assume that the constraint is falsified due to a subrun of r that contains an occurrence of a state $q \in Q_A^\infty$. Under these assumptions, note that r could be modified to obtain a valid run r' of A as follows: it suffices to fix the conflicting subrun of r by replacing its subrun reaching q by another one, chosen among the infinitely many available runs reaching q whose constrained subruns are all higher than r itself. Hence, during the construction of a run of A it is possible to ignore subruns that reach states in Q_A^∞, since any possible conflict they have can be fixed.

At this point, it only remains to focus on the states that are not in Q_A^∞. Hence, consider a state $q \notin Q_A^\infty$, and assume that it occurs in the global constraint. We prove the following property: the runs of A reaching q where there is exactly one occurrence of a constrained state, i.e., the q at the root position, necessarily satisfy that their height is linearly bounded by the size of A (otherwise, it would be possible to perform pumpings on some of those runs, concluding that $q \in Q_A^\infty$). Moreover, since $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ ensure that the global disequality constraint is a reflexive relation, the previous property leads to a proof for the following crucial insight: there is a bound for the number of subruns within any run that reach a constrained state not in Q_A^∞ and that have no other occurrence of constrained states (otherwise, there would be two of these subruns reaching the same constrained state and recognizing the same term, thus falsifying the reflexive disequality constraint). A direct consequence of this fact is that we are able to bound the number of parallel positions in a run where the pending subruns contain constrained states not in Q_A^∞.

Using the previous ideas, we propose an inference rule $\mathfrak{R}$ that non-deterministically tries to construct accepting runs, starting from their root position and expanding them towards the leaves at each inference step. We ensure termination of this process in triple exponential time by (i) leaving the subruns reaching a state in Q_A^∞ unexpanded since they are irrelevant to our goal, (ii) proving that the number of relevant subruns to expand occurring at parallel positions can be bounded, and also (iii) avoiding cycles in the inference. In order to give further intuition on how the top-down construction of runs proceeds, the following example applies an analogous idea in the setting of `TA`.

Example 5.5. *Consider the language of terms over $\Sigma = \{a{:}0,\ b{:}0,\ f{:}2\}$ with an even number of a's. This language can be recognized by the* `TA` *A having the following set of transition rules:*

$$\begin{aligned} a &\to q_1\\ b &\to q_0\\ f(q_0,q_0) &\to q_0\\ f(q_0,q_1) &\to q_1\\ f(q_1,q_0) &\to q_1\\ f(q_1,q_1) &\to q_0 \end{aligned}$$

where q_0 is the only final state of A. Our goal is to find a term in $\mathcal{L}(A)$ by non-deterministically constructing an accepting run of A. To this end, we proceed by successively expanding top-down the run being constructed. To simplify the presentation, we denote the positions where such an expansion needs to be performed as:

$$q?$$

where q is a state of A representing the fact that the expansion of this node must produce a subrun reaching q. Since we focus on generating an accepting run, we begin the construction at a final state of A, and hence, the computation starts with the following node:

$$q_0?$$

To expand the previous node, we can choose between three different rules of A. We non-deterministically choose $f(q_1,q_1)\to q_0$ and obtain:

$$\begin{tikzcd}[column sep=tiny] & f(q_1,q_1)\to q_0 \arrow[dl, no head] \arrow[dr, no head] & \\ q_1? & & q_1? \end{tikzcd}$$

Note that now we have two new nodes to expand: the direct children of the root node. This forces us to choose in which order we want to proceed the expansion. We decide to prioritize the expansion of the nodes that have greater estimated height. Proceeding *in order of height is not strictly necessary for plain* TA*, but it will be crucial when adapting the method to* $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$*. We non-deterministically decide that the height of the subrun at position* 1 *will be greater than the height of the subrun at position* 2*. Therefore, we expand at position* 1 *by non-deterministically guessing the transition rule* $f(q_1, q_0) \to q_1$*:*

$$\begin{tikzcd}
 & & f(q_1,q_1) \to q_0 \arrow[dl, dash] \arrow[dr, dash] & \\
 & f(q_1,q_0) \to q_1 \arrow[dl, dash] \arrow[dr, dash] & & q_1? \\
q_1? & & q_0? &
\end{tikzcd}$$

At this point we have three nodes to expand at positions 1.1*,* 1.2*, and* 2*, and again need to estimate their height in order to proceed. Non-deterministically we decide that all the subruns at such positions will have the same height (which is consistent with the previous decision), and thus, we expand them simultaneously. We also guess non-deterministically the transition rules* $a \to q_1$*,* $b \to q_0$*, and* $a \to q_1$ *for each of the positions, respectively, and obtain:*

$$\begin{tikzcd}
 & & f(q_1,q_1) \to q_0 \arrow[dl, dash] \arrow[dr, dash] & \\
 & f(q_1,q_0) \to q_1 \arrow[dl, dash] \arrow[dr, dash] & & a \to q_1 \\
a \to q_1 & & b \to q_0 &
\end{tikzcd}$$

Since there are no more nodes to expand, the computation ends at this point. Note that we have successfully generated an accepting run on the term $f(f(a,b),a)$*, which is in* $\mathcal{L}(A)$*.*

Adapting from TA to $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ the generation of runs sketched in the previous example requires to take especial care with constrained states. For instance, if at a concrete step two nodes with the states q_1 and q_2 are expanded, and the atom $q_1 \not\approx q_2$ occurs in the constraint, then it must be guaranteed that the expansion of those nodes will eventually produce subruns recognizing different subterms. This can be achieved either by making such subterms differ at their root position (i.e., performing an initial expansion of the nodes using transition rules whose alphabet symbols differ), or by deferring it to be checked at some later point of the expansion of the respective children. Note that, due to the fact that the expansion is done in order of height, if those nodes with q_1 and q_2 are expanded at different inference steps, then they will satisfy the constraint for free, since the expansion will produce subruns with different height. The previous ideas are focused on ensuring that the inferences are sound, but nothing is said about completeness.

5.3 Compatible runs

We start by defining a notion of compatibility between runs which, in the next section, allows us to simplify the decision of emptiness and finiteness. Note that any run of a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ A is a run of $\mathtt{ta}(A)$, but the converse is not true since a run r of $\mathtt{ta}(A)$ may not satisfy the global constraint of A. In such case, there must exist two different positions p_1, p_2 of r satisfying that the atom $\mathsf{rhs}(r(p_1)) \not\approx \mathsf{rhs}(r(p_2))$ occurs in the constraint of A and $\mathsf{term}(r|_{p_1}) = \mathsf{term}(r|_{p_2})$. We can see the subruns $r|_{p_1}$ and $r|_{p_2}$ as incompatible, since, whenever a run r' of $\mathtt{ta}(A)$ contains them as subruns, r' is not a run of A since the constraint is falsified.

Definition 5.6. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$. Two runs r_1, r_2 of A are* compatible *if, for every pair of positions $p_1 \in \mathsf{Pos}(r_1)$, $p_2 \in \mathsf{Pos}(r_2)$ such that the global constraint of A contains the atom $\mathsf{rhs}(r_1(p_1)) \not\approx \mathsf{rhs}(r_2(p_2))$, it holds that $\mathsf{term}(r_1|_{p_1}) \neq \mathsf{term}(r_2|_{p_2})$.*

We say that a set of runs of A is a compatible set *if its runs are pairwise compatible.*

The following example illustrates how the incompatibility between runs affects the language recognized by a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$.

Example 5.7. *Let Σ be the signature $\{a{:}0,\ h{:}1,\ f{:}2\}$. Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ more explicitly defined as $A = \langle \{q_f, q_h, q_a\}, \Sigma, \{q_f\}, \Delta, q_a \not\approx q_a \rangle$, where the set of transition rules Δ is:*

$$\begin{aligned} a &\to q_a \\ h(q_a) &\to q_h \\ h(q_h) &\to q_h \\ f(q_h, q_h) &\to q_f \end{aligned}$$

It is easy to see that the language recognized when ignoring the global disequality constraint is $\mathcal{L}(\mathtt{ta}(A)) = \{f(h^n(a), h^m(a)) \mid n, m \geq 1\}$, whereas, as justified below, trying to satisfy the constraint leads to $\mathcal{L}(A)$ being empty. First, note that any term of the form $h^n(a)$, with $n \geq 1$, has an associated run of A of the form:

$$\begin{array}{c} n-1 \begin{cases} h(q_h) \to q_h \\ \quad\vdots \\ h(q_h) \to q_h \end{cases} \\ | \\ h(q_a) \to q_h \\ | \\ a \to q_a \end{array}$$

Now, consider two runs r_1, r_2 of A on the terms $h^n(a)$ and $h^m(a)$, with $n, m \geq 1$, and let p_1, p_2 be the positions of the single leaf in r_1 and r_2, respectively. Note that $\mathsf{term}(r_1|_{p_1}) = \mathsf{term}(r_2|_{p_2}) = a$ and $\mathsf{rhs}(r_1(p_1)) = \mathsf{rhs}(r_2(p_2)) = q_a$. Since the constraint of A is $q_a \not\approx q_a$, the runs r_1, r_2 are not compatible. It follows that any compatible set of runs of A reaching q_h has at most one single run, even though there are infinitely many runs reaching q_h. Thus, $\mathcal{L}(A) = \emptyset$ since reaching the final state q_f requires two compatible subruns reaching q_h.

Let us give some intuition on how the notion of compatibility is related with the reflexivity of the constraints of a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ A, since it is a key point in the proof of the following lemma. Let R be a compatible set of runs of A. Consider a run $r \in R$ and its subrun $r|_p$, for any position $p \in \mathsf{Pos}(r)$ such that $r|_p$ reaches a constrained state. Recall that when a state q occurs in the constraint of a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$, it necessarily occurs at least in an atom of the form $q \not\approx q$, i.e., in a reflexive atom. This fact implies that the only run of R containing $r|_p$ as a subrun is precisely r, since otherwise R would not be a compatible set. Actually, the definition of compatibility further guarantees that only r has a subrun reaching the state $\mathsf{rhs}(r(p))$ after recognizing $\mathsf{term}(r|_p)$. This fact is used in the proof of the following lemma to bound the number of runs in R that are incompatible with a certain fixed run.

Lemma 5.8. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx\mathcal{R}}$. Let r be a run of A, and let R be an infinite compatible set of runs of A.*

Then, there exists $S \subseteq R$ such that $S \cup \{r\}$ is an infinite compatible set of runs.

Proof. Let A be $\langle Q, \Sigma, F, \Delta, C\rangle$ more explicitly written. Let $p \in \mathsf{Pos}(r)$ be a position such that $\mathsf{rhs}(r(p))$ is a constrained state. Since R is a compatible set, there are at most $|Q|$ runs $r' \in R$ for which there exists a position p' satisfying that C contains the atom $\mathsf{rhs}(r'(p')) \not\approx \mathsf{rhs}(r(p))$ and $\mathsf{term}(r'|_{p'}) = \mathsf{term}(r|_p)$. By defining S as the result of removing from R all such runs r' for all such positions p, the result follows. ■

Example 5.9. *Let Σ be the signature $\{a{:}0,\ h{:}1,\ f{:}2\}$. Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx\mathcal{R}}$ more explicitly defined as $A = \langle\{q_f, q_h, q\}, \Sigma, \{q_f\}, \Delta, q_h \not\approx q_h\rangle$, where the set of transition rules Δ is:*

$$\begin{aligned} a &\to q \\ h(q) &\to q \\ h(q) &\to q_h \\ f(q_h, q_h) &\to q_f \end{aligned}$$

Note that its recognized language is $\mathcal{L}(A) = \{f(h^n(a), h^m(a)) \mid n, m \geq 1 \wedge n \neq m\}$. Let R be the set of runs of A on terms of the form $f(h^{2\cdot n}(a), h^{2\cdot n+1}(a))$, with $n \geq 1$. Note that R is infinite, and moreover, it is a compatible set. Consider a run r of A on a term $f(h^{m_1}(a), h^{m_2}(a))$, with $m_1, m_2 \geq 1$ and $m_1 \neq m_2$. By Lemma 5.8, r is compatible with an infinite number of runs in R. In fact, it is easy to see that r can be incompatible with at most two runs in R, the ones recognizing a term where the root node has $h^{m_1}(a)$ or $h^{m_2}(a)$ as a direct child. Hence, there exists an infinite compatible set of runs from R and containing r.

Corollary 5.10. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx\mathcal{R}}$. Let $S_1, \ldots, S_n$ be infinite compatible sets of runs of A.*

Then, there exists $S \subsetneq (S_1 \cup \ldots \cup S_n)$ such that S is an infinite compatible set of runs, and, for every $i \in \{1, \ldots, n\}$, $S \cap S_i$ is infinite.

Proof. By applying Lemma 5.8 several times, we can guarantee that there exists a selection $r_1 \in S_1, \ldots, r_n \in S_n$ satisfying the following conditions: the r_i's are pairwise different, $E := \{r_1, \ldots, r_n\}$ is a compatible set, and there exist $S'_1 \subsetneq S_1, \ldots, S'_n \subsetneq S_n$ such that $S'_1 \uplus E, \ldots, S'_n \uplus E$ are infinite compatible sets. We can repeat the same argument in order to find a new selection $r'_1 \in S'_1, \ldots, r'_n \in S'_n$ satisfying that the r'_i's are pairwise different, $E' := \{r'_1, \ldots, r'_n\}$ is a compatible set, and there exist $S''_1 \subsetneq S'_1, \ldots, S''_n \subsetneq S'_n$ such that $S''_1 \uplus E', \ldots, S''_n \uplus E'$ are infinite compatible sets. Note that $E \uplus E'$ is a compatible set. Clearly, this process can be iterated to generate an infinite number of compatible sets $E, E', E'', \ldots$, pairwise disjoint, and such that the infinite union $S := E \uplus E' \uplus E'' \uplus \ldots$ is the infinite compatible set of the statement. ■

To simplify the decision procedure for emptiness and finiteness of the language recognized by a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ A, we compute which states have infinite sets of compatible runs reaching them. In the next section, this allows us to define from A a simpler automaton such that the emptiness of the recognized language is preserved, and its finiteness may change only under certain conditions.

Definition 5.11. *Let $A = \langle Q, \Sigma, F, \Delta, C\rangle$ be a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$. We define its* set of states with infinite compatible runs*, denoted by Q^{∞}_{A}, to be the set of states $q \in Q$ such that there exists an infinite compatible set of runs of A reaching q.*

We use Q^{∞} as a shorthand when A is clear from the context.

Algorithm 5.12 computes the set Q^{∞} for a given $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ A. Its correctness is stated in the following two lemmas. With respect to its running time, first note that the algorithm checks finiteness and emptiness of the language recognized by the states of the $\mathtt{TA}$ A^0 constructed in step 1. The construction of A^0 can be done in polynomial time, and such properties can be decided in polynomial time for $\mathtt{TA}$ [CDG+07]. Next, in step 3, it incrementally computes the set InfCom in at most $|Q| - 1$ steps, using operations that can be all computed in polynomial time. It follows that the algorithm takes polynomial time.

Algorithm 5.12 Computation of the set Q^{∞} for a given $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ A.

```
Input: a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ $A = \langle Q, \Sigma, F, \Delta, C\rangle$.
(1) Let $A^0$ be the $\mathtt{TA}$ $\langle Q, \Sigma, F, \Delta^0\rangle$, where $\Delta^0$ is the subset of rules in $\Delta$ that have
    no constrained state, i.e., $\Delta^0 = \{(f(q_1, \ldots, q_m) \to q) \in \Delta \mid q_1, \ldots, q_m, q \notin C\}$.
(2) $\mathsf{InfCom} := \{q \in Q \mid \mathcal{L}(A^0, q)$ is infinite$\}$.
(3) While there exists a transition rule $(f(q_1, \ldots, q_m) \to q) \in \Delta$ satisfying that:
      • $q \notin \mathsf{InfCom}$
      • $\exists i \in \{1, \ldots, m\} : q_i \in \mathsf{InfCom}$
      • $\forall i \in \{1, \ldots, m\} : q_i \in \mathsf{InfCom} \vee \mathcal{L}(A^0, q_i) \neq \emptyset$
    do:
      • $\mathsf{InfCom} := \mathsf{InfCom} \uplus \{q\}$.
(4) Output $\mathsf{InfCom}$.
```

Example 5.13. *We apply Algorithm 5.12 to the $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ A from Example 5.9 in order to compute Q^{∞}_{A}. The $\mathtt{TA}$ A^0 constructed in step 1 is obtained from A by discarding the global constraint and the transition rules that involve constrained states. Since only q_h is a constrained state, A^0 is the $\mathtt{TA}$ $\langle\{q_f, q_h, q\}, \Sigma, \{q_f\}, \{a \to q,\ h(q) \to q\}\rangle$. The definition of the set of final states of A^0 as $\{q_f\}$ is arbitrary since we are not interested in the language recognized by A^0 (which in this example is empty), but in the language recognized by each of its states. In particular, note that $\mathcal{L}(A^0, q_h) = \mathcal{L}(A^0, q_f) = \emptyset$ and $\mathcal{L}(A^0, q)$ is infinite. Thus, in step 2,* InfCom *is initialized to be $\{q\}$. Next, the loop of step 3 is executed twice:*

- *In the first iteration, the transition rule $h(q) \to q_h$ of A satisfies all the required conditions, and thus,* InfCom *is set to $\{q, q_h\}$.*

- *In the second iteration, the transition rule $f(q_h, q_h) \to q_f$ of A satisfies all the required conditions, and thus,* InfCom *is set to $\{q, q_h, q_f\}$.*

As InfCom *already contains all the states of A, step 3 finishes, and the output of the algorithm is the set $\{q, q_h, q_f\}$. This was the expected result since $Q_A^\infty = \{q, q_h, q_f\}$ clearly follows from the explanations in Example 5.9: the set R defined there is an infinite compatible set of runs on terms of the form $f(h^{2\cdot n}(a), h^{2\cdot n+1}(a))$, with $n \geq 1$, and hence, $q_f \in Q_A^\infty$ since all the runs in R reach q_f, and moreover, $q, q_h \in Q_A^\infty$ since it is possible to define infinite compatible sets of runs for them by properly extracting subruns of the runs in R.*

Lemma 5.14. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$. Let* InfCom *be the output of Algorithm 5.12 on input A. Let r be a run of A satisfying that $\mathsf{rhs}(r(p)) \in \mathsf{InfCom}$ for each position $p \in \mathsf{Pos}(r) \setminus \{\lambda\}$ such that $r|_p$ reaches a constrained state, and moreover, there is some $p \in \mathsf{Pos}(r)$ such that $\mathsf{rhs}(r(p)) \in \mathsf{InfCom}$.*

Then, $\mathsf{rhs}(r(\lambda)) \in \mathsf{InfCom}$.

Proof. Let S be the set of positions $p \in \mathsf{Pos}(r)$ satisfying that $r|_p$ reaches a state in InfCom and such that they are minimal with respect to the prefix relation $\leq$. Note that S is a set of parallel positions and that, by the assumptions, $S \neq \emptyset$ holds. Let P be the set of prefixes of the positions in S, i.e., $P = \{p \in \mathsf{Pos}(r) \mid \exists p' \in S : p \leq p'\}$. By induction on the terms pending at positions in P, it is easy to see that, for each $p \in P$, the state $\mathsf{rhs}(r(p))$ is added to InfCom either in step 2 or step 3 of the algorithm, since the second and third conditions of step 3 are satisfied by the assumption of the lemma and induction hypothesis. Since $\lambda \in P$, the statement holds. ■

Lemma 5.15. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$. Let* InfCom *be the output of Algorithm 5.12 on input A.*

Then, $\mathsf{InfCom} = Q_A^\infty$.

Proof. Let A be $\langle Q, \Sigma, F, \Delta, C\rangle$ more explicitly written. We prove each direction separately:

$\subseteq$) We first prove soundness showing that any state $q \in \mathsf{InfCom}$ is also in Q_A^∞. We use induction on the number of iterations of the algorithm until q was added to InfCom.

First assume that q was added in step 2 of the algorithm. Hence, there are infinitely many different runs of A^0 reaching q and, since runs of A^0 are compatible runs of A, it follows directly that $q \in Q_A^\infty$.

Now assume that q was added after some iterations of step 3 of the algorithm. Hence, there exists a rule $(f(q_1, \ldots, q_m) \to q) \in \Delta$ where each q_i either has already been added to InfCom or it has non-empty language in A^0. Moreover, there exists at least one j such that q_j is in InfCom. To prove $q \in Q_A^\infty$, it suffices to show that there exists an infinite compatible set of runs of A reaching q with the rule $f(q_1, \ldots, q_m) \to q$ at their root position. We show that this set can be obtained by properly selecting runs reaching each q_i. We consider the following cases:

(a) For every state $q_i \notin \mathsf{InfCom}$, we take any run of A^0 reaching that q_i, which exists because in this case $\mathcal{L}(A^0, q_i) \neq \emptyset$. Note that the selected run of A^0 is also a run of A, and moreover, it is always compatible with any other run of A since it does not involve constrained states.

(b) For all the states in InfCom appearing in the left-hand side of the rule, say the states with indexes $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, m\}$, by induction hypothesis it holds that $q_{i_1}, \ldots, q_{i_k} \in Q_A^\infty$. Moreover, note that there exists at least one such state, i.e., $1 \leq k \leq m$. Hence, by Corollary 5.10, there exists a compatible set containing an infinite number of runs reaching each q_{i_j}.

Using the runs selected in (a) and the infinitely many compatible runs of (b), we can obtain infinitely many compatible runs of A with $f(q_1, \ldots, q_m) \to q$ at their root position, and hence, $q \in Q_A^\infty$.

$\supseteq$) We prove completeness by contradiction. Assume that there is a state $q \in Q_A^\infty$ such that $q \notin \mathsf{InfCom}$. Since $q \in Q_A^\infty$, there exists an infinite compatible set of runs S such that $\mathsf{rhs}(r(\lambda)) = q$, for each $r \in S$. We distinguish two cases.

First, if there exists an infinite subset of runs in S not containing any state occurring in the constraint of A, then $\mathcal{L}(A^0, q)$ is infinite. Therefore, q was added to InfCom in step 2 of the algorithm, in contradiction with the assumption.

Otherwise, there exists an infinite compatible set $R \subseteq S$ such that every run in R contains a state occurring in the constraint of A. Note that, by Lemma 5.14 and the fact that $q \notin \mathsf{InfCom}$ holds by the assumption, we can conclude that each run $r \in R$ satisfies the following property: there exists a position $p \in \mathsf{Pos}(r) \setminus \{\lambda\}$ such that $r|_p$ reaches a constrained state not in InfCom. We define $\tilde{R}$ as the set of subruns of runs in R such that the only occurrence of a constrained state not in InfCom is at their root position. More formally:

$$\begin{aligned}\tilde{R} = \{r|_p \mid\ & r \in R,\ p \in \mathsf{Pos}(r),\\ & \forall p' \in \mathsf{Pos}(r|_p) : (\mathsf{rhs}(r|_p(p')) \in C \wedge \mathsf{rhs}(r|_p(p')) \notin \mathsf{InfCom} \Leftrightarrow p' = \lambda)\}\end{aligned}$$

Note that $\tilde{R}$ is an infinite compatible set. Moreover, every strict subrun of each $r \in \tilde{R}$ does not contain constrained states, since otherwise, $\mathsf{rhs}(r(\lambda)) \in \mathsf{InfCom}$ by Lemma 5.14. It follows that all strict subruns of runs in $\tilde{R}$ are runs of A^0.

Since $\tilde{R}$ is infinite and Δ is finite, there exists an infinite compatible set $\tilde{R}' \subseteq \tilde{R}$ such that every run in $\tilde{R}'$ has the same rule $f(q_1, \ldots, q_m) \to \tilde{q}'$ at root position. Finally, since $\tilde{R}'$ is infinite, there exists $j \in \{1, \ldots, m\}$ such that $\mathcal{L}(A^0, q_j)$ is infinite, and thus, $q_j \in \mathsf{InfCom}$. Hence, $\tilde{q}'$ was added to InfCom in step 3 of the algorithm, a contradiction with the definition of $\tilde{R}$. ■

5.4 Transformation of the automaton

Taking advantage of the fact that the set Q^∞ can be computed, we simplify our problem by transforming the initial $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$ and adopting a slightly different notion of run. One of the goals of the transformation, as shown in the following example, is to simplify runs of the $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ by ignoring subruns reaching states in Q^{∞}, since they are not relevant in our setting.

Example 5.16. *Let A be the $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ defined in Example 5.9. Recall that the constraint of A concerns runs reaching q_h, and yet, as seen in Example 5.13, $q_h \in Q^{\infty}_{A}$. Therefore, the constraint is not relevant for the emptiness decision, since enough compatible runs reaching q_h can always be found. For this reason, we will represent all terms reaching q_h with a new constant symbol $\blacktriangle$. If we replace the rule $h(q) \to q_h$ by the new rule $\blacktriangle \to q_h$, then emptiness of the language is preserved under a relaxed notion of run. In this new notion, satisfiability of constraints is reinterpreted so that they are additionally satisfied when the symbol $\blacktriangle$ appears in the terms associated with the involved subruns, i.e., a subrun with $\blacktriangle$ always satisfies a disequality.*

As a final remark, note that after representing all the terms reaching q_h by the constant symbol $\blacktriangle$, the language recognized is $\{f(\blacktriangle, \blacktriangle)\}$, which is finite although the original language was infinite. Hence, finiteness of the recognized language is not preserved by this transformation, but it is easy to see that any occurrence of the symbol $\blacktriangle$ in a term of the language guarantees that the original language was infinite.

The actual transformation of the automaton that we will propose is more complex than the previous sketch. In particular, it will modify the recognized language of all *states in Q^{∞}_{A}. Thus, A would be transformed in such a way that its recognized language would be reduced to $\{\blacktriangle\}$ instead of $\{f(\blacktriangle, \blacktriangle)\}$, since its final state q_f is in Q^{∞}_{A}.*

As seen in the previous example, we need to compare terms using the following notion, which depends on a special symbol of the signature denoted by $\blacktriangle$.

Definition 5.17. *Let Σ be a signature, and let $\blacktriangle$ be a symbol in Σ. We define the relation $=_{\blacktriangle}$ on $\mathcal{T}(\Sigma)$ as $t_1 =_{\blacktriangle} t_2$ if and only if $t_1 = t_2$ and there is no occurrence of $\blacktriangle$ in t_1 or t_2.*

Hence, if $\blacktriangle$ occurs in t_1 or t_2, then $t_1 \neq_{\blacktriangle} t_2$. Note that $=_{\blacktriangle}$ is a partial equivalence relation, i.e., it is symmetric and transitive, but not reflexive.

Now, we formally define the notion of run commented in the previous example in terms of $=_{\blacktriangle}$. The difference with the usual definition of run is that a term containing $\blacktriangle$ always satisfies a disequality with any other term (even itself).

Definition 5.18. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ with signature Σ, and let $\blacktriangle$ be a symbol in Σ. We define a $\blacktriangle$-*run *r of A as a run of $\mathtt{ta}(A)$ satisfying that, for every pair of distinct positions $p_1, p_2 \in \mathsf{Pos}(r)$ such that the global constraint of A contains the atom $\mathsf{rhs}(r(p_1)) \not\approx \mathsf{rhs}(r(p_2))$, it holds that $\mathsf{term}(r|_{p_1}) \neq_{\blacktriangle} \mathsf{term}(r|_{p_2})$. With $\mathcal{L}_{\blacktriangle}(A)$ we denote the set of terms $t \in \mathcal{T}(\Sigma)$ such that there exists a $\blacktriangle$-run of A on t reaching a final state.*

*Analogously to Definition 5.6, two $\blacktriangle$-runs r_1, r_2 of A are $\blacktriangle$-*compatible *if, for every pair of positions $p_1 \in \mathsf{Pos}(r_1)$, $p_2 \in \mathsf{Pos}(r_2)$ such that the global constraint of A contains the atom $\mathsf{rhs}(r_1(p_1)) \not\approx \mathsf{rhs}(r_2(p_2))$, it holds that $\mathsf{term}(r_1|_{p_1}) \neq_{\blacktriangle} \mathsf{term}(r_2|_{p_2})$.*

We are now ready to formalize the transformation of the automaton sketched in Example 5.16. It suffices to introduce a new constant symbol $\blacktriangle$, a new state $q_{\blacktriangle}$, and a new rule $\blacktriangle \to q_{\blacktriangle}$, also make such new state $q_{\blacktriangle}$ final in the case where $F \cap Q^{\infty}$ was not empty, remove all transition rules where the right-hand side is a state in Q^∞, and finally, replace by $q_\blacktriangle$ any state in Q^∞ occurring in the left-hand side of the remaining transition rules. It is clear that any state that was in Q^∞ becomes useless after such transformation, and thus, we could remove all of them from the state set, final state set, and the global constraint, without further affecting the recognized language. It is also possible that other states become useless if they only appeared in runs reaching states in Q^∞. These states could also be cleaned. In either case, such cleaning is not strictly necessary, and thus, we keep the useless states in the resulting automaton in order to simplify the description of the transformation.

To ease the presentation of our decision procedure for the emptiness and finiteness for $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$, we combine the previous transformation introducing $\blacktriangle$ with another one that extends the information recorded by the states. In particular, we want that the state reached at the root position of any $\blacktriangle$-run has information of whether a constrained state has appeared in such $\blacktriangle$-run. To this end, it suffices to split each state q into a state q^{c} and a state $q^{\bar{\mathsf{c}}}$, and modify the set of rules to guarantee the following property: a state of the form q^{c} is only reachable with $\blacktriangle$-runs that involve some constrained state, and similarly, a state of the form $q^{\bar{\mathsf{c}}}$ is only reachable with $\blacktriangle$-runs that do not involve any constrained state. The following definition formalizes this combined transformation.

Definition 5.19. *Let $A = \langle Q, \Sigma, F, \Delta, C\rangle$ be a $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$, and let $\blacktriangle$ be a symbol not in Σ. We define the $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$ $A_\blacktriangle$ as $\langle Q^{\mathsf{c}}_\blacktriangle \uplus Q^{\bar{\mathsf{c}}}_\blacktriangle, \Sigma_\blacktriangle, F_\blacktriangle, \Delta_\blacktriangle, C_\blacktriangle\rangle$, where:*

- $Q^x_\blacktriangle = \{q^x \mid q \in Q \uplus \{q_\blacktriangle\}\}$ *with* $x \in \{\mathsf{c}, \bar{\mathsf{c}}\}$,
- $\Sigma_\blacktriangle = \Sigma \uplus \{\blacktriangle{:}0\}$,
- $F_\blacktriangle = \{q^x \mid q \in F \wedge x \in \{\mathsf{c}, \bar{\mathsf{c}}\}\} \uplus \{q^x_\blacktriangle \mid F \cap Q^\infty_A \neq \emptyset \wedge x \in \{\mathsf{c}, \bar{\mathsf{c}}\}\}$,
- $C_\blacktriangle = \{q_1^{\mathsf{c}} \not\approx q_2^{\mathsf{c}} \mid (q_1 \not\approx q_2) \in C\}$,
- $\Delta_\blacktriangle$ *is the set of transition rules containing $\blacktriangle \to q^{\bar{\mathsf{c}}}_\blacktriangle$ and also all the rules of the form $f(q_1^{x_1}, \ldots, q_m^{x_m}) \to q^x$, with $x_1, \ldots, x_m, x \in \{\mathsf{c}, \bar{\mathsf{c}}\}$ such that $x = \mathsf{c}$ if and only if $q \in C$ or there is some $i \in \{1, \ldots, m\}$ such that $x_i = \mathsf{c}$, and satisfying that there exists a rule $(f(q'_1, \ldots, q'_m) \to q) \in \Delta$ such that $q \notin Q^\infty_A$ and, for every $i \in \{1, \ldots, m\}$, if $q'_i \in Q^\infty_A$ then $q_i = q_\blacktriangle$, and $q_i = q'_i$ otherwise.*

Note that, by definition, no rule of $A_\blacktriangle$ has $q^{\mathsf{c}}_\blacktriangle$ as right-hand side, and thus, such state is useless. The only reason to distinguish between $q^{\mathsf{c}}_\blacktriangle$ and $q^{\bar{\mathsf{c}}}_\blacktriangle$ was to simplify the presentation of the transformation, and henceforth, we refer to $q^{\bar{\mathsf{c}}}_\blacktriangle$ simply as $q_\blacktriangle$. Also note that $A_\blacktriangle$ can be computed with time in $\mathcal{O}(|\Delta| \cdot 2^{\mathsf{maxar}} + |Q| + |C|)$, since, in particular, $|\Delta_\blacktriangle| \leq 1 + |\Delta| \cdot 2^{\mathsf{maxar}}$ and $|Q^{\mathsf{c}}_\blacktriangle \uplus Q^{\bar{\mathsf{c}}}_\blacktriangle| = 2 \cdot (|Q| + 1)$.

Example 5.20. *Let Σ be the signature $\{a{:}0, h{:}1, g{:}2, f{:}2\}$. Consider the language of terms of the form $f(e_1, f(e_2, \ldots f(e_m, a) \ldots))$, where each e_i is a term of the form $g(h^l(a), h^r(a))$ with $l, r \geq 1$, satisfying that all the subterms $e_1|_1, \ldots, e_m|_1$ are pairwise different (i.e., the l's are distinct), and all the subterms $e_1|_2, \ldots, e_m|_2$ are pairwise different (i.e., the r's are distinct) and have height bounded by a given $n \geq 1$ (i.e., all the r's satisfy $1 \leq r \leq n$). Such language is recognized by the $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$*

$A = \langle\{q_{\mathsf{list}}, q_g, q, q_l, q_0, \ldots, q_{n-1}, q_r\}, \Sigma, \{q_{\mathsf{list}}\}, \Delta, q_l \not\approx q_l \wedge q_r \not\approx q_r\rangle$, where the set of transition rules Δ is:

$$\begin{array}{rll}
a \rightarrow & q & \\
h(q) \rightarrow & q & \\
h(q) \rightarrow & q_l & \\
a \rightarrow & q_0 & \\
h(q_i) \rightarrow & q_{i+1} & \textit{for } i \in \{0, \ldots, n-2\} \\
h(q_i) \rightarrow & q_r & \textit{for } i \in \{0, \ldots, n-1\} \\
g(q_l, q_r) \rightarrow & q_g & \\
a \rightarrow & q_{\mathsf{list}} & \\
f(q_g, q_{\mathsf{list}}) \rightarrow & q_{\mathsf{list}} &
\end{array}$$

Note that the constraint on q_r could be equivalently checked with plain `TA` *techniques since $\mathcal{L}(A, q_r)$ is finite. Nevertheless, this would require an automaton with a number of states in $2^{\mathcal{O}(n)}$.*

To apply the construction of Definition 5.19 on A, we first need to compute Q_A^∞. It is easy to see that Q_A^∞ only contains the states q and q_l since, in particular, the states $q_0, \ldots, q_{n-1}, q_r$ recognize finite languages, and this fact combined with the constraint on q_r implies that there are no infinite compatible sets of runs reaching q_g or q_{list}. Thus, the $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ $A_\blacktriangle$ over the extended signature $\Sigma_\blacktriangle = \Sigma \uplus \{\blacktriangle{:}0\}$ is defined as follows. Its set of states is obtained by adding the new state $q_\blacktriangle$ and applying the labels c (meaning that the $\blacktriangle$-run has seen a constrained state) and $\bar{\mathsf{c}}$ (meaning that only non-constrained states have been seen in the $\blacktriangle$-run) to each of the states, i.e., its set of states is $\{q^x_\blacktriangle, q^x_{\mathsf{list}}, q^x_g, q^x, q^x_l, q^x_0, \ldots, q^x_{n-1}, q^x_r \mid x \in \{\mathsf{c}, \bar{\mathsf{c}}\}\}$. Its set of final states is $\{q^{\mathsf{c}}_{\mathsf{list}}, q^{\bar{\mathsf{c}}}_{\mathsf{list}}\}$, since the original set of final states only contained q_{list}, and such state is not in Q_A^∞ (otherwise, we would also include $q^{\mathsf{c}}_\blacktriangle$ and $q^{\bar{\mathsf{c}}}_\blacktriangle$ as final states). Its global disequality constraint is $q^{\mathsf{c}}_l \not\approx q^{\mathsf{c}}_l \wedge q^{\mathsf{c}}_r \not\approx q^{\mathsf{c}}_r$. Finally, to obtain its set of transition rules we ignore the rules of A reaching a state in Q_A^∞ (i.e., $a \rightarrow q$, $h(q) \rightarrow q$, $h(q) \rightarrow q_l$), replace by $q_\blacktriangle$ all states in Q_A^∞ occurring in the left-hand side of the remaining rules, and properly label the states with c or $\bar{\mathsf{c}}$. We also have to add the rule $\blacktriangle \rightarrow q^{\bar{\mathsf{c}}}_\blacktriangle$. Hence, the resulting set of transition rules is:

$$\begin{array}{rll}
\blacktriangle \rightarrow & q^{\bar{\mathsf{c}}}_\blacktriangle & \\
a \rightarrow & q^{\bar{\mathsf{c}}}_0 & \\
h(q^x_i) \rightarrow & q^x_{i+1} & \textit{for } i \in \{0, \ldots, n-2\} \textit{ and } x \in \{\mathsf{c}, \bar{\mathsf{c}}\} \\
h(q^x_i) \rightarrow & q^{\mathsf{c}}_r & \textit{for } i \in \{0, \ldots, n-1\} \textit{ and } x \in \{\mathsf{c}, \bar{\mathsf{c}}\} \\
g(q^x_\blacktriangle, q^y_r) \rightarrow & q^z_g & \textit{for } x, y, z \in \{\mathsf{c}, \bar{\mathsf{c}}\} \textit{ such that } (x = \mathsf{c} \vee y = \mathsf{c}) \Leftrightarrow (z = \mathsf{c}) \\
a \rightarrow & q^{\bar{\mathsf{c}}}_{\mathsf{list}} & \\
f(q^x_g, q^y_{\mathsf{list}}) \rightarrow & q^z_{\mathsf{list}} & \textit{for } x, y, z \in \{\mathsf{c}, \bar{\mathsf{c}}\} \textit{ such that } (x = \mathsf{c} \vee y = \mathsf{c}) \Leftrightarrow (z = \mathsf{c})
\end{array}$$

Note that $A_\blacktriangle$ has many useless states. For instance, $q^{\mathsf{c}}_0, \ldots, q^{\mathsf{c}}_{n-1}$ are unreachable, and since q_r is constrained in A, $q^{\bar{\mathsf{c}}}_r$ is unreachable in $A_\blacktriangle$, and thus, so is $q^{\bar{\mathsf{c}}}_g$. Also, as expected, q^x and q^x_l with $x \in \{\mathsf{c}, \bar{\mathsf{c}}\}$ are useless because there is no rule reaching them since $q, q_l \in Q_A^\infty$. As a final remark, it is easy to verify that any $\blacktriangle$-run of $A_\blacktriangle$ reaches a state labeled with c if and only if it has some occurrence of a constrained state.

In order to give more intuition on the transformation, we discuss some important properties of the automaton $A_\blacktriangle$. As a first remark, note that the fact that a $\blacktriangle$-run r

of $A_\blacktriangle$ reaches a state in $Q^{\mathsf{c}}_\blacktriangle$ if and only if r involves some state occurring in $C_\blacktriangle$ can be proved by induction on $\mathsf{height}(r)$ and distinguishing cases according to the definition of $\Delta_\blacktriangle$. Now, note that any state of the form $q^{\bar{\mathsf{c}}}$ necessarily recognizes a finite language over $\mathcal{T}(\Sigma_\blacktriangle)$. This is easy to see when q is a constrained state of A, as $q^{\bar{\mathsf{c}}}$ is useless and its language is empty. Otherwise, the property can be proved by contradiction: if $q^{\bar{\mathsf{c}}}$ can be reached by an infinite number of $\blacktriangle$-runs not involving constrained states, then $q \in Q^\infty_A$ by Lemmas 5.14 and 5.15, and hence, $q^{\bar{\mathsf{c}}}$ is useless and its language is empty by Definition 5.19, a contradiction. Next, note that every $\blacktriangle$-run of $A_\blacktriangle$ with a rule of the form $f(\ldots, q_\blacktriangle, \ldots) \to q^x$ at its root position, for $x \in \{\mathsf{c}, \bar{\mathsf{c}}\}$, necessarily satisfies $q^x \in Q^{\mathsf{c}}_\blacktriangle$, i.e., $x = \mathsf{c}$. If this was not the case, then it is trivial to see that $q \in Q^\infty_A$ follows from Lemmas 5.14 and 5.15, which leads to a contradiction with the form of the rule and the definition of $\Delta_\blacktriangle$. Finally, we give a technical statement that is also related to rules having an occurrence of $q_\blacktriangle$ at their left-hand side.

Lemma 5.21. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx\mathcal{R}}$. Let r be a $\blacktriangle$-run of $A_\blacktriangle$. Let p be a position and let k_1 be a natural number such that $p.k_1 \in \mathsf{Pos}(r)$ and $\mathsf{rhs}(r(p.k_1)) = q_\blacktriangle$.*

Then, there is a natural number k_2 such that $p.k_2 \in \mathsf{Pos}(r)$ and $\mathsf{rhs}(r(p.k_2)) \neq q_\blacktriangle$.

Proof. Assuming that $r(p)$ is a rule of the form $f(q_\blacktriangle, \ldots, q_\blacktriangle) \to q$ leads to a contradiction with the fact that $\mathsf{rhs}(r(p)) \neq q_\blacktriangle$ by Lemmas 5.14 and 5.15, and Definition 5.19. ■

We now characterize how the language recognized by a $\mathtt{TAG}^{\wedge}_{\not\approx\mathcal{R}}$ A is related to $\mathcal{L}_\blacktriangle(A_\blacktriangle)$ in terms of emptiness and finiteness. We start with a technical lemma stating that, if $\mathcal{L}_\blacktriangle(A_\blacktriangle)$ contains a term with an occurrence of $\blacktriangle$, then $\mathcal{L}(A)$ has infinite terms.

Lemma 5.22. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx\mathcal{R}}$. Let r be an accepting $\blacktriangle$-run of $A_\blacktriangle$ on a term with an occurrence of $\blacktriangle$.*

Then, there exist infinitely many accepting runs of A.

Proof. The case where $r = (\blacktriangle \to q_\blacktriangle)$ follows trivially. Otherwise, let A be more explicitly written as $\langle Q, \Sigma, F, \Delta, C\rangle$, and let $M : \mathsf{Pos}(r) \to Q$ be a mapping satisfying:

- $M(p) = q$ if $r(p) = (l \to q^x) \neq (\blacktriangle \to q_\blacktriangle)$, where $x \in \{\mathsf{c}, \bar{\mathsf{c}}\}$,
- $M(p) = q$ if $r(p) = (\blacktriangle \to q_\blacktriangle)$, where $q \in Q^\infty_A$,
- for each $p \in \mathsf{Pos}(r)$ such that $\mathsf{term}(r)$ at position p is labeled by a symbol $f \in \Sigma^{(m)}$ different from $\blacktriangle$, the rule $f(M(p.1), \ldots, M(p.m)) \to M(p)$ is in Δ.

Note that such mapping M exists by Definition 5.19. Let $\{p_1, \ldots, p_n\}$ be the set of positions in $\mathsf{Pos}(r)$ satisfying $r(p_i) = (\blacktriangle \to q_\blacktriangle)$ and, for each $i \in \{1, \ldots, n\}$, let S_i be an infinite set of compatible runs of A reaching $M(p_i)$. Such infinite sets exist by definition of M since $M(p_i) \in Q^\infty_A$. By Corollary 5.10, there exists $S \subseteq (S_1 \cup \ldots \cup S_n)$ such that S is an infinite compatible set of runs of A and, for every $i \in \{1, \ldots, n\}$, $S \cap S_i$ is infinite. This fact guarantees that it is possible to replace the $\blacktriangle$-subruns of r at positions $p_1, \ldots, p_n$ to obtain infinitely many accepting runs of A. ■

The following two lemmas state that emptiness of the recognized language is preserved by the transformation, and show how its finiteness is changed.

Lemma 5.23. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$. $\mathcal{L}(A)$ is empty if and only if $\mathcal{L}_{\blacktriangle}(A_{\blacktriangle})$ is empty.*

Proof. We prove each direction separately:

$\Rightarrow$) Given an accepting run r of A, it is easy to construct an accepting $\blacktriangle$-run of $A_{\blacktriangle}$. Since the precise construction is quite technical, we just describe it briefly. Intuitively, it consists in replacing the subruns of r at minimal positions (with respect to the prefix relation $\leq$) that reach a state in Q^{∞}_{A} by the $\blacktriangle$-run ($\blacktriangle \to q_{\blacktriangle}$), and changing each state q by q^{c} or $q^{\bar{\mathsf{c}}}$ depending on whether there is a constrained state in the corresponding $\blacktriangle$-subrun.

$\Leftarrow$) Let r be an accepting $\blacktriangle$-run of $A_{\blacktriangle}$ on a term t. The case where t does not contain any occurrence of $\blacktriangle$ is straightforward: it suffices to replace in r every occurrence of a state of the form q^{c} or $q^{\bar{\mathsf{c}}}$ by q, thus obtaining an accepting run r' of A on t. Otherwise, the statement follows from Lemma 5.22. ■

Lemma 5.24. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$. $\mathcal{L}(A)$ is infinite if and only if $\mathcal{L}_{\blacktriangle}(A_{\blacktriangle})$ is infinite or it contains a term with an occurrence of $\blacktriangle$.*

Proof. We prove each direction separately:

$\Rightarrow$) If there exist infinitely many accepting runs of A not containing any state in Q^{∞}_{A}, then those runs can easily be converted into accepting $\blacktriangle$-runs of $A_{\blacktriangle}$ by properly changing each state q by q^{c} or $q^{\bar{\mathsf{c}}}$, and thus, $\mathcal{L}_{\blacktriangle}(A_{\blacktriangle})$ is infinite and the statement holds. Otherwise, let r be an accepting run of A containing some state in Q^{∞}_{A}. It is easy to construct from r an accepting $\blacktriangle$-run r' of $A_{\blacktriangle}$ by replacing the subruns of r at minimal positions (with respect to the prefix relation $\leq$) that reach a state in Q^{∞}_{A} by the $\blacktriangle$-run ($\blacktriangle \to q_{\blacktriangle}$), and changing each state q by q^{c} or $q^{\bar{\mathsf{c}}}$ depending on whether there is a constrained state in the corresponding $\blacktriangle$-subrun. The accepting $\blacktriangle$-run r' recognizes a term with an occurrence of $\blacktriangle$, and we are done.

$\Leftarrow$) If there exist infinitely many accepting $\blacktriangle$-runs of $A_{\blacktriangle}$ not containing the state $q_{\blacktriangle}$, then those $\blacktriangle$-runs can easily be converted into accepting runs of A by replacing each state of the form q^{c} or $q^{\bar{\mathsf{c}}}$ by q, and thus, $\mathcal{L}(A)$ is infinite and the statement holds. Otherwise, in the case where there exists an accepting $\blacktriangle$-run of $A_{\blacktriangle}$ containing the state $q_{\blacktriangle}$, the statement follows from Lemma 5.22. ■

The following lemma is crucial in our global approach. It gives an upper bound for the number of $\blacktriangle$-runs that can be pairwise $\blacktriangle$-compatible when each of the $\blacktriangle$-runs contains some constrained state.

Lemma 5.25. *Let $A = \langle Q, \Sigma, F, \Delta, C\rangle$ be a $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$. Let $r_1, \dots, r_n$ be $\blacktriangle$-runs of $A_{\blacktriangle}$ pairwise $\blacktriangle$-compatible and such that r_i reaches a state in $Q^{\mathsf{c}}_{\blacktriangle}$, for $i \in \{1, \dots, n\}$.*

Then, $n \leq |Q| \cdot |\Sigma|^{\mathsf{maxar}^{|Q|}}$.

Proof. For each $i \in \{1, \dots, n\}$, let r'_i be a $\blacktriangle$-subrun of r_i reaching a state in $Q^{\mathsf{c}}_{\blacktriangle}$ and with no other occurrence of states in $Q^{\mathsf{c}}_{\blacktriangle}$ (such $\blacktriangle$-subrun exists by the assumptions of the lemma). Note that $r'_1, \dots, r'_n$ are pairwise $\blacktriangle$-compatible.

We argue by contradiction assuming that $n > |Q| \cdot |\Sigma|^{\mathsf{maxar}^{|Q|}}$. Let each r'_i be more explicitly written as $(l_i \to q_i^{\mathsf{c}})(r'_{i,1}, \ldots, r'_{i,m_i})$. Note that, for each $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m_i\}$, the state $q_\blacktriangle$ does not occur in $r'_{i,j}$, since otherwise $q_i \in Q_A^\infty$ holds by Lemmas 5.14 and 5.15, and Definition 5.19, implying that q_i^{c} is useless by the definition of $\Delta_\blacktriangle$, a contradiction. Moreover, $\mathsf{height}(r'_{i,j}) < |Q| - 1$, since otherwise $r'_{i,j}$ can be pumped, implying again that $q_i \in Q_A^\infty$ by Lemmas 5.14 and 5.15, and Definition 5.19, and leading to a contradiction. The bound $|Q| - 1$ is enough since $r'_{i,j}$ cannot have any occurrence of a state of the form q^{c} (otherwise $r'_{i,j}$ would have a constrained state) or any occurrence of $q_i^{\bar{\mathsf{c}}}$ (which is useless since $q_i \in C$). Now, note that the number of different terms of height h is bounded by $|\Sigma|^{\mathsf{maxar}^{h+1}}$, and hence, by the assumption that $n > |Q| \cdot |\Sigma|^{\mathsf{maxar}^{|Q|}}$, it follows that there exist different $i, j \in \{1, \ldots, n\}$ such that $q_i^{\mathsf{c}} = q_j^{\mathsf{c}}$ and $\mathsf{term}(r'_i) =_\blacktriangle \mathsf{term}(r'_j)$. This is in contradiction with the $\blacktriangle$-compatibility of r'_i and r'_j since the atom $q_i^{\mathsf{c}} \not\approx q_j^{\mathsf{c}}$ is necessarily in $C_\blacktriangle$. ■

The following corollary is not used in the remaining arguments, but we have included it since it is a direct consequence of Lemma 5.25, and its statement gives more intuition on an important property of $\blacktriangle$-runs. Intuitively, it states that, for any $\blacktriangle$-run r of $A_\blacktriangle$, there exists a bound for the number of occurrences of states in $Q_\blacktriangle^{\mathsf{c}}$ at parallel positions of r.

Corollary 5.26. *Let $A = \langle Q, \Sigma, F, \Delta, C \rangle$ be a $\mathtt{TAG}^{\wedge}_{\not\approx\mathcal{R}}$. Let r be a $\blacktriangle$-run of $A_\blacktriangle$. Let $p_1, \ldots, p_n \in \mathsf{Pos}(r)$ be pairwise parallel positions such that the $\blacktriangle$-subrun $r|_{p_i}$ reaches a state in $Q_\blacktriangle^{\mathsf{c}}$, for $i \in \{1, \ldots, n\}$.*

Then, $n \leq |Q| \cdot |\Sigma|^{\mathsf{maxar}^{|Q|}}$.

5.5 Emptiness and finiteness decision algorithms

As a consequence of Lemma 5.23, deciding emptiness of the language recognized by a given $\mathtt{TAG}^{\wedge}_{\not\approx\mathcal{R}}$ A can be reduced to test whether there exists an accepting $\blacktriangle$-run of $A_\blacktriangle$. And, as a consequence of Lemma 5.24, deciding finiteness of the language recognized by A can be reduced to test whether $\mathcal{L}_\blacktriangle(A_\blacktriangle)$ is infinite or contains a term with an occurrence of the symbol $\blacktriangle$. We tackle both problems with an algorithm that non-deterministically simulates the construction of accepting $\blacktriangle$-runs in a top-down manner. More concretely, in an intermediate step of the algorithm, the top-most part of an accepting $\blacktriangle$-run r has been already non-deterministically constructed, and it remains to determine its $\blacktriangle$-subruns $r_1, \ldots, r_n$ at certain parallel positions. Moreover, such r_i's are required to reach some specific states, and to recognize terms satisfying certain equality and disequality constraints (with respect to $=_\blacktriangle$) between them. The states that the r_i's have to reach are determined by the part of r that has already been constructed. The (dis)equality constraints between the terms recognized by the r_i's are either determined by the constraint $C_\blacktriangle$ and the states reached by the r_i's, or they are inherited from the part of r that has already been constructed. The algorithm proceeds by guessing the rule at the root position of some of the r_i's, hence extending the constructed part of r. In particular, the r_i's whose root is determined at this step of the algorithm are the ones that are guessed to have maximal height among all the r_i's. By always extending r in this order, we partially construct in the same step all the $\blacktriangle$-subruns of r that have identical height, which allows to either satisfy or propagate the (dis)equality constraints that have to be satisfied.

The algorithm is presented as an inference system that deals with pairs of the form $\langle M, S\rangle$, where M and S are partitions of labeled states of $A_\blacktriangle$, i.e., M and S are sets of non-empty disjoint sets of pairs $\langle q, \ell\rangle$. To ease the presentation, we denote the labeled states $\langle q, \ell\rangle$ as q^ℓ. Our labels are used simply as identifiers to distinguish repeated occurrences of the same state. We define the labels as sequences of natural numbers standing for the position in the constructed $\blacktriangle$-run where the state occurs.

Let us specify the role of $\langle M, S\rangle$ and how this data structure is helpful to formalize the behaviour of the algorithm as sketched above. The inference starts with a pair of the form $\langle\{\{q_f^\lambda\}\}, \emptyset\rangle$, where q_f is guessed among the final states of $A_\blacktriangle$, and then non-deterministically constructs a $\blacktriangle$-run r reaching q_f, if possible. This construction is done top-down, by guessing the rules of $\Delta_\blacktriangle$ to be used. In an intermediate step, $\langle M, S\rangle$ contains the states at the deepest positions of the partially constructed r, i.e., for an element q^ℓ of $\langle M, S\rangle$ it holds that $\mathsf{rhs}(r(\ell)) = q$. The process guarantees some invariant properties on $\langle M, S\rangle$ to keep track of the constraints imposed by the automaton. Consider two different elements $q_1^{\ell_1}, q_2^{\ell_2}$ of $\langle M, S\rangle$. If both of them occur in M, then $\mathsf{height}(r|_{\ell_1}) = \mathsf{height}(r|_{\ell_2})$. If one of them, say $q_1^{\ell_1}$, occurs in M and the other one, say $q_2^{\ell_2}$, occurs in S, then $\mathsf{height}(r|_{\ell_1}) > \mathsf{height}(r|_{\ell_2})$. Moreover, $q_1^{\ell_1}$ and $q_2^{\ell_2}$ belong to the same part in M or S if and only if $\mathsf{term}(r|_{\ell_1}) =_\blacktriangle \mathsf{term}(r|_{\ell_2})$.

Before introducing the inference system, we start by giving a definition that relates pairs $\langle M, S\rangle$ with $\blacktriangle$-runs that satisfy the conditions imposed by the pair. Recall that, given a set of sets T, we use the notation $\bigcup T$ as shorthand for $\bigcup_{P\in T} P$, and that we write $e_1 \sim_T e_2$ to denote that the elements e_1 and e_2 belong to the same set in T.

Definition 5.27. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx\mathcal{R}}$. Let M, S be such that $M \uplus S$ is a partition of the set $\{q_1^{\ell_1}, \dots, q_n^{\ell_n}\}$ of labeled states of $A_\blacktriangle$. Let $r_1, \dots, r_n$ be $\blacktriangle$-runs of $A_\blacktriangle$. We say that $r_1, \dots, r_n$* fit *$\langle M, S\rangle$ if the following conditions hold:*

(F$_1$) r_i reaches the state q_i, for each $i \in \{1, \dots, n\}$,

(F$_2$) $\left(q_i^{\ell_i} \in \bigcup M\right) \Leftrightarrow (\mathsf{height}(r_i) = \max\{\mathsf{height}(r_j) \mid j \in \{1, \dots, n\}\})$, for each $i \in \{1, \dots, n\}$ such that $q_i \neq q_\blacktriangle$,

(F$_3$) $\left(q_i^{\ell_i} \sim_{M\uplus S} q_j^{\ell_j}\right) \Leftrightarrow (\mathsf{term}(r_i) =_\blacktriangle \mathsf{term}(r_j))$, for each distinct $i, j \in \{1, \dots, n\}$,

(F$_4$) $r_1, \dots, r_n$ are pairwise $\blacktriangle$-compatible.

Let us remark that condition (F$_2$) ignores the $\blacktriangle$-runs reaching the state $q_\blacktriangle$. This is because such $\blacktriangle$-runs recognize the term $\blacktriangle$, which is used as a representation for an infinite number of terms. Also note that a particular consequence of condition (F$_3$) is that, if there is some occurrence of $q_\blacktriangle$ in M or S, then it must be in a part of the form $\{q_\blacktriangle^\ell\}$, otherwise it is impossible to find $\blacktriangle$-runs fitting such partition.

Example 5.28. *Let Σ be the signature $\{a{:}0,\ b{:}0,\ h{:}1,\ f{:}2\}$. Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx\mathcal{R}}$ defined* *as $\langle\{q_f, q_h, q\}, \Sigma, \{q_f\}, \Delta, q \not\approx q\rangle$, where the set of transition rules Δ is:*

$$\begin{aligned} a &\rightarrow q \\ b &\rightarrow q \\ h(q) &\rightarrow q_h \\ h(q_h) &\rightarrow q_h \\ f(q_h, q_h) &\rightarrow q_f \end{aligned}$$

Note that $\mathcal{L}(A) = \{f(h^n(\alpha_1), h^m(\alpha_2)) \mid n, m \geq 1 \wedge \alpha_1, \alpha_2 \in \{a, b\} \wedge \alpha_1 \neq \alpha_2\}$, and that $\mathcal{L}(A) = \mathcal{L}_\blacktriangle(A_\blacktriangle)$ since $Q_A^\infty = \emptyset$. Moreover, it is easy to see that the states of $A_\blacktriangle$ of the form $q^{\bar{\mathsf{c}}}$ are useless. For this reason, in the discussion below we refer to the states of the form q^{c} simply as q.

Note that any accepting $\blacktriangle$-run of $A_\blacktriangle$ fits the pair $\langle\{\{q_f^\lambda\}\}, \emptyset\rangle$. In particular, the $\blacktriangle$-run on the term $t = f(h(h(a)), h(b))$ fits such pair. Now, consider the $\blacktriangle$-runs on the terms $t|_1 = h(h(a))$ and $t|_2 = h(b)$, and note that they fit the pair $\langle\{\{q_h^1\}\}, \{\{q_h^2\}\}\rangle$ but not $\langle\{\{q_h^1\}, \{q_h^2\}\}, \emptyset\rangle$, since the height of $t|_1$ is greater than the height of $t|_2$ (thus falsifying condition (F$_2$) of fitness for the second pair). Moreover, observe that the $\blacktriangle$-runs on the terms $t|_{1.1} = h(a)$ and $t|_2 = h(b)$ do not fit $\langle\{\{q_h^1\}\}, \{\{q_h^2\}\}\rangle$ since their height coincides (thus falsifying condition (F$_2$) of fitness). Finally, note that there are no $\blacktriangle$-runs fitting pairs of the form $\langle\{\{q^{\ell_1}, q^{\ell_2}\}\}, \emptyset\rangle$, since the atom $q \not\approx q$ occurring in the constraint requires the $\blacktriangle$-runs reaching q to recognize different terms (with respect to $=_\blacktriangle$) in order to be $\blacktriangle$-compatible, and yet, they are forced to be equal since q^{ℓ_1} and q^{ℓ_2} are in the same part (thus either condition (F$_3$) or (F$_4$) of fitness is falsified).

As a final ingredient to present the inference system, we define the clean operation on partitions of labeled states. Its goals are (i) to erase the occurrences of the state $q_\blacktriangle$ from the given partition, and (ii) to collapse any two labeled states q^{ℓ_1}, q^{ℓ_2} occurring in the same part to just one of them whenever $q \in Q_\blacktriangle^{\bar{\mathsf{c}}}$. This technical operation allows to bound $|\bigcup(M \uplus S)|$ for the pairs $\langle M, S\rangle$ considered by the decision procedure, and hence, is key to guarantee its termination.

Definition 5.29. *Let $A = \langle Q, \Sigma, F, \Delta, C\rangle$ be a $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$, and let T be a partition of labeled states of $A_\blacktriangle$. We define $\mathsf{clean}(T)$ as the partition of labeled states $\{\mathsf{fold}(P) \mid P \in T \setminus \{\{q_\blacktriangle^\ell\} \mid \ell \in \mathbb{N}^*\}\}$, where $\mathsf{fold}(P)$ is a maximal subset of P such that each two distinct $q^{\ell_1}, q^{\ell_2} \in \mathsf{fold}(P)$ satisfy $q \in Q_\blacktriangle^{\mathsf{c}}$.*

Note that we have not fixed a precise definition for fold, as any maximal subset can be chosen. We could have specified that, e.g., when a part contains multiple occurrences of a state in $q \in Q_\blacktriangle^{\bar{\mathsf{c}}}$, the fold of such part erases all of them except the one whose label is minimum in lexicographical order. However, such precision is not relevant for our inference system.

Example 5.30. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$, let T be a partition of labeled states of $A_\blacktriangle$, and let P be a part in T. We start considering $P = \{\hat{q}^{\ell_1}, \hat{q}^{\ell_2}, \check{q}^{\ell_3}, q^{\ell_4}\}$, where $\hat{q}, \check{q}$ are distinct states in $Q_\blacktriangle^{\bar{\mathsf{c}}}$, and $q \in Q_\blacktriangle^{\mathsf{c}}$. By the definition of fold, any two occurrences of a state in $Q_\blacktriangle^{\bar{\mathsf{c}}}$ have to be collapsed into just one of them. Thus, $\mathsf{fold}(P)$ erases either $\hat{q}^{\ell_1}$ or $\hat{q}^{\ell_2}$, and hence, $\mathsf{fold}(P)$ is either $\{\hat{q}^{\ell_1}, \check{q}^{\ell_3}, q^{\ell_4}\}$ or $\{\hat{q}^{\ell_2}, \check{q}^{\ell_3}, q^{\ell_4}\}$. Note that $\check{q}^{\ell_3}$ is kept since it is the single occurrence of the state $\check{q} \in Q_\blacktriangle^{\bar{\mathsf{c}}}$, and q^{ℓ_4} has a state in $Q_\blacktriangle^{\mathsf{c}}$.*

We now consider some cases dealing with the state $q_\blacktriangle$. First, if P is a part of the form $\{q_\blacktriangle^\ell\}$, then $\mathsf{fold}(P)$ *is never considered due to the definition of* clean*, and hence, $P \notin \mathsf{clean}(T)$. Now consider that P is a part of the form $\{q_\blacktriangle^{\ell_1}, q_\blacktriangle^{\ell_2}\}$. In this case, since $q_\blacktriangle \in Q_\blacktriangle^{\bar{\mathsf{c}}}$ by definition,* $\mathsf{fold}(P)$ *is either $\{q_\blacktriangle^{\ell_1}\}$ or $\{q_\blacktriangle^{\ell_2}\}$. Moreover,* $\mathsf{fold}(P) \in \mathsf{clean}(T)$*, and hence,* $\mathsf{clean}(T)$ *contains a part of the form $\{q_\blacktriangle^\ell\}$. Although one of the goals of the* clean *operation is precisely to erase the occurrences of the state $q_\blacktriangle$, this is not a contradiction since the inference system will guarantee that $q_\blacktriangle$ only appears in parts of the form $\{q_\blacktriangle^\ell\}$ before applying the* clean *operation.*

Finally, since T is a partition, any two distinct parts $P_1, P_2 \in T$ that are not of the form $\{q_\blacktriangle^\ell\}$ satisfy that $\mathsf{fold}(P_1)$ *and* $\mathsf{fold}(P_2)$ *are disjoint and parts of* $\mathsf{clean}(T)$*.*

The clean operation preserves the fitness property when the state $q_\blacktriangle$ only occurs in parts of the form $\{q_\blacktriangle^\ell\}$: the fact that there exist $\blacktriangle$-runs fitting a pair $\langle M, S\rangle$ trivially implies the existence of $\blacktriangle$-runs fitting $\langle \mathsf{clean}(M), \mathsf{clean}(S)\rangle$, and the other direction is stated in the next lemma. Let us remark that condition (b) in the statement is rather technical. It guarantees the preservation of occurrences of the state $q_\blacktriangle$ in the $\blacktriangle$-runs of the fitting, which is useful later to prove decidability of finiteness for $\mathtt{TAG}_{\not\approx_\mathcal{R}}^{\wedge}$.

Lemma 5.31. *Let A be a $\mathtt{TAG}_{\not\approx_\mathcal{R}}^{\wedge}$. Let M, S be such that $M \uplus S$ is a partition of the set $\{q_1^{\ell_1}, \dots, q_n^{\ell_n}\}$ of labeled states of $A_\blacktriangle$, and such that the state $q_\blacktriangle$ only occurs in M and S in parts of the form $\{q_\blacktriangle^\ell\}$. Let $\hat{M}$ be* $\mathsf{clean}(M)$ *and $\hat{S}$ be* $\mathsf{clean}(S)$*, where $\bigcup(\hat{M} \uplus \hat{S}) = \{\hat{q}_1^{\hat{\ell}_1}, \dots, \hat{q}_{\hat{n}}^{\hat{\ell}_{\hat{n}}}\}$. Let $\hat{r}_1, \dots, \hat{r}_{\hat{n}}$ be $\blacktriangle$-runs of $A_\blacktriangle$ fitting $\langle \hat{M}, \hat{S}\rangle$.*

Then, there exist $\blacktriangle$-runs $r_1, \dots, r_n$ of $A_\blacktriangle$ fitting $\langle M, S\rangle$ and satisfying:

(a) $\max\{\mathsf{height}(r_i) \mid i \in \{1, \dots, n\}\} = \max\{\mathsf{height}(\hat{r}_i) \mid i \in \{1, \dots, \hat{n}\}\}$,

(b) there exists a $\blacktriangle$-subrun of $r_1, \dots, r_n$ reaching $q_\blacktriangle$ if and only if $q_\blacktriangle$ occurs among $q_1, \dots, q_n$ or there exists a $\blacktriangle$-subrun of $\hat{r}_1, \dots, \hat{r}_{\hat{n}}$ reaching $q_\blacktriangle$.

Proof. Let $A_\blacktriangle$ be $\langle Q_\blacktriangle^{\mathsf{c}} \uplus Q_\blacktriangle^{\bar{\mathsf{c}}}, \Sigma_\blacktriangle, F_\blacktriangle, \Delta_\blacktriangle, C_\blacktriangle\rangle$ more explicitly written. We construct the $\blacktriangle$-runs $r_1, \dots, r_n$ fitting $\langle M, S\rangle$ as follows, where for each r_i we distinguish cases depending on whether $q_i^{\ell_i}$ occurs in $\bigcup(\hat{M} \uplus \hat{S})$ or it has been removed by clean:

(i) For each $i \in \{1, \dots, n\}$ such that $q_i^{\ell_i} \in \bigcup(\hat{M} \uplus \hat{S})$, we define r_i to be $\hat{r}_j$, where j is the index satisfying $q_i^{\ell_i} = \hat{q}_j^{\hat{\ell}_j}$. Note that the set of such r_i's is $\{\hat{r}_1, \dots, \hat{r}_{\hat{n}}\}$.

(ii) For each $i \in \{1, \dots, n\}$ such that $q_i^{\ell_i} \notin \bigcup(\hat{M} \uplus \hat{S})$, by definition of the clean operation, either (ii.1) $q_i = q_\blacktriangle$ holds or (ii.2) $q_i \in Q_\blacktriangle^{\bar{\mathsf{c}}}$ holds. In the case (ii.1), by the assumptions of the lemma, note that the element $q_i^{\ell_i}$ appears in $M \uplus S$ in a part of the form $\{q_i^{\ell_i}\}$. In this case, we define r_i to be the $\blacktriangle$-run $(\blacktriangle \to q_\blacktriangle)$. In the case (ii.2), by definition of the clean operation, there exists exactly one $j \in \{1, \dots, n\}$ satisfying $q_j^{\ell_j} \in \bigcup(\hat{M} \uplus \hat{S})$, $q_i^{\ell_i} \sim_{M \uplus S} q_j^{\ell_j}$, and $q_i = q_j$. In this case, we define r_i to be the r_j defined in (i). In both cases, r_i is $\blacktriangle$-compatible with any other $\blacktriangle$-run.

The fact that the $\blacktriangle$-runs $r_1, \dots, r_n$ fit $\langle M, S\rangle$ trivially follows from the fact that $\hat{r}_1, \dots, \hat{r}_{\hat{n}}$ fit $\langle \hat{M}, \hat{S}\rangle$ and the conditions on the definitions done in (ii). Condition (a) of the statement trivially follows from the facts that $r_1, \dots, r_n$ include all $\hat{r}_1, \dots, \hat{r}_{\hat{n}}$, that the definitions of the r_i's done in (i) and (ii.2) preserve the maximum height, and that the definition done in (ii.1) has height 0, i.e., minimum height. Condition (b) of the statement holds since $r_1, \ldots, r_n$ include all $\hat{r}_1, \ldots, \hat{r}_{\hat{n}}$, the construction done in (ii.1) introduces the $\blacktriangle$-run $(\blacktriangle \to q_\blacktriangle)$ if and only if there exists $i \in \{1, \ldots, n\}$ such that $q_i = q_\blacktriangle$, and (ii.2) only replicates $\blacktriangle$-runs among $\hat{r}_1, \ldots, \hat{r}_{\hat{n}}$. ■

Our inference system uses the rule $\mathfrak{R}$ in Definition 5.32 to non-deterministically construct $\blacktriangle$-runs. As commented at the beginning of this section, the construction proceeds top-down and prioritizes the expansion of the $\blacktriangle$-subruns that are guessed to be maximal in height. In our formalism, this corresponds to guess a rule reaching each of the labeled states in M, replace such states by the states occurring in the left-hand side of the guessed rules, and leave the labeled states in S unchanged (condition (a) in the application of $\mathfrak{R}$). The resulting set of labeled states is then non-deterministically partitioned (also condition (a)) satisfying the following properties:

- Two labeled states $q_1^{\ell_1}, q_2^{\ell_2}$ in the same part stay in the same part when they are in S (condition (c), left-to-right direction). Otherwise, if they are in the same part of M, rules having q_1 and q_2 as right-hand sides and with the same alphabet symbol are guessed. Moreover, the corresponding states in the left-hand sides of the guessed rules are placed in the same parts (condition (b), left-to-right direction). Eventually, in both cases, two $\blacktriangle$-compatible $\blacktriangle$-runs reaching states q_1 and q_2 and recognizing the same term (with respect to $=_\blacktriangle$) will be generated.

  The right-to-left direction of conditions (b) and (c) guarantee that such treatment is only given to labeled states belonging to the same part.

- Labeled states $q_1^{\ell_1}, q_2^{\ell_2}$ are placed in different parts whenever $C_\blacktriangle$ contains the atom $q_1 \not\approx q_2$ (condition (d)), in order to guarantee that $C_\blacktriangle$ is satisfied.

- Since each labeled state q^ℓ in M must be reached by a term of maximal height, at least one state in the left-hand side of the rule guessed for q^ℓ must also be reached by a term of maximal height (condition (e)).

Definition 5.32. *Let $A = \langle Q, \Sigma, F, \Delta, C\rangle$ be a $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$. Let M, S be such that $M \uplus S$ is a partition of labeled states of $A_\blacktriangle$, and each two distinct $q_i^{\ell_i}, q_j^{\ell_j} \in \bigcup(M \uplus S)$ satisfy that $\ell_i \parallel \ell_j$. Let $\bigcup M$ be more explicitly written as $\{q_1^{\ell_1}, \ldots, q_n^{\ell_n}\}$. We define the non-deterministic* inference rule $\mathfrak{R}$ *as follows:*

$$\mathfrak{R} : \frac{\langle M, S\rangle}{\langle \mathsf{clean}(M'), \mathsf{clean}(S')\rangle}$$

where a rule $(f_i(q_{i,1}, \ldots, q_{i,m_i}) \to q_i) \in \Delta_\blacktriangle$ is guessed for each $i \in \{1, \ldots, n\}$, and M', S' are guessed satisfying:

(a) $M' \uplus S'$ is a partition of $\{q_{i,k}^{\ell_i.k} \mid i \in \{1, \ldots, n\},\, k \in \{1, \ldots, m_i\}\} \uplus \bigcup S$,

(b) $\left(q_i^{\ell_i} \sim_M q_j^{\ell_j}\right) \Leftrightarrow \left(f_i = f_j \,\wedge\, \forall k \in \{1, \ldots, m_i\} : q_{i,k}^{\ell_i.k} \sim_{M' \uplus S'} q_{j,k}^{\ell_j.k}\right)$ for each different $q_i^{\ell_i}, q_j^{\ell_j} \in \bigcup M$,

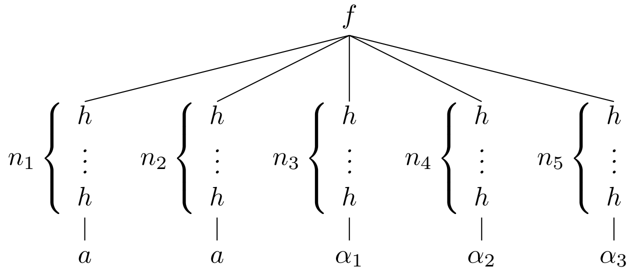
(c) $\left(\hat{q}^{\hat{\ell}} \sim_S \check{q}^{\check{\ell}}\right) \Leftrightarrow \left(\hat{q}^{\hat{\ell}} \sim_{M' \uplus S'} \check{q}^{\check{\ell}}\right)$ *for each different* $\hat{q}^{\hat{\ell}}, \check{q}^{\check{\ell}} \in \bigcup S$,

(d) $\left(\hat{q}^{\hat{\ell}} \not\sim_{M' \uplus S'} \check{q}^{\check{\ell}}\right)$ *for each different* $\hat{q}^{\hat{\ell}}, \check{q}^{\check{\ell}} \in \bigcup(M' \uplus S')$ *such that one of* $\hat{q}, \check{q}$ *is* $q_\blacktriangle$ *or* $C_\blacktriangle$ *contains the atom* $\hat{q} \not\approx \check{q}$,

(e) either $\bigcup(M' \uplus S') = \emptyset$, *or for each* $i \in \{1, \ldots, n\}$ *there is some* $k \in \{1, \ldots, m_i\}$ *such that* $\left(q_{i,k} \neq q_\blacktriangle \ \wedge \ q_{i,k}^{\ell_i.k} \in \bigcup M'\right)$.

We denote by $\rightarrow_{\mathfrak{R}}$ the derivation relation between pairs of partitions of labeled states. As usual, $\rightarrow_{\mathfrak{R}}^{+}$ denotes its transitive closure and $\rightarrow_{\mathfrak{R}}^{*}$ its reflexive-transitive closure. By abuse of notation, $\langle M, S\rangle \rightarrow_{\mathfrak{R}}^{+} \langle M', S'\rangle$ and $\langle M, S\rangle \rightarrow_{\mathfrak{R}}^{*} \langle M', S'\rangle$ are also used to denote concrete derivations with $\mathfrak{R}$ from $\langle M, S\rangle$ to $\langle M', S'\rangle$, having at least one derivation step in the case of $\rightarrow_{\mathfrak{R}}^{+}$ and with any number of derivation steps in the case of $\rightarrow_{\mathfrak{R}}^{*}$. The length of a derivation $\langle M, S\rangle \rightarrow_{\mathfrak{R}}^{*} \langle M', S'\rangle$ is its number of steps, and is denoted as $|\langle M, S\rangle \rightarrow_{\mathfrak{R}}^{*} \langle M', S'\rangle|$. Finally, to make explicit the guesses M', S' done by $\mathfrak{R}$, we use the notation $\langle M, S\rangle \rightarrow_{\mathfrak{R}} \langle \mathsf{clean}(M'), \mathsf{clean}(S')\rangle$ or $\langle M, S\rangle \rightarrow_{\mathfrak{R}}^{+} \langle \mathsf{clean}(M'), \mathsf{clean}(S')\rangle$, where the latter denotes the guess done at the last derivation step.

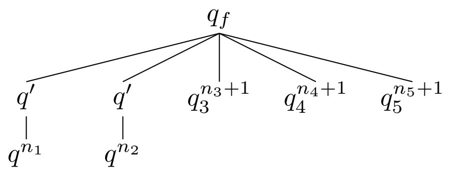
Example 5.33. *Let* Σ *be the signature* $\{a{:}0,\ b{:}0,\ h{:}1,\ f{:}5\}$, *and consider the language over* Σ *of the terms of the form:*



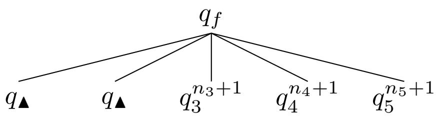
where the n_i*'s are natural numbers such that* $n_1 \neq n_2$, *and the* α_j*'s are symbols in* $\{a, b\}$ *such that* $\alpha_1 \neq \alpha_2 \neq \alpha_3$. *Note that this last condition implies that* $\alpha_1, \alpha_2, \alpha_3$ *may either be* a, b, a *or* b, a, b, *respectively. This language is recognized by the* $\mathtt{TAG}_{\not\approx_{\mathcal{R}}}^{\wedge}$ $A = \langle\{q_f, q, q', q_3, q_4, q_5\}, \Sigma, \{q_f\}, \Delta, C\rangle$, *where:*

- C *is the constraint* $\{q' \not\approx q'\} \uplus \{q_i \not\approx q_i \mid i \in \{3, \ldots, 5\}\} \uplus \{q_3 \not\approx q_4,\ q_4 \not\approx q_5\}$,
- Δ *is the set* $\{f(q', q', q_3, q_4, q_5) \rightarrow q_f\} \uplus \{a \rightarrow q,\ a \rightarrow q',\ h(q) \rightarrow q,\ h(q) \rightarrow q'\} \uplus \{a \rightarrow q_i,\ b \rightarrow q_i,\ h(q_i) \rightarrow q_i \mid i \in \{3, \ldots, 5\}\}$.

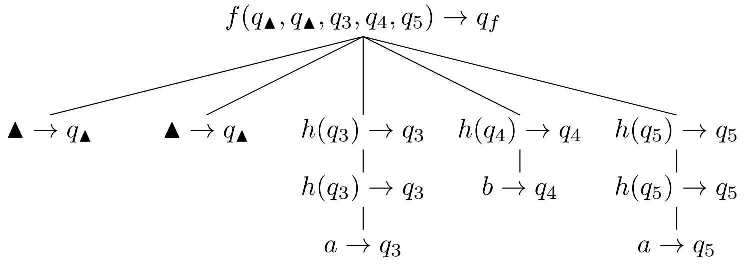
Note that the accepting runs of A *are of the following form, where only the right-hand side of the rules is depicted and numeric exponents are used to denote unary trees of a specific height within the run:*



Since $Q_A^\infty = \{q, q'\}$, the accepting $\blacktriangle$-runs of $A_\blacktriangle$ are of the following form, depicting them with the same simplification as above:



where q_f, q_3, q_4, q_5 are states in $Q_\blacktriangle^{\mathsf{c}}$, for which the label c *has been omitted to ease the presentation. Consider the following accepting $\blacktriangle$-run r:*


$f(q_\blacktriangle, q_\blacktriangle, q_3, q_4, q_5) \to q_f$
$\blacktriangle \to q_\blacktriangle$
$\blacktriangle \to q_\blacktriangle$
$h(q_3) \to q_3$
$h(q_4) \to q_4$
$h(q_5) \to q_5$
$h(q_3) \to q_3$
$b \to q_4$
$h(q_5) \to q_5$
$a \to q_3$
$a \to q_5$

The following derivation with $\mathfrak{R}$ implicitly constructs the previous $\blacktriangle$-run r:

$$
\begin{aligned}
&\langle \{\{q_f^\lambda\}\}, \emptyset \rangle \\
\to_{\mathfrak{R}}\ &\langle \{\{q_3^3, q_5^5\}\}, \{\{q_4^4\}\} \rangle \\
\to_{\mathfrak{R}}\ &\langle \{\{q_3^{3.1}, q_5^{5.1}\}, \{q_4^4\}\}, \emptyset \rangle \\
\to_{\mathfrak{R}}\ &\langle \{\{q_3^{3.1.1}, q_5^{5.1.1}\}, \{q_4^{4.1}\}\}, \emptyset \rangle \\
\to_{\mathfrak{R}}\ &\langle \emptyset, \emptyset \rangle
\end{aligned}
$$

The derivation starts from the final state q_f. In the first step, the transition rule $f(q_\blacktriangle, q_\blacktriangle, q_3, q_4, q_5) \to q_f$ is guessed, and the elements of the form $q_\blacktriangle^\ell$ are removed by the clean *operation. Moreover, since $q_3 \not\approx q_4$ and $q_4 \not\approx q_5$ occur in the constraint, q_3^3 and q_5^5 have to be placed in a different part than q_4^4 (see condition (d) in the application of $\mathfrak{R}$). In this derivation, the terms that correspond to q_3^3 and q_5^5 have been guessed to be equal with respect to $=_\blacktriangle$, and for this reason, they are placed in the same part. Moreover, the terms that correspond to q_3^3 and q_5^5 have been guessed to be higher than the term that corresponds to q_4^4. Checking that the remaining derivation steps are correct is analogous. As a final remark, note that the $\blacktriangle$-subrun $r|_\lambda$ fits the starting pair of the derivation, that $r|_3$, $r|_5$, $r|_4$ fit the second pair, that $r|_{3.1}$, $r|_{5.1}$, $r|_4$ fit the third pair, and finally that $r|_{3.1.1}$, $r|_{5.1.1}$, $r|_{4.1}$ fit the fourth pair of the derivation.*

The following lemma and corollary state the correctness of $\mathfrak{R}$, i.e., that a derivation of the form $\langle \{\{q_f^\lambda\}\}, \emptyset \rangle \to_{\mathfrak{R}}^* \langle \emptyset, \emptyset \rangle$, where q_f is a final state, corresponds to the existence of an accepting $\blacktriangle$-run. Properties (C_1) and (C_2) in the lemma relate the form of the derivation with the form of the $\blacktriangle$-run (in particular, with its height and occurrences of $q_\blacktriangle$). This is later useful when deciding finiteness.

Lemma 5.34. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$. Let M, S be partitions of labeled states of $A_\blacktriangle$ such that $\langle \{\{q^\lambda\}\}, \emptyset \rangle \to_{\mathfrak{R}}^* \langle M, S \rangle$, where q is a state of $A_\blacktriangle$. Let $\bigcup(M \uplus S)$ be more explicitly written as $\{q_1^{\ell_1}, \ldots, q_n^{\ell_n}\}$.*

Then, there exists a derivation d of the form $\langle M, S\rangle \rightarrow^*_{\mathfrak{R}} \langle\emptyset, \emptyset\rangle$ if and only if there exist $\blacktriangle$-runs $r_1, \dots, r_n$ of $A_\blacktriangle$ fitting $\langle M, S\rangle$. Moreover, d and $r_1, \dots, r_n$ satisfy the following conditions:

(C_1) $|d| = \max\{1 + \mathsf{height}(r_i) \mid i \in \{1, \dots, n\}\}$,

(C_2) there exists a $\blacktriangle$-subrun of $r_1, \dots, r_n$ reaching $q_\blacktriangle$ if and only if d can be written of the form $\langle M, S\rangle \rightarrow^+_{\mathfrak{R}} \langle \mathsf{clean}(M'), \mathsf{clean}(S')\rangle \rightarrow^*_{\mathfrak{R}} \langle\emptyset, \emptyset\rangle$ for some M', S' such that a part of the form $\{q^\ell_\blacktriangle\}$ occurs in $M' \uplus S'$ or $M \uplus S$.

Proof. Since $\langle M, S\rangle$ is derived from $\langle\{\{q^\lambda\}\}, \emptyset\rangle$ using $\mathfrak{R}$, either $\langle M, S\rangle = \langle\{\{q^\lambda\}\}, \emptyset\rangle$ or $\langle M, S\rangle = \langle \mathsf{clean}(M'), \mathsf{clean}(S')\rangle$, for some M', S' satisfying conditions (a) to (e) of Definition 5.32 with respect to some M'', S'' such that $\langle\{\{q^\lambda\}\}, \emptyset\rangle \rightarrow^*_{\mathfrak{R}} \langle M'', S''\rangle \rightarrow_{\mathfrak{R}} \langle M, S\rangle$. In either case, it is easy to see that that the elements occurring in M are distinct from the elements occurring in S, i.e., that $M \uplus S$ is a partition of labeled states of $A_\blacktriangle$, that the labels occurring in M, S are pairwise parallel, that $q_i^{\ell_i} \not\sim_{M \uplus S} q_j^{\ell_j}$ for each different $q_i^{\ell_i}, q_j^{\ell_j}$ such that the atom $q_i \not\approx q_j$ occurs in the global constraint of $A_\blacktriangle$, and that $M = \emptyset$ implies $S = \emptyset$. These properties trivially hold because of conditions (a), (d), and (e) in the application of $\mathfrak{R}$ and the fact that the initial $\langle\{\{q^\lambda\}\}, \emptyset\rangle$ satisfies them. Moreover, since the clean operation removes all parts of the form $\{q^\ell_\blacktriangle\}$, the presence of such a part in $M \uplus S$ necessarily implies that $q = q_\blacktriangle$ and $\langle M, S\rangle = \langle\{\{q^\lambda_\blacktriangle\}\}, \emptyset\rangle$.

After those general remarks, we prove each direction separately:

$\Rightarrow$) We use induction on $|d|$. For the base case, i.e., when the derivation has 0 steps, $M = S = \emptyset$ and the statement trivially holds (in particular, condition (C$_1$) holds since the maximum of an empty set is 0, by convention). For the inductive case, we write d more explicitly as $\langle M, S\rangle \rightarrow_{\mathfrak{R}} \langle \mathsf{clean}(\hat{M}), \mathsf{clean}(\hat{S})\rangle \rightarrow^*_{\mathfrak{R}} \langle\emptyset, \emptyset\rangle$, where $\bigcup(\hat{M} \uplus \hat{S}) = \{\hat{q}_1^{\hat{\ell}_1}, \dots, \hat{q}_{\hat{n}}^{\hat{\ell}_{\hat{n}}}\}$.

To construct the $\blacktriangle$-runs $r_1, \dots, r_n$ fitting $\langle M, S\rangle$ of the statement, we first need to obtain $\blacktriangle$-runs $\hat{r}_1, \dots, \hat{r}_{\hat{n}}$ fitting $\langle \hat{M}, \hat{S}\rangle$. As a first step, by induction hypothesis, there exist $\blacktriangle$-runs $\tilde{r}_1, \dots, \tilde{r}_{\tilde{n}}$ fitting $\langle \mathsf{clean}(\hat{M}), \mathsf{clean}(\hat{S})\rangle$ and satisfying conditions (C$_1$) and (C$_2$) for the subderivation $\langle \mathsf{clean}(\hat{M}), \mathsf{clean}(\hat{S})\rangle \rightarrow^*_{\mathfrak{R}} \langle\emptyset, \emptyset\rangle$ of length $|d| - 1$. Since condition (d) in the application of $\mathfrak{R}$ guarantees that the state $q_\blacktriangle$ only occurs in $\hat{M}$ and $\hat{S}$ in parts of the form $\{q^\ell_\blacktriangle\}$, we can apply Lemma 5.31 on $\hat{M}, \hat{S}$ and $\tilde{r}_1, \dots, \tilde{r}_{\tilde{n}}$ and conclude that there exist $\blacktriangle$-runs $\hat{r}_1, \dots, \hat{r}_{\hat{n}}$ fitting $\langle \hat{M}, \hat{S}\rangle$ and satisfying the following conditions:

- $\max\{1+\mathsf{height}(\hat{r}_i) \mid i \in \{1, \dots, \hat{n}\}\} = |d|-1$, by the fact that condition (C$_1$) is satisfied for $\tilde{r}_1, \dots, \tilde{r}_{\tilde{n}}$ and by property (a) of Lemma 5.31,
- there exists a $\blacktriangle$-subrun of $\hat{r}_1, \dots, \hat{r}_{\hat{n}}$ reaching $q_\blacktriangle$ if and only if a part of the form $\{q^\ell_\blacktriangle\}$ is in $\hat{M} \uplus \hat{S}$ or there exists a $\blacktriangle$-subrun of $\tilde{r}_1, \dots, \tilde{r}_{\tilde{n}}$ reaching $q_\blacktriangle$, by property (b) of Lemma 5.31.

We now construct $r_1, \dots, r_n$ from $\hat{r}_1, \dots, \hat{r}_{\hat{n}}$ and the guesses done in the application of $\mathfrak{R}$. For each $i \in \{1, \dots, n\}$ such that $q_i^{\ell_i} \in \bigcup S$, by condition (a) of $\mathfrak{R}$ it follows that there is $j \in \{1, \dots, \hat{n}\}$ such that $q_i^{\ell_i} = \hat{q}_j^{\hat{\ell}_j}$. In this case we define r_i as $\hat{r}_j$. For each $i \in \{1,\dots,n\}$ such that $q_i^{\ell_i} \in \bigcup M$, we proceed as follows. Let $f_i(q_{i,1},\dots,q_{i,m_i}) \to q_i$ be the rule of $A_\blacktriangle$ guessed for $q_i^{\ell_i}$ in the application of $\mathfrak{R}$. Again by condition (a) of $\mathfrak{R}$, it follows that, for $k \in \{1,\dots,m_i\}$, there exist $j_k \in \{1,\dots,\hat{n}\}$ such that $q_{i,k}^{\ell_i.k} = \hat{q}_{j_k}^{\hat{\ell}_{j_k}}$. In this case we define r_i as $(f_i(q_{i,1},\dots,q_{i,m_i}) \to q_i)(\hat{r}_{j_1},\dots,\hat{r}_{j_{m_i}})$.

It remains to prove that the constructed $r_1,\dots,r_n$ fit $\langle M,S\rangle$ and satisfy conditions (C$_1$) and (C$_2$). We prove separately each of the conditions of fitness from Definition 5.27, condition (C$_1$) is proved together with (F$_2$), and (C$_2$) is trivially satisfied by construction.

- Condition (F$_1$) is satisfied by construction.
- We prove that condition (F$_2$) is satisfied distinguishing cases depending on whether $\mathsf{clean}(\hat{M})$ is empty or not.

  First assume that $\mathsf{clean}(\hat{M}) = \emptyset$. Note that $\mathsf{clean}(\hat{S}) = \emptyset$ follows from the fact that $\tilde{r}_1,\dots,\tilde{r}_{\hat{n}}$ fit $\langle \mathsf{clean}(\hat{M}), \mathsf{clean}(\hat{S})\rangle$, and hence, they satisfy condition (F$_2$). Thus, $S = \emptyset$ by condition (a) in the application of $\mathfrak{R}$. Moreover, the rules guessed for each $q_i^{\ell_i} \in \bigcup M$ in the application of $\mathfrak{R}$ are either of the form $f_i(q_\blacktriangle,\dots,q_\blacktriangle) \to q_i$ or $a_i \to q_i$, where a_i is a constant symbol. The former case is not possible by Lemma 5.21, and hence, $\mathsf{height}(r_1) = \ldots = \mathsf{height}(r_n) = 0$. Since we had $S = \emptyset$, it follows that condition (F$_2$) holds. Finally, $|d| = 1 = \max\{1 + \mathsf{height}(r_i) \mid i \in \{1,\dots,n\}\}$, thus satisfying condition (C$_1$) in this case.

  Now assume that $\mathsf{clean}(\hat{M}) \neq \emptyset$. Recall that $|d| - 1 = \max\{1 + \mathsf{height}(\hat{r}_j) \mid j \in \{1,\dots,\hat{n}\}\}$. By construction of $r_1,\dots,r_n$, condition (e) in the application of $\mathfrak{R}$, and the fact that $\hat{r}_1,\dots,\hat{r}_{\hat{n}}$ fit $\langle \hat{M},\hat{S}\rangle$, the following holds for each $q_i^{\ell_i} \in \bigcup(M \uplus S)$. If $q_i = q_\blacktriangle$, then condition (F$_2$) holds trivially. If $q_i^{\ell_i} \in \bigcup M$, then $1 + \mathsf{height}(r_i) = 1 + (|d| - 1) = |d|$. If $q_i^{\ell_i} \in \bigcup S$, then $q_i^{\ell_i} \in \bigcup(\hat{M} \uplus \hat{S})$ by condition (a) in the application of $\mathfrak{R}$, and hence, $1 + \mathsf{height}(r_i) \leq |d| - 1$. It follows that condition (F$_2$) is satisfied also in this case. Moreover, since $\bigcup M$ is not empty, $|d| = \max\{1 + \mathsf{height}(r_i) \mid i \in \{1,\dots,n\}\}$, thus satisfying condition (C$_1$) also in this case.
- We consider any two distinct labeled states $q_i^{\ell_i}, q_j^{\ell_j} \in \bigcup(M \uplus S)$ in order to see that condition (F$_3$) is satisfied. First, assume that both of them are in $\bigcup M$. In this case, $\mathsf{term}(r_i) =_\blacktriangle \mathsf{term}(r_j)$ if and only if $q_i^{\ell_i} \sim_M q_j^{\ell_j}$ holds by the fact that $\hat{r}_1,\dots,\hat{r}_{\hat{n}}$ fit $\langle \hat{M},\hat{S}\rangle$ and condition (b) in the application of $\mathfrak{R}$. Second, assume that both of them are in $\bigcup S$. In this case, condition (F$_3$) holds by the fact that $\hat{r}_1,\dots,\hat{r}_{\hat{n}}$ fit $\langle \hat{M},\hat{S}\rangle$ and condition (c) in the application of $\mathfrak{R}$. Third, assume that one of them is in $\bigcup M$ and the other one is in $\bigcup S$. In this case, $\mathsf{term}(r_i) \neq_\blacktriangle \mathsf{term}(r_j)$ since $r_1,\dots,r_n$ satisfy condition (F$_2$).
- We consider any two distinct labeled states $q_i^{\ell_i}, q_j^{\ell_j} \in \bigcup(M \uplus S)$ in order to see that condition (F$_4$) is satisfied. First, assume that both of them are in $\bigcup M$. Note that any two strict $\blacktriangle$-subruns of r_i and r_j are $\blacktriangle$-compatible since $\hat{r}_1,\dots,\hat{r}_{\hat{n}}$ fit $\langle \hat{M},\hat{S}\rangle$. If the atom $q_i \not\approx q_j$ does not occur in the

global constraint of $A_\blacktriangle$, then r_i and r_j are trivially $\blacktriangle$-compatible. Otherwise, if the atom $q_i \not\approx q_j$ occurs in the global constraint of $A_\blacktriangle$, then $q_i^{\ell_i} \not\sim_M q_j^{\ell_j}$ holds by condition (d) in the application of $\mathfrak{R}$ in the derivation $\langle\{\{q^\lambda\}\},\emptyset\rangle \to_{\mathfrak{R}}^* \langle M,S\rangle$, and hence, $\mathsf{term}(r_i) \neq_\blacktriangle \mathsf{term}(r_j)$ since condition (F$_3$) is satisfied, thus implying that r_i and r_j are $\blacktriangle$-compatible. Second, assume that $q_i^{\ell_i} \in \bigcup S$. In this case, r_i is $\blacktriangle$-compatible with any other r_j by the fact that $\hat{r}_1,\ldots,\hat{r}_{\hat{n}}$ fit $\langle\hat{M},\hat{S}\rangle$ and $r_1,\ldots,r_n$ satisfy condition (F$_2$). The case $q_j^{\ell_j} \in \bigcup S$ is analogous.

$\Leftarrow$) We assume that there exist $\blacktriangle$-runs $r_1,\ldots,r_n$ fitting $\langle M,S\rangle$, and use induction on the value $h = \max\{\mathsf{height}(r_j) \mid j \in \{1,\ldots,n\}\}$ to prove $\langle M,S\rangle \to_{\mathfrak{R}}^* \langle\emptyset,\emptyset\rangle$ and conditions (C$_1$) and (C$_2$). For the base case, assume that $h=0$. If $n=0$, then the statement trivially holds, because $M=S=\emptyset$ and $\max\{1+\mathsf{height}(r_i) \mid i \in \{1,\ldots,n\}\} = \max\emptyset = 0$ by convention. Otherwise, when $n>0$, each $\blacktriangle$-run r_i is of the form $(f_i \to q_i)$, and $S=\emptyset$ by condition (F$_2$) of fitting. Consider the case in which the rule guessed for each $q_i^{\ell_i} \in \bigcup M$ in the application of $\mathfrak{R}$ is precisely $f_i \to q_i$. Note that condition (b) in the application of $\mathfrak{R}$ holds since $r_1,\ldots,r_n$ fit $\langle M,S\rangle$, and the remaining conditions of $\mathfrak{R}$ are trivially satisfied. Therefore, by defining d as $\langle M,S\rangle \to_{\mathfrak{R}} \langle\emptyset,\emptyset\rangle$, condition (C$_1$) is satisfied since $|d| = 1 = \max\{1+\mathsf{height}(r_i) \mid i \in \{1,\ldots,n\}\}$, and condition (C$_2$) trivially holds.

For the inductive case, i.e., $h>0$, we construct $\hat{M},\hat{S}$ satisfying that $\langle M,S\rangle \to_{\mathfrak{R}} \langle\mathsf{clean}(\hat{M}),\mathsf{clean}(\hat{S})\rangle$ and show that there are $\blacktriangle$-runs fitting $\langle\mathsf{clean}(\hat{M}),\mathsf{clean}(\hat{S})\rangle$ with height strictly smaller than h. Consider that the rule guessed for each $q_i^{\ell_i} \in \bigcup M$ in the application of $\mathfrak{R}$ is precisely $r_i(\lambda) = (f_i(q_{i,1},\ldots,q_{i,m_i}) \to q_i)$. Assume that $\hat{M}$ and $\hat{S}$ are guessed satisfying the following conditions, where we denote an element $q_i^{\ell_i} \in \bigcup S$ as $q_{i,\lambda}^{\ell_i.\lambda}$ in order to simplify the presentation:

(i) $\hat{M}$ is a partition of the set $\{q_{i,j}^{\ell_i.j} \mid i \in \{1,\ldots,n\},\ j \in \mathsf{Pos}(r_i),\ |j| \leq 1,\ q_{i,j} = \mathsf{rhs}(r_i(j)),\ \mathsf{height}(r_i|_j) = h-1\}$,

(ii) $\hat{S}$ is a partition of the set $\{q_{i,j}^{\ell_i.j} \mid i \in \{1,\ldots,n\},\ j \in \mathsf{Pos}(r_i),\ |j| \leq 1,\ q_{i,j} = \mathsf{rhs}(r_i(j)),\ \mathsf{height}(r_i|_j) < h-1,\ (\mathsf{height}(r_i) < h \Rightarrow j = \lambda)\}$,

(iii) $\left(q_{i_1,j_1}^{\ell_{i_1}.j_1} \sim_{\hat{M}\uplus\hat{S}} q_{i_2,j_2}^{\ell_{i_2}.j_2}\right) \Leftrightarrow (\mathsf{term}(r_{i_1}|_{j_1}) =_\blacktriangle \mathsf{term}(r_{i_2}|_{j_2}))$, for each different $q_{i_1,j_1}^{\ell_{i_1}.j_1}, q_{i_2,j_2}^{\ell_{i_2}.j_2} \in \bigcup(\hat{M}\uplus\hat{S})$.

We now prove that $\langle\mathsf{clean}(\hat{M}),\mathsf{clean}(\hat{S})\rangle$ can be derived from $\langle M,S\rangle$ with $\mathfrak{R}$ using the considered guesses. Condition (a) in the application of $\mathfrak{R}$ is trivially satisfied by conditions (i) and (ii) in the definition of $\hat{M}$ and $\hat{S}$. Condition (b) follows from the fact that $r_1,\ldots,r_n$ fit $\langle M,S\rangle$, the selections of the rules, and condition (iii). Condition (c) follows from the fact that $r_1,\ldots,r_n$ fit $\langle M,S\rangle$, and condition (iii). In order to see that condition (d) holds, first note that, for each different $q_{i_1,j_1}^{\ell_{i_1}.j_1}, q_{i_2,j_2}^{\ell_{i_2}.j_2} \in \bigcup(\hat{M}\uplus\hat{S})$ such that the atom $q_{i_1,j_1} \not\approx q_{i_2,j_2}$ occurs in the global constraint of $A_\blacktriangle$, necessarily $\mathsf{term}(r_{i_1}|_{j_1}) \neq_\blacktriangle \mathsf{term}(r_{i_2}|_{j_2})$ since $r_1,\ldots,r_n$ fit $\langle M,S\rangle$, and thus, r_{i_1}, r_{i_2} are $\blacktriangle$-compatible. Hence, $q_{i_1,j_1}^{\ell_{i_1}.j_1} \not\sim_{\hat{M}\uplus\hat{S}} q_{i_2,j_2}^{\ell_{i_2}.j_2}$ follows from condition (iii). The other case of condition (d), i.e., when q_{i_1,j_1} or q_{i_2,j_2} is $q_\blacktriangle$, also holds by condition (iii). Finally, condition (e) follows from Lemma 5.21 and condition (i). Altogether implies that $\langle M, S\rangle \rightarrow_{\mathfrak{R}} \langle \mathsf{clean}(\hat{M}), \mathsf{clean}(\hat{S})\rangle$.

By the definition of $\hat{M}, \hat{S}$ and the clean operation, $\langle \mathsf{clean}(\hat{M}), \mathsf{clean}(\hat{S})\rangle$ is fitted by the $r_i|_j$'s such that $q_{i,j}^{\ell_i.j} \in \bigcup(\mathsf{clean}(\hat{M}) \uplus \mathsf{clean}(\hat{S}))$. Moreover, the maximum height of such $r_i|_j$'s is $h-1$ by conditions (i) and (ii), and the fact that $\bigcup \hat{M}$ is not empty. Thus, we can apply induction hypothesis and conclude that $\langle \mathsf{clean}(\hat{M}), \mathsf{clean}(\hat{S})\rangle \rightarrow^*_{\mathfrak{R}} \langle \emptyset, \emptyset\rangle$ satisfying conditions (C$_1$) and (C$_2$) for the $\blacktriangle$-runs $r_i|_j$'s such that $q_{i,j}^{\ell_i.j} \in \bigcup(\mathsf{clean}(\hat{M}) \uplus \mathsf{clean}(\hat{S}))$. Hence, the derivation d exists, condition (C$_1$) is satisfied since $|d| = 1 + |\langle \mathsf{clean}(\hat{M}), \mathsf{clean}(\hat{S})\rangle \rightarrow^*_{\mathfrak{R}} \langle \emptyset, \emptyset\rangle| = 1 + h = \max\{1 + \mathsf{height}(r_i) \mid i \in \{1, \ldots, n\}\}$, and condition (C$_2$) holds by construction of $\hat{M}, \hat{S}$. ■

Corollary 5.35. *Let A be a $\mathtt{TAG}^{\wedge}_{\not\approx\mathcal{R}}$. $\mathcal{L}(A)$ is not empty if and only if there exists a derivation of the form $\langle\{\{q_f^\lambda\}\}, \emptyset\rangle \rightarrow^*_{\mathfrak{R}} \langle \emptyset, \emptyset\rangle$, where q_f is a final state of $A_\blacktriangle$.*

Proof. Follows by Lemmas 5.23 and 5.34. ■

Lemma 5.36. *Let $A = \langle Q, \Sigma, F, \Delta, C\rangle$ be a $\mathtt{TAG}^{\wedge}_{\not\approx\mathcal{R}}$. Let q be a state of $A_\blacktriangle$ such that $\langle\{\{q^\lambda\}\}, \emptyset\rangle \rightarrow^*_{\mathfrak{R}} \langle M, S\rangle \rightarrow^*_{\mathfrak{R}} \langle \emptyset, \emptyset\rangle$.*

Then, $|\bigcup(M \uplus S)| \leq 2 \cdot |Q| \cdot |\Sigma|^{\mathsf{maxar}^{|Q|}}$.

Proof. We assume that $q \neq q_\blacktriangle$, since the case where $q = q_\blacktriangle$ follows trivially. Let $\bigcup(M \uplus S)$ be more explicitly written as $\{q_1^{\ell_1}, \ldots, q_n^{\ell_n}\}$, assuming without loss of generality that the states $q_1, \ldots, q_k$ are sorted satisfying that $q_1, \ldots, q_k \in Q_\blacktriangle^{\bar{\mathsf{c}}}$ and $q_{k+1}, \ldots, q_n \in Q_\blacktriangle^{\mathsf{c}}$. Note that the cases $k = 0$ and $k = n$ are possible. Let $r_1, \ldots, r_n$ be $\blacktriangle$-runs of $A_\blacktriangle$ fitting $\langle M, S\rangle$, which are guaranteed to exist by Lemma 5.34.

First, consider the $\blacktriangle$-runs $r_1, \ldots, r_k$ reaching the states $q_1, \ldots, q_k \in Q_\blacktriangle^{\bar{\mathsf{c}}}$, respectively. We write those states more explicitly as $\bar{q}_1^{\bar{\mathsf{c}}}, \ldots, \bar{q}_k^{\bar{\mathsf{c}}}$, respectively. Note that the state $q_\blacktriangle$ does not occur in any of such r_i, since otherwise $\bar{q}_i \in Q_A^\infty$ follows by Lemmas 5.14 and 5.15, and Definition 5.19, implying that $\bar{q}_i^{\bar{\mathsf{c}}}$ is useless by the definition of $\Delta_\blacktriangle$, a contradiction. Moreover, $\mathsf{height}(r_i) < |Q|$, since otherwise r_i can be pumped, implying again that $\bar{q}_i \in Q_A^\infty$ by Lemmas 5.14 and 5.15, and Definition 5.19, and leading to a contradiction. Hence, since the number of different terms of height h is bounded by $|\Sigma|^{\mathsf{maxar}^{h+1}}$, it follows that $|\Sigma|^{\mathsf{maxar}^{|Q|}}$ bounds the number of different parts in $M \uplus S$ where the labeled states $q_1^{\ell_1}, \ldots, q_k^{\ell_k}$ occur. Finally, since the definition of clean guarantees that each part in $M \uplus S$ may contain at most $|Q|$ occurrences of states in $Q_\blacktriangle^{\bar{\mathsf{c}}}$, it follows that $k \leq |Q| \cdot |\Sigma|^{\mathsf{maxar}^{|Q|}}$.

Now, consider the $\blacktriangle$-runs $r_{k+1}, \ldots, r_n$ reaching the states $q_{k+1}, \ldots, q_n \in Q_\blacktriangle^{\mathsf{c}}$, respectively. These runs are pairwise $\blacktriangle$-compatible, since $r_1, \ldots, r_n$ fit $\langle M, S\rangle$. Hence, we can apply Lemma 5.25 and conclude that $n - k \leq |Q| \cdot |\Sigma|^{\mathsf{maxar}^{|Q|}}$.

In summary, $|\bigcup(M \uplus S)| = n = k + (n-k) \leq 2 \cdot |Q| \cdot |\Sigma|^{\mathsf{maxar}^{|Q|}}$. ■

We are finally ready to tackle the emptiness problem for $\mathtt{TAG}^{\wedge}_{\not\approx\mathcal{R}}$. To ease the presentation, from now on we assume that two pairs of partitions of labeled states $\langle M, S\rangle$ and $\langle M', S'\rangle$ are *equivalent*, denoted $\langle M, S\rangle \equiv \langle M', S'\rangle$, if they are equal up to renaming of the labels.

Theorem 5.37. *Emptiness of the language recognized by a* $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ *A can be decided in triple exponential time.*

Proof. Let A be $\langle Q, \Sigma, F, \Delta, C\rangle$. By Corollary 5.35, emptiness of $\mathcal{L}(A)$ can be reduced to the existence of a derivation of the form $\langle\{\{q_f^\lambda\}\}, \emptyset\rangle \to^*_{\mathfrak{R}} \langle\emptyset, \emptyset\rangle$, where q_f is a final state of $A_\blacktriangle$. Recall that $A_\blacktriangle$ can be computed with time in $\mathcal{O}(|\Delta| \cdot 2^{\mathsf{maxar}} + |Q| + |C|)$. Note that we do not have to consider derivations containing a subderivation of the form $\langle M, S\rangle \to^+_{\mathfrak{R}} \langle M', S'\rangle$, with $\langle M, S\rangle \equiv \langle M', S'\rangle$, since the existence of a derivation $\langle M', S'\rangle \to^*_{\mathfrak{R}} \langle\emptyset, \emptyset\rangle$ implies the existence of a derivation $\langle M, S\rangle \to^*_{\mathfrak{R}} \langle\emptyset, \emptyset\rangle$ of the same length. Intuitively, this corresponds to ignore cyclic subderivations in an alternative setting where the pairs $\langle M, S\rangle$ do not contain labels, and instead, M and S are defined as partitions of a *multiset* of states of $A_\blacktriangle$. By Lemma 5.36, the pairs $\langle M, S\rangle$ that have to be considered satisfy that $|\bigcup(M \uplus S)| \leq 2 \cdot |Q| \cdot |\Sigma|^{\mathsf{maxar}^{|Q|}}$. In the interpretation with multisets, such bound implies that there exists a multiset U of states of $A_\blacktriangle$ whose cardinal is in $2^{2^{\mathcal{O}(\log(\log(|\Sigma|)\cdot \mathsf{maxar})\cdot|Q|)}}$ and satisfying that, for any of the pairs $\langle M, S\rangle$ to be considered, $M \uplus S$ is a partition of a subset of U. Also note that, still in such interpretation, the number of partitions of subsets of U is in $2^{\mathcal{O}(|U|\cdot\log(|U|))}$. By the previous facts and by the observations in the alternative setting of multisets, it is easy to see that the total number of non-equivalent pairs that have to be considered in the derivations with $\mathfrak{R}$ is triple exponential. ■

Note that it can be derived from our arguments that there exists a triple exponential upper bound for the height of a minimal accepting ▲-run. The traditional approach to decide emptiness consists in generating all terms with height smaller than the bound, and checking whether one of them is accepted by the given automaton. However, this approach would lead to an algorithm with cost doubly exponential with respect to the bound for the height.

In order to conclude, we tackle the finiteness problem for $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$. The following definition and its corresponding lemma show how derivations with $\mathfrak{R}$ relate to the finiteness of the recognized language.

Definition 5.38. *Let A be a* $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$. *A final state q_f of $A_\blacktriangle$ is said to be a* witness of infiniteness *if it satisfies one of the following conditions:*

(W_1) The state q_f is $q_\blacktriangle$.

(W_2) There is a derivation of the form $\langle\{\{q_f^\lambda\}\}, \emptyset\rangle \to^+_{\mathfrak{R}} \langle\mathsf{clean}(M), \mathsf{clean}(S)\rangle \to^*_{\mathfrak{R}} \langle\emptyset, \emptyset\rangle$ *such that a part of the form $\{q_\blacktriangle^\ell\}$ occurs in $M \uplus S$.*

(W_3) There is a derivation of the form $\langle\{\{q_f^\lambda\}\}, \emptyset\rangle \to^*_{\mathfrak{R}} \langle M, S\rangle \to^+_{\mathfrak{R}} \langle M', S'\rangle \to^*_{\mathfrak{R}} \langle\emptyset, \emptyset\rangle$ *such that $\langle M, S\rangle \equiv \langle M', S'\rangle$.*

Lemma 5.39. *Let A be a* $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$. *$\mathcal{L}(A)$ is infinite if and only if there exists a final state q_f of $A_\blacktriangle$ such that q_f is a witness of infiniteness.*

Proof. We prove each direction separately:

$\Rightarrow$) Let q be a final state of A such that $\mathcal{L}(A, q)$ is infinite. Note that such state is guaranteed to exist by the assumption. We consider distinct cases for q, and for each of them prove the existence of a witness.

First, assume that $q \in Q_A^\infty$. In this case, $q_\blacktriangle$ is a final state of $A_\blacktriangle$ and condition (W_1) trivially holds for $q_f := q_\blacktriangle$.

Second, assume that $q \notin Q_A^\infty$ and that there exists an accepting run r of A reaching q and containing a state in Q_A^∞. In this case, it is easy to construct from r an accepting $\blacktriangle$-run r' of $A_\blacktriangle$ having some occurrence of the state $q_\blacktriangle$. Intuitively, it suffices to replace the subruns of r at minimal positions (with respect to the prefix relation $\leq$) that reach a state in Q_A^∞ by the $\blacktriangle$-run $(\blacktriangle \to q_\blacktriangle)$, and add c or $\bar{\mathsf{c}}$ to each state depending on whether there is a constrained state in the corresponding $\blacktriangle$-subrun. By condition (C_2) of Lemma 5.34, $q_f := \mathsf{rhs}(r'(\lambda))$ satisfies condition (W_2).

Finally, assume that $q \notin Q_A^\infty$ and that there is no run of A reaching q and involving states in Q_A^∞. In this case, since $\mathcal{L}(A, q)$ is infinite, there necessarily exist arbitrarily high accepting runs of A reaching q and involving some constrained state. It follows that there exist arbitrarily high accepting $\blacktriangle$-runs of $A_\blacktriangle$ reaching $q_f := q^{\mathsf{c}}$. Thus, by condition (C_1) of Lemma 5.34, there are arbitrarily long derivations of the form $\langle\{\{q_f^\lambda\}\}, \emptyset\rangle \to_{\mathfrak{R}}^* \langle\emptyset, \emptyset\rangle$. Since any derived pair $\langle M, S\rangle$ satisfies that $|\bigcup(M \uplus S)|$ is bounded as stated in Lemma 5.36, there exists a derivation of the form $\langle\{\{q_f^\lambda\}\}, \emptyset\rangle \to_{\mathfrak{R}}^* \langle M, S\rangle \to_{\mathfrak{R}}^+ \langle M', S'\rangle \to_{\mathfrak{R}}^* \langle\emptyset, \emptyset\rangle$, with $\langle M, S\rangle \equiv \langle M', S'\rangle$, and thus, q_f satisfies condition (W_3).

$\Leftarrow$) If q_f satisfies condition (W_1), then $(\blacktriangle \to q_\blacktriangle)$ is an accepting $\blacktriangle$-run of $A_\blacktriangle$ and the statement follows from Lemma 5.24. If q_f satisfies condition (W_2), then, by condition (C_2) of Lemma 5.34, there exists an accepting $\blacktriangle$-run of $A_\blacktriangle$ reaching q_f and containing the $\blacktriangle$-subrun $(\blacktriangle \to q_\blacktriangle)$. Hence, the statement follows again from Lemma 5.24. Finally, if q_f satisfies condition (W_3), note that the subderivation $\langle M, S\rangle \to_{\mathfrak{R}}^+ \langle M', S'\rangle$ can be pumped, and hence, we can construct arbitrarily long derivations. Thus, by condition (C_1) of Lemma 5.34, an infinite number of accepting $\blacktriangle$-runs of $A_\blacktriangle$ fitting $\langle\{\{q_f^\lambda\}\}, \emptyset\rangle$ exist. Thus, $\mathcal{L}_\blacktriangle(A_\blacktriangle)$ is infinite and the statement follows by Lemma 5.24. ■

Theorem 5.40. *Finiteness of the language recognized by a* $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$ *A can be decided in triple exponential time.*

Proof. By Lemma 5.39, infiniteness of $\mathcal{L}(A)$ can be reduced to the existence of a witness of infiniteness in $A_\blacktriangle$. Finding a witness satisfying condition (W_1) of Definition 5.38 is straightforward. The justification for the time complexity to detect a witness satisfying conditions (W_2) or (W_3) is analogous to the arguments in the proof of Theorem 5.37. The only difference is that, for condition (W_3), we need to consider derivations with at most one subderivation of the form $\langle M, S\rangle \to_{\mathfrak{R}}^+ \langle M', S'\rangle$, with $\langle M, S\rangle \equiv \langle M', S'\rangle$. This modification does not affect the time complexity, and the statement holds. ■

5.6 Unranked ordered terms

Our results on $\mathtt{TAG}^{\wedge}_{\not\approx_\mathcal{R}}$ can be generalized from ranked to unranked ordered terms following the same approach as in [Vac10, BCG+13]. In the unranked setting, terms are constructed over a given *unranked signature* Σ, i.e., over a set of symbols that do not have an associated arity. Hence, the only difference between ranked and unranked ordered terms is that, in the latter, the number of children of any position is arbitrary since it does not depend on the symbol labeling it. We denote as $\mathcal{U}(\Sigma)$ the set of unranked ordered terms over Σ, and recall from [Vac10, BCG$^+$13] the definition extending the automaton model for unranked ordered terms of [Mur99] with global constraints.

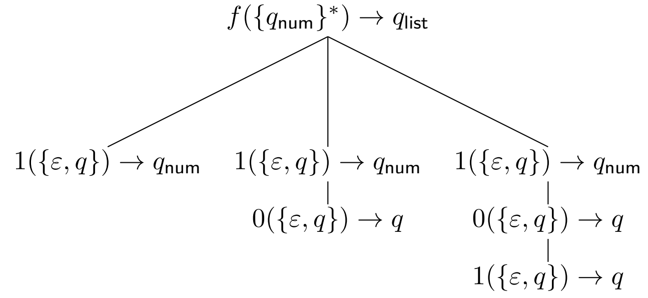
Definition 5.41. *A* hedge automaton with global constraints *over the constraint types $\tau_1, \ldots, \tau_n$, denoted $\mathtt{HAG}_{\tau_1,\ldots,\tau_n}$, is a tuple $A = \langle Q, \Sigma, F, \Delta, C\rangle$, where Q is a finite set of states, Σ is an unranked signature, $F \subseteq Q$ is the subset of final states, C is a Boolean combination of atomic constraints of types $\tau_1, \ldots, \tau_n$, and Δ is a finite 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 the alphabet Q, assumed given by a finite state automaton with input alphabet Q. Analogously to Definition 5.1, the subclass of $\mathtt{HAG}_{\tau_1,\ldots,\tau_n}$ where the global constraint is a conjunction of positive literals is denoted $\mathtt{HAG}^{\wedge}_{\tau_1,\ldots,\tau_n}$.*

The notion of run of $\mathtt{TAG}_{\tau_1,\ldots,\tau_n}$ is extended to $\mathtt{HAG}_{\tau_1,\ldots,\tau_n}$ in the natural way. A run *of A on an unranked ordered term $t \in \mathcal{U}(\Sigma)$ is a mapping $r : \mathsf{Pos}(t) \to \Delta$ satisfying that, for each position $p \in \mathsf{Pos}(t)$ with n children, if $r(p.1), \ldots, r(p.n)$ are rules with right-hand side states $q_1, \ldots, q_n$, respectively, then $r(p)$ is a rule of the form $t(p)(L) \to q$ such that the word $q_1 \cdots q_n$ belongs to L. Moreover, r satisfies the global constraint C. A run r is called* accepting *if the right-hand side state of $r(\lambda)$ is in F. By $\mathcal{L}(A)$ we denote the* language recognized *by A, that is, the set of unranked ordered terms t such that there exists an accepting run of A on t.*

Example 5.42. *Recall from Example 1.3 the language representing lists of pairwise different (encodings of) numbers: a term of the form $f(e_1, f(e_2, \ldots f(e_m, \bot) \ldots))$ is used to represent the list of numbers $e_1, \ldots, e_m$, where each of the e_i's is a term of the form $b_n(b_{n-1}(\ldots b_1(b_0(\bot))\ldots))$, with $n > 0$ and $b_0, \ldots, b_n \in \{0, 1\}$. Note that the symbol f with arity 2 is used for chaining the elements in the list, and the symbol $\bot$ with arity 0 for starting the numbers and the list. The same idea can be more naturally expressed in the unranked setting as follows. Let Σ be the unranked signature $\{0, 1, f\}$, and let L be the language of unranked ordered terms of the form $f(e_1, \ldots, e_m)$, where the e_i's are pairwise different and each of them is of the form $b_n(b_{n-1}(\ldots b_1(b_0)\ldots))$, with $n > 0$ and $b_0, \ldots, b_n \in \{0, 1\}$. Such language L can be recognized by the $\mathtt{HAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ $A = \langle \{q, q_{\mathsf{num}}, q_{\mathsf{list}}\}, \Sigma, \{q_{\mathsf{list}}\}, \Delta, q_{\mathsf{num}} \not\approx q_{\mathsf{num}}\rangle$, where the set of transition rules Δ is:*

$$\begin{aligned}
0(\{\varepsilon, q\}) &\to q\\
1(\{\varepsilon, q\}) &\to q\\
0(\{\varepsilon, q\}) &\to q_{\mathsf{num}}\\
1(\{\varepsilon, q\}) &\to q_{\mathsf{num}}\\
f(\{q_{\mathsf{num}}\}^*) &\to q_{\mathsf{list}}
\end{aligned}$$

where we have explicitly written the regular word languages on the left-hand side of the rules instead of giving a finite state automaton recognizing each of them, and use ε to denote the empty word. An unranked ordered term representing, e.g., the numbers $1, 2, 5$ is for instance $f(1, 1(0), 1(0(1)))$ (note that the encoding of a number is not *unique, as any amount of leading zeros is admitted by A), and it is recognized by A with the following accepting run.*

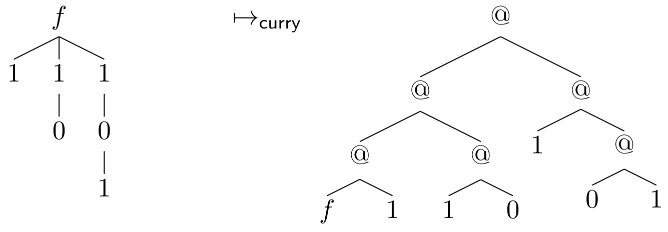

$f(\{q_{\mathsf{num}}\}^*) \to q_{\mathsf{list}}$
$1(\{\varepsilon, q\}) \to q_{\mathsf{num}}$
$1(\{\varepsilon, q\}) \to q_{\mathsf{num}}$
$0(\{\varepsilon, q\}) \to q$
$1(\{\varepsilon, q\}) \to q_{\mathsf{num}}$
$0(\{\varepsilon, q\}) \to q$
$1(\{\varepsilon, q\}) \to q$

Note that the global disequality constraint is satisfied since all the subterms reaching q_{num} are pairwise different.

In order to translate the decidability results from the ranked to the unranked setting, the *extension* encoding described in [CDG⁺07] is used in [Vac10, BCG⁺13] to reduce the question on the unranked setting to the ranked one. Such encoding allows to transform unranked ordered terms into binary ranked ordered terms; briefly: let @ be a new symbol, and let $\Sigma_@ := \{a{:}0 \mid a \in \Sigma\} \uplus \{@{:}2\}$ for any given unranked signature Σ, then $\mathsf{curry} : \mathcal{U}(\Sigma) \to \mathcal{T}(\Sigma_@)$ is a bijective function recursively defined as follows:

$$\mathsf{curry}(a(t_1, \ldots, t_n)) = \begin{cases} a & \text{if } n = 0 \\ @(\mathsf{curry}(a(t_1, \ldots, t_{n-1})), \mathsf{curry}(t_n)) & \text{otherwise} \end{cases}$$

Example 5.43. *The unranked ordered term $f(1, 1(0), 1(0(1)))$ from Example 5.42 is transformed through* curry *as follows:*



Proposition 5.44 ([Vac10, BCG⁺13])**.** *Let A be an $\mathtt{HAG}_{\approx,\not\approx,\mathbb{N}}$ with unranked signature Σ and global constraint C.*

Then, a $\mathtt{TAG}_{\approx,\not\approx,\mathbb{N}}$ A' over $\Sigma_@$ and with global constraint C can be constructed in linear time satisfying that $\mathcal{L}(A') = \{\mathsf{curry}(t) \mid t \in \mathcal{L}(A)\}$.

The linear time complexity as well as the preservation of the global constraint stated in Proposition 5.44 are not present in the original statement from [Vac10, BCG⁺13], but can easily be deduced from the construction done in the proof.

By Proposition 5.44 and Theorems 5.37 and 5.40, we reach the following result.

Corollary 5.45. *Emptiness and finiteness are decidable for $\mathtt{HAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ in triple exponential time.*

Chapter 6

Conclusions

We have tackled decidability problems on tree automata enhanced with local constraints and global constraints. An analysis of the obtained results, together with discussions on possible extensions, is presented below.

6.1 Local constraints, and tree homomorphisms

The new classes of automata introduced in Chapter 3 extend the current literature by allowing constraints that test the height of sibling subterms. The obtained time complexity to decide emptiness and finiteness is exponential for the simplest case $\mathtt{TACBB}_{\mathsf{h}}$, and double exponential for $\mathtt{TACBB}_{\mathsf{H}}$, $\mathtt{TACBB}_{\mathsf{he}}$, and $\mathtt{TACBB}_{\mathsf{He}}$. For $\mathtt{TACBB}_{\mathsf{he}}$ and $\mathtt{TACBB}_{\mathsf{He}}$ both problems are at least EXPTIME-hard due to the equality tests [CDG$^+$07], but the precise hardness for our automata is unknown and deserves further analysis. Additionally, it would also be interesting to study other extensions of these constraints. In particular, we have focused on constraints between brother positions, but emptiness and finiteness with arbitrary positions for the height constraints might also be decidable. Moreover, several classes of (dis)equality constraints with non-brother positions are known to be decidable, such as the class $\mathtt{TA}_{\not\approx}$ of automata with arbitrary local disequality constraints [CJ03], or the class of deterministic and complete reduction automata, i.e., automata with arbitrary local (dis)equality constraints but with a bound on the maximum number of equality tests that can be performed at each branch of the input term [DCC95]. Extending those models with height constraints might preserve the decidability, and should also be considered.

In Chapter 4 we have proved that the emptiness and finiteness problems for the class $\mathtt{TA}_{\mathsf{ihom},\not\approx}$ of tree automata with local constraints are decidable in exponential time. As a consequence, we have obtained EXPTIME-completeness of set inclusion, finiteness of set difference, and regularity (HOM problem) for languages defined as images of regular tree languages under tree homomorphisms. Hence, we have determined the exact complexity of HOM and other problems that were proved decidable in triple exponential time in [GG13]. To this end, we have used some intermediate results from [GG13]. It would be interesting to study whether such intermediate results can be obtained in a simpler and clearer manner using the new class $\mathtt{TA}_{\mathsf{ihom},\not\approx}$. Also, we have obtained simpler combinatoric arguments than the ones used in [CJ03] to prove decidability of emptiness for $\mathtt{TA}_{\not\approx}$. Hence, it could be interesting to study whether those proofs can be rewritten with the present approach in order to make them more accessible. In [DCC95], emptiness of deterministic and complete reduction automata is proved decidable. It could also be worth studying whether our techniques can be applied to this problem in order to improve the obtained time complexity.

6.2 Global constraints

In Chapter 5 we have presented triple exponential time algorithms for the emptiness and finiteness problems for $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$, i.e., for tree automata with global constraints where the constraint is a conjunction of atoms over the predicate $\not\approx$, and the formula defines a reflexive relation on the states occurring in it. This automaton model is a meaningful fragment of the class $\mathtt{TAG}_{\approx,\not\approx}$ from [Vac10] that is incomparable with the class $\mathtt{TAGED}$ from [FTT08, FTT10]. Our results on $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ are naturally translated to $\mathtt{HAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$, i.e., to unranked tree automata with global reflexive disequality constraints, concluding that its emptiness and finiteness problems are also in 3EXPTIME. We have not tackled the hardness of any of those problems, but such study should be a natural next step of research. Additionally, our work on $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{R}}}$ can be extended in different directions. On the one hand, several variants like adding equality constraints or removing the reflexivity condition are interesting and deserve further study. On the other hand, it might be possible to generalize the interpretation of the global disequality constraint. In particular, we believe that our results (on the ranked setting) can be extended to the case where term equality is interpreted modulo commutativity of some alphabet symbols. To this end, it seems necessary to adapt the conditions in the application of the inference rule $\mathfrak{R}$ (recall Definition 5.32). More precisely, condition (b) should establish a bijection between the respective direct children of two labeled states in the same part, ensuring that bijected children go to the same part, and hence generate the same term; moreover, for labeled states in different parts such a bijection should be impossible, or the guessed alphabet symbols f_i and f_j should differ. More general interpretations of term equality, such as the equality modulo flat equational theories of $\mathtt{TABG}_{\approx,\not\approx,\mathbb{N}}$ [BCG$^+$13], would require further refinements to the inference rule, but are also interesting and should be considered.

Many relevant problems are still open in the general setting of global constraints. In particular, emptiness for $\mathtt{TAG}_{\approx,\not\approx}$ is known to be decidable [Vac10], but with non-elementary time complexity. A challenging question would be to investigate the precise complexity of such problem, avoiding the use of Higman's Lemma in the algorithm. To this end, several partial results are known: in [FTT08, FTT10], a direct reduction into solving positive and negative set constraints [CP94, GTT94, Ste94] is used to show that emptiness is decidable in NEXPTIME for $\mathtt{TAG}^{\wedge}_{\not\approx_{\mathcal{A}}}$, and furthermore, in [Vac10], an algorithm from [Cha99] on t-dag automata is adapted to decide in NEXPTIME the emptiness problem for $\mathtt{TAG}^{\wedge}_{\approx_{\mathcal{R}},\not\approx_{\mathcal{A}}}$; on the other hand, the problem is at least EXPTIME-hard, since it is for the fragment $\mathtt{TAG}^{\wedge}_{\approx_{\mathcal{R}}}$ [FTT08]. Finally, it would be interesting to study whether the class $\texttt{TABG}_{\approx,\not\approx,\mathbb{N}}$ from [BCG$^+$13] mixing the global constraints of $\texttt{TAG}_{\approx,\not\approx,\mathbb{N}}$ with the local constraints of $\texttt{AWCBB}$ can be further extended with the height constraints introduced in Chapter 3. However, proving decidability of such extension might require a distinct approach than the one proposed in [BCG$^+$13]: height constraints force a specific ordering on (the height of) siblings, and this seems to conflict with the definition of the well quasi-ordered set in [BCG$^+$13], which is crucial in the current approach. This is an interesting problem and deserves further analysis.

Bibliography

[BCG$^+$13] Luis Barguñó, Carles Creus, Guillem Godoy, Florent Jacquemard, and Camille Vacher. Decidable classes of tree automata mixing local and global constraints modulo flat theories. *Logical Methods in Computer Science*, 9(2), 2013.
Cited on pages 9, 13, 89, 119, 120, 121, 124, and 125.

[BCH$^+$09] Véronique Benzaken, Giuseppe Castagna, Haruo Hosoya, Benjamin C. Pierce, and Stijn Vansummeren. XML typechecking. In *Encyclopedia of Database Systems*, pages 3646–3650. Springer US, 2009.
Cited on page 4.

[BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, New York, 1998.
Cited on page 15.

[BST99] Bruno Bogaert, Franck Seynhaeve, and Sophie Tison. The recognizability problem for tree automata with comparison between brothers. In *Foundations of Software Science and Computation Structures (FoSSaCS)*, pages 150–164, 1999.
Cited on page 8.

[BT92] Bruno Bogaert and Sophie Tison. Equality and disequality constraints on direct subterms in tree automata. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 161–171, 1992.
Cited on pages 6, 7, 12, and 23.

[CDG$^+$07] Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications. Available at `http://www.grappa.univ-lille3.fr/tata`, 2007.
Cited on pages 3, 12, 15, 19, 42, 99, 121, and 123.

[CG14] Carles Creus and Guillem Godoy. Tree automata with height constraints between brothers. In *Rewriting Techniques and Applications (RTA)*, pages 149–163, 2014.
Cited on page 10.

[CGG13] Carles Creus, Adrià Gascón, and Guillem Godoy. Emptiness and finiteness for tree automata with global reflexive disequality constraints. *Journal of Automated Reasoning*, 51(4):371–400, 2013.
Cited on page 12.

[CGGR12] Carles Creus, Adrià Gascón, Guillem Godoy, and Lander Ramos. The HOM problem is EXPTIME-complete. In *Logic in Computer Science (LICS)*, pages 255–264, 2012.
Cited on page 11.

[Cha99] Witold Charatonik. Automata on DAG representations of finite trees. Research Report MPI-I-1999-2-001, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, March 1999.
Cited on page 124.

[CJ03] Hubert Comon and Florent Jacquemard. Ground reducibility is EXPTIME-complete. *Information and Computation*, 187(1):123–153, 2003.
Cited on pages 6, 40, 41, 42, 51, 52, 54, 79, 123, and 124.

[CJP08] Hubert Comon-Lundh, Florent Jacquemard, and Nicolas Perrin. Visibly tree automata with memory and constraints. *Logical Methods in Computer Science*, 4(2:8), 2008.
Cited on page 11.

[CP94] Witold Charatonik and Leszek Pacholski. Set constraints with projections are in NEXPTIME. In *Foundations of Computer Science (FOCS)*, pages 642–653, 1994.
Cited on page 124.

[CSTT99] Anne-Cécile Caron, Franck Seynhaeve, Sophie Tison, and Marc Tommasi. Deciding the satisfiability of quantifier free formulae on one-step rewriting. In *Rewriting Techniques and Applications (RTA)*, pages 103–117, 1999.
Cited on page 6.

[DCC95] Max Dauchet, Anne-Cécile Caron, and Jean-Luc Coquidé. Automata for reduction properties solving. *Journal of Symbolic Computation*, 20(2):215–233, 1995.
Cited on pages 6, 123, and 124.

[Don70] John Doner. Tree acceptors and some of their applications. *Journal of Computer System Sciences*, 4:406–451, 1970.
Cited on page 4.

[DTT02] Max Dauchet, Sophie Tison, and Marc Tommasi. Réduction de la non-linéarité des morphismes d'arbres Recognizable tree-languages and non-linear morphisms. *Theoretical Computer Science*, 281(1-2):219–233,

2002.
Cited on page 8.

[Eng75] Joost Engelfriet. Bottom-up and top-down tree transformations - A comparison. *Mathematical Systems Theory*, 9(3):198–231, 1975.
Cited on page 8.

[FTT07] Emmanuel Filiot, Jean-Marc Talbot, and Sophie Tison. Satisfiability of a spatial logic with tree variables. In *Computer Science Logic (CSL)*, volume 4646 of *Lecture Notes in Computer Science*, pages 130–145, 2007.
Cited on pages 8 and 9.

[FTT08] Emmanuel Filiot, Jean-Marc Talbot, and Sophie Tison. Tree automata with global constraints. In *Developments in Language Theory (DLT)*, pages 314–326, 2008.
Cited on pages 8, 9, 13, 88, and 124.

[FTT10] Emmanuel Filiot, Jean-Marc Talbot, and Sophie Tison. Tree automata with global constraints. *International Journal of Foundations of Computer Science*, 21(4):571–596, 2010.
Cited on pages 8, 9, 13, 88, 89, 90, and 124.

[Fül94] Zoltán Fülöp. Undecidable properties of deterministic top-down tree transducers. *Theoretical Computer Science*, 134:311–328, 1994.
Cited on page 8.

[GG13] Guillem Godoy and Omer Giménez. The HOM problem is decidable. *Journal of the ACM*, 60(4):23:1–23:44, 2013.
Cited on pages 6, 7, 8, 12, 39, 40, 42, 43, 44, 45, 47, 48, 78, 80, 123, and 124.

[GGM11] Omer Giménez, Godoy Godoy, and Sebastian Maneth. Deciding regularity of the set of instances of a set of terms with regular constraints is EXPTIME-complete. *SIAM Journal on Computing*, 40(2):446–464, 2011.
Cited on pages 8, 12, and 42.

[GMT08] Guillem Godoy, Sebastian Maneth, and Sophie Tison. Classes of tree homomorphisms with decidable preservation of regularity. In *Foundations of Software Science and Computation Structures (FoSSaCS)*, pages 127–141, 2008.
Cited on page 8.

[GS84] Ferenc Gécseg and Magnus Steinby. *Tree Automata*. Akadémiai Kiadó, 1984.
Cited on page 19.

[GS97] Ferenc Gécseg and Magnus Steinby. Tree languages. In *Handbook of Formal Languages*, volume 3, pages 1–68. Springer-Verlag, 1997.
Cited on pages 3 and 19.

[GT95] Rémi Gilleron and Sophie Tison. Regular tree languages and rewrite systems. *Fundamenta Informaticae*, 24(1/2):157–174, 1995.
Cited on page 80.

[GTT94] Rémi Gilleron, Sophie Tison, and Marc Tommasi. Some new decidability results on positive and negative set constraints. In *Constraints in Computational Logics (CCL)*, volume 845 of *Lecture Notes in Computer Science*, pages 336–351, 1994.
Cited on page 124.

[HH92] Dieter Hofbauer and Maria Huber. Computing linearizations using test sets. In *Conditional Term Rewriting Systems (CTRS)*, pages 287–301, 1992.
Cited on page 8.

[HHK12] Pierre-Cyrille Héam, Vincent Hugot, and Olga Kouchnarenko. On positive TAGED with a bounded number of constraints. In *Conference on Implementation and Application of Automata (CIAA)*, pages 329–336, 2012.
Cited on page 13.

[JKV11] Florent Jacquemard, Francis Klay, and Camille Vacher. Rigid tree automata and applications. *Information and Computation*, 209(3):486–512, 2011.
Cited on pages 9, 13, and 89.

[JRV08] Florent Jacquemard, Michaël Rusinowitch, and Laurent Vigneron. Tree automata with equality constraints modulo equational theories. *Journal of Logic and Algebraic Programming*, 75(2):182–208, 2008.
Cited on page 10.

[KB70] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra*, pages 263–297, 1970.
Cited on page 56.

[KT95] Gregory Kucherov and Mohamed Tajine. Decidability of regularity and related properties of ground normal form languages. *Information and Compuation*, 118:91–100, 1995.
Cited on page 8.

[MLMK05] Makoto Murata, Dongwon Lee, Murali Mani, and Kohsuke Kawaguchi. Taxonomy of XML schema languages using formal language theory. *ACM*

Transactions of Internet Technologies, 5(4):660–704, 2005.
Cited on page 4.

[Mon81] Jocelyne 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.
Cited on page 6.

[Mur99] Makoto Murata. Hedge automata: a formal model for XML schemata. Technical report, Fuji Xerox Information Systems, 1999.
Cited on page 120.

[MW67] Jorge E. Mezei and Jesse B. Wright. Algebraic automata and context-free sets. *Information and Control*, 11:3–29, 1967.
Cited on page 4.

[Ste94] Kjartan Stefánsson. Systems of set constraints with negative constraints are NEXPTIME-complete. In *Logic in Computer Science (LICS)*, pages 137–141, 1994.
Cited on page 124.

[Tha69] James W. Thatcher. Transformations and translations from the point of view of generalized finite automata theory. In *Symposium on Theory of Computing (STOC)*, pages 129–142, 1969.
Cited on page 8.

[Tre00] Ralf Treinen. Predicate logic and tree automata with tests. In *Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 1784 of *Lecture Notes in Computer Science*, pages 329–343, 2000.
Cited on page 11.

[Vac10] Camille Vacher. *Automates d'arbres à contraintes globales pour la vérification de propriétés de sécurité.* PhD thesis, Laboratoire Spécification et Vérification, ENS Cachan, France, 2010.
Cited on pages 9, 13, 89, 90, 91, 119, 120, 121, and 124.

[VG92] Sandor Vágvölgyi and Rémi Gilleron. For a rewrite system it is decidable whether the set of irreducible, ground terms is recognizable. *Bulletin of the EATCS*, 48:197–209, 1992.
Cited on page 8.