Model Checking Constraint LTL over Trees*

Alexander Kartzow[1,2] and Thomas Weidner[1]

[1] Institut für Informatik, Universität Leipzig, Germany
[2] Department für Elektrotechnik und Informatik, Universität Siegen, Germany

Abstract Constraint automata are an adaptation of Büchi-automata that process data words where the data comes from some relational structure $\mathfrak{S}$. Every transition of such an automaton comes with constraints in terms of the relations of $\mathfrak{S}$. A transition can only be fired if the current and the next data values satisfy all constraints of this transition. These automata have been used in the setting where $\mathfrak{S}$ is a linear order for deciding constraint LTL with constraints over $\mathfrak{S}$. In this paper, $\mathfrak{S}$ is the infinitely branching infinite order tree $\mathfrak{T}$. We provide a PSPACE algorithm for emptiness of $\mathfrak{T}$-constraint automata. This result implies PSPACE-completeness of the satisfiability and the model checking problem for constraint LTL with constraints over $\mathfrak{T}$.

1 Introduction

Temporal logics like LTL or CTL* are nowadays standard languages for specifying system properties in verification. These logics are interpreted over node labelled graphs, where the node labels (also called atomic propositions) represent abstract properties of a system (for instance, a computer program). Clearly, such an abstracted system state does not in general contain all the information of the original system state. This may lead to incorrect results in model checking.

In order to overcome this weakness, extensions of temporal logics by atomic (local) constraints over some structure $\mathfrak{A}$ have been proposed (cf. [8,11]). For instance, LTL *with local constraints* is evaluated over infinite words where the letters are tuples over $\mathfrak{A}$ of a fixed size. For instance, for $\mathfrak{A} = (\mathbb{Z}, <)$, this logic is standard LTL where atomic propositions are replaced by atomic constraints of the form $\mathsf{X}^i x_j < \mathsf{X}^l x_k$. This constraint is satisfied by a path π if the j-th element of the i-th letter of π is less than the k-th element of the l-th letter of π.

While temporal logics with integer constraints are suitable to reason about programs manipulating counters, reasoning about systems manipulating pushdowns requires constraints over words over a fixed alphabet and the prefix relation (which is equivalent to constraints over an infinite k-ary tree with descendant/ancestor relations). There are numerous investigations on satisfiability and model checking for temporal logics with constraints over the integers (cf. [8,2,11,13,3,4]). Contrary, temporal logics with constraints over trees have not yet been investigated

* This work is supported by the DFG Research Training Group 1763 (QuantLA) and the DFG research project GELO.

much, although questions concerning decidability of the satisfiability problem for LTL or CTL* with such constraints have been asked for instance in [11,7]. A first (negative) result by Carapelle et al. [5] shows that a technique developed in [7,4] for satisfiability results of branching-time logics (like CTL* or ECTL*) with integer constraints cannot be used to resolve the satisfiability status of temporal logics with constraints over trees.

Our goal is to show that satisfiability of LTL with constraints over the tree is decidable. At first, we analyse the emptiness problem of $\mathfrak{T}$-constraint automata (cf. [13,10]) where $\mathfrak{T}$ is the infinitely branching infinite tree with prefix relation. These automata are Büchi-automata that process (multi-)data words where the data values are elements of $\mathfrak{T}$ where applicability of transitions depends on the order of the data values at the current and the next position. Our technical main result shows that emptiness for these automata is PSPACE-complete. Having obtained an algorithm for the emptiness problem, we can easily provide algorithms for the satisfiability and model checking problems for LTL with constraints over $\mathfrak{T}$. We exactly mimic the automata based algorithms for standard LTL of Vardi and Wolper [14] noting that the constraints in the transitions are exactly what is needed to deal with the atomic constraints in the local constraint version of LTL. It follows directly that satisfiability of LTL with constraints over $\mathfrak{T}$ and model checking models defined by constraint automata against LTL with constraints over $\mathfrak{T}$ is PSPACE-complete.

Finally, we extend our results to the case of constraints over the infinite k-ary tree for every $k \in \mathbb{N}$ by providing a reduction to LTL with constraints over $\mathfrak{T}$. Thus, satisfiability and model checking for LTL with constraints over the infinite k-ary tree is also in PSPACE.

Upon finishing our paper, we have become aware that Demri and Deters (abbreviated DD in the following) have submitted a paper [9] that shows above mentioned results on satisfiability using a reduction of constraints over trees to constraints over the integers. Even though the main results of both papers coincide, there are major differences.

1. DD's result extends to satisfiability of the corresponding version of CTL*, but DD do not consider the model checking problem.
2. DD's result holds even if the logic is enriched by length constraints that compare the lengths of the interpretations of variables. Since our approach abstracts away the concrete length of words, we cannot reprove this result. On the other hand, we can enrich the logic with constraints using the lexicographic order on the tree as well. DD's approach can not deal with this order. Thus, the logic in each paper is incomparable to the logic of the other.
3. DD conjecture that (branching-degree) uniform satisfiability problem is in PSPACE. This problem asks, given a formula and a $k \in \mathbb{N} \cup \{ \infty \}$ whether there is a model with values in the k-ary infinite tree that satisfies the formula. We confirm DD's conjecture.
4. Finally, our proof is self-contained. In contrast, DD's proof seems to be more elegant and less technical, but this comes at the cost of relying on the

decidability result for satisfiability of LTL with constraints over the integers [3], which is again quite technical to prove.[3]

Our result leaves open several further research directions. Firstly, DD's result on CTL* with constraints over trees does not yield any reasonable complexity bound because the complexity of their algorithm relies on the results of Bojańczyk and Toruńczyk [1] on weak monadic second order logic with the unbounding quantifier. Thus, without any progresses concerning the complexity of this logic, DD's approach cannot be used to obtain better bounds. In contrast, the concept of $\mathfrak{T}$-constraint automata can be easily lifted to a $\mathfrak{T}$-constraint tree-automaton model. Complexity bounds on the emptiness problem for this model would directly imply bounds on the satisfiability for CTL* with constraints over $\mathfrak{T}$. Thus, investigating whether our approach transfers to a result on the emptiness problem of $\mathfrak{T}$-constraint tree-automata might be a fruitful approach. Secondly, it may be possible to lift our results to the global model checking problem similar to the work of Bozelli and Pinchinat [3] on LTL with constraints over the integers. Finally, it is a very challenging task to decide whether DD's result and our result can be unified to a result on LTL with constraints over the tree with prefix order, lexicographic order and length-comparisons (of maximal common prefixes).

2 Model Checking LTL with Constraints over Trees

We first introduce LTL($\{ \preceq, \sqsubseteq, S \}$), a variant of LTL with local constraints. A model of a formula of this logic is a (multi-) data word where the data comes from some $\{ \preceq, \sqsubseteq, S \}$-structure. We are particularly interested in the case where this structure is an order tree with lexicographic order $\sqsubseteq$. We want to adjust the automata-based model checking methods for LTL to this setting. For this purpose we then recall the definition of tree-constraint automata. The technical core of this paper shows that emptiness of tree-constraint automata is PSPACE-complete. Before we delve into this technical part, we prove that satisfiability and model checking for LTL($\{ \preceq, \sqsubseteq, S \}$) formulas with constraints over the full infinitely branching tree are in PSPACE due to a reduction to the emptiness problem of tree-constraint automata. We conclude this section by providing a reduction of satisfiability and model checking for LTL($\{ \preceq, \sqsubseteq, S \}$) with constraints over the full tree of branching degree k to the corresponding problem over the full infinitely branching tree.

2.1 LTL with Constraints

Constraint LTL over signature $\{ =, \preceq, \sqsubseteq, s_1, s_2, \ldots, s_m \}$ where $S = \{ s_1, \ldots, s_m \}$ is a set of constant symbols, abbreviated LTL($\{ \preceq, \sqsubseteq, S \}$), is given by the grammar

$$\phi ::= \mathsf{X}^i x_1 * s \mid s * \mathsf{X}^i x_1 \mid \mathsf{X}^i x_1 * \mathsf{X}^j x_2 \mid \neg\phi \mid (\phi \wedge \phi) \mid \mathsf{X}\phi \mid \phi \mathsf{U} \psi \mid \mathsf{G}\phi$$

[3] In fact, our proof can be easily adapted to reprove this result.

where $* \in \{ =, \preceq, \sqsubseteq \}$, i, j are natural numbers, x_1, x_2 are variables from some countable fixed set $\mathcal{V}$ and $s \in S$ is a constant symbol. Given a structure $\mathfrak{A} = (A, \preceq^{\mathfrak{A}}, \sqsubseteq^{\mathfrak{A}} s_1^{\mathfrak{A}}, s_2^{\mathfrak{A}}, \ldots, s_m^{\mathfrak{A}})$, an *n-dimensional data word over* $\mathfrak{A}$ is a sequence $(\bar{a}_i)_{i \in \mathbb{N}}$ with $\bar{a}_i \in A^n$. We evaluate a formula ϕ (where $x_1, \ldots, x_n \in \mathcal{V}$ are the variables occurring in ϕ) on n-dimensional data words $(\bar{a}_i)_{i \in \mathbb{N}}$. We write a_i^j for the j-th component of $\bar{a}_i$. We say $(\bar{a}_i)_{i \in \mathbb{N}}$ *is a model of* ϕ, denoted as $(\bar{a}_i)_{i \in \mathbb{N}} \models \phi$, if the usual conditions for LTL hold, and the following additional rules apply for $* \in \{ =, \preceq, \sqsubseteq \}$:

- $(\bar{a}_i)_{i \in \mathbb{N}} \models (\mathsf{X}^i x_k) * (\mathsf{X}^j x_l)$ if and only if $\mathfrak{A} \models a_i^l * a_j^k$,
- $(\bar{a}_i)_{i \in \mathbb{N}} \models (\mathsf{X}^i x_l) * s_j$ (or $s_j * (\mathsf{X}^i x_l)$, resp.) if and only if $\mathfrak{A} \models a_i^l * s_j$ (or $\mathfrak{A} \models s_j * a_i^l$, respectively).

Note that our constraint LTL does not use atomic propositions. On nontrivial structures, proposition p can be resembled by constraints of the form $x_{p_1} = x_{p_2}$.

As for usual LTL one defines dual operators. Then every formula has an equivalent negation normal form where negation only appears in front of atomic constraints ($(\mathsf{X}^i x_1) \preceq (\mathsf{X}^j x_2)$, $s \preceq \mathsf{X}^i x$ or $\mathsf{X}^i x \preceq s$). Using that $X^n(X^i x_k * X^j x_\ell) \equiv X^{i+n} x_k * X^{j+n} x_\ell$ and by introducing auxiliary variables, it is also easy to eliminate exponents in terms:

Proposition 1. *There is a polynomial time algorithm that computes, on input a* LTL($\{ \preceq, \sqsubseteq, S \}$)*-formula* ϕ *an equivalent* LTL($\{ \preceq, \sqsubseteq, S \}$)*-formula* ψ *such that* ψ *does not contain terms of the form* $\mathsf{X}^i x$ *with* $i \geq 2$.

We want to investigate LTL($\{ \preceq, \sqsubseteq, S \}$) in the cases where the structure $\mathfrak{A}$ is one of the following order trees. For each $k \in \{ 2, 3, 4, \ldots \}$, let

$$\mathfrak{T}_\infty^C = (\mathbb{Q}^*, \preceq, \sqsubseteq c_1, c_2, \ldots, c_m) \text{ and } \mathfrak{T}_k^C = (\{1, 2, \ldots, k\}^*, \preceq, \sqsubseteq, c_1, \ldots, c_m)$$

where $\preceq$ is the prefix order, $\sqsubseteq$ is the lexicographic order defined by $w \sqsubseteq v$ if either $w \preceq v$ or there are $q_1, q_2 \in Q$ such that $(w \sqcap v)q_1 \preceq w$, $(w \sqcap v)q_2 \preceq v$ and $q_1 < q_2$, where $<$ is the natural order on $\mathbb{Q}$ and $\sqcap$ denotes the (binary) *greatest common prefix operator*, and $C = (c_1, c_2, \ldots c_m)$ is a tuple of constants in $\mathbb{Q}^*$ or $\{ 1, 2, \ldots, k \}^*$, respectively.

2.2 Constraint Automata

In the following, we investigate the satisfiability and model checking problems for LTL($\{ \preceq, \sqsubseteq, S \}$) over models with data values in one of the trees $\mathfrak{T}_k^C$ for $k \in \{ \infty, 2, 3, 4, \ldots \}$. We follow closely the automata theoretic approach of Vardi and Wolper [14] which provides a reduction of model checking for LTL to the emptiness problem of Büchi automata. In order to deal with the constraints, we use $\mathfrak{T}_k^C$*-constraint automata* (cf. [13]) instead of Büchi automata. Next we recall the definition of constraint automata and state our main result concerning emptiness of constraint automata. We then derive analogous results of Vardi and Wolper's decidability results on LTL for LTL($\{ \preceq, \sqsubseteq, S \}$)with constraints

over $\mathfrak{T}_k^C$. A $\mathfrak{T}_k^C$-*constraint automaton* is defined as a usual Büchi automaton but instead of labelling transitions by some letter from a finite alphabet we label them by Boolean combinations of constraints which the current and the next data values have to satisfy in order to apply the transition.

Definition 2. – *An* n-dimensional $\mathfrak{T}_k^C$-constraint automaton *is a quadruple* $\mathbb{A} = (Q, I, F, \delta)$ *where* Q *is a finite set of states,* $I \subseteq Q$ *the initial states,* $F \subseteq Q$ *the set of accepting states and* $\delta \subseteq Q \times B_n^C \times Q$ *the* transition relation *where* B_n^C *is the set of all quantifier-free formulas over signature* $\{ \preceq, \sqsubseteq \} \cup C$ *with variables* $x_1, \ldots, x_n, y_1, \ldots, y_n$*, i.e., propositional logic formulas with atomic formulas* $v * v'$*, with* $* \in \{=, \preceq, \sqsubseteq\}$ *and* v*,* v' *are variables or constants.*
– *A* configuration *of the automaton* $\mathbb{A}$ *is a tuple in* $Q \times (\{ 1, 2, \ldots, k \}^*)^n)$ *(or* $(\mathbb{Q}^*)^n$ *if* $k = \infty$*).*
– *We define* $(q, \bar{w}) \to (p, \bar{v})$ *iff there is a transition* $(q, \beta(x_1, \ldots x_n, y_1, \ldots, y_n), p)$ *such that* $\mathfrak{T}_k^C \models \beta(\bar{w}, \bar{v})$*.*
– *A* run *of* $\mathbb{A}$ *is a finite or infinite sequence of configurations* $r = (c_j)_{j \in J}$ *(*$J \subseteq \mathbb{N}$ *an interval) such that* $c_j \to c_{j+1}$ *for all* $j, j+1 \in J$*. For a finite run* $r = (c_i)_{i_1 \leq i \leq i_2}$ *with* $i_1 \leq i_2 \in \mathbb{N}$ *we say* r *is a* run from c_{i_1} to c_{i_2}*.*
– *A run* $r = (c_i)_{i \in \mathbb{N}}$ *is* accepting *if* $c_0 = (q, d_1, \ldots, d_n)$ *for an initial state* $q \in I$ *and a final state* $f \in F$ *appears in infinitely many configurations of* r*.*
– *The* set of all words accepted by $\mathbb{A}$ *comprises all* $\bar{w}_1 \bar{w}_2 \cdots \in ((\mathbb{Q}^*)^n)^\omega$ *(or* $(\{1, \ldots, k\})^n)^\omega$ *if* $k \neq \infty$*) such that there is an accepting infinite run* $(c_i)_{i \in \mathbb{N}}$ *with* $c_i = (q_i, \bar{w}_i)$*.*

In the following sections (see Theorem 6) we prove that emptiness of n-dimensional $\mathfrak{T}_\infty^C$-constraint automata is PSPACE-complete in terms of $|Q| + |C| + k + m$ where m is the length of the longest constant occurring in C. We next apply this result in order to obtain PSPACE-completeness of satisfiability and model checking.

2.3 Satisfiability and Model Checking of Constraint LTL

Definition 3. *Let* $k \in \{ \infty, 2, 3, 4, \ldots \}$*.*

SAT($\mathfrak{T}_k^C$) *denotes the* satisfiability problem *for* LTL($\{ \preceq, \sqsubseteq, S \}$) *over* $\mathfrak{T}_k^C$*: given a set of constants* C *and a* LTL($\{ \preceq, \sqsubseteq, S \}$)*-formula* φ*, is there a data word* $(\bar{w}_i)_{i \in \mathbb{N}}$ *over* $\mathfrak{T}_k^C$ *such that* $(\bar{w}_i)_{i \in \mathbb{N}} \models \varphi$*?*

MC($\mathfrak{T}_k^C$) *denotes the* model checking problem for $\mathfrak{T}_k^C$-constraint automata against LTL($\{ \preceq, \sqsubseteq, S \}$)*: given a set of constants* C*, a* $\mathfrak{T}_k^C$*-constraint automaton* $\mathbb{A}$ *and a* LTL($\{ \preceq, \sqsubseteq, S \}$)*-formula* φ*, is there a data word* $(\bar{w}_i)_{i \in \mathbb{N}}$ *over* $\mathfrak{T}_k^C$ *accepted by* $\mathbb{A}$ *such that* $(\bar{w}_i)_{i \in \mathbb{N}} \models \varphi$*?*

Theorem 4. *Let* $k \in \{ \infty, 2, 3, 4, \ldots \}$ *and* C *a set of constants.* SAT($\mathfrak{T}_k^C$) *and* MC($\mathfrak{T}_k^C$) *are* PSPACE*-complete.*

Proof. Since there is an automaton accepting all data words, the satisfiability problem reduces to the model checking problem whence it suffices to prove the claim on model checking. Hardness follows directly from the known results for

LTL. We first prove $\mathrm{MC}(\mathfrak{T}_\infty^C) \in \mathrm{PSPACE}$ and then we provide a reduction of $\mathrm{MC}(\mathfrak{T}_k^C)$ to $\mathrm{MC}(\mathfrak{T}_\infty^C)$ for all other k.

Case $k = \infty$. Let $C \subseteq \mathbb{Q}^*$ be a finite set of constants, $\mathbb{A}$ a $\mathfrak{T}_\infty^C$-constraint automaton and $\varphi \in \mathrm{LTL}(\{\preceq, \sqsubseteq, S\})$. Due to Proposition 1 we can assume that all atomic constraints occurring in φ only concern the current and the next data values. Recall that Vardi and Wolper [14] provided a translation from LTL to Büchi automata such that the resulting automaton accepts some word if and only if it is a model of the formula.

This translation directly lifts to a translation of $\mathrm{LTL}(\{\preceq, \sqsubseteq, S\})$ over $\mathfrak{T}_\infty$ to $\mathfrak{T}_\infty$-constraint automata. As in the standard construction, each state of the automaton is a subset of (the negation closure of) the set of subformulas of the $\mathrm{LTL}(\{\preceq, \sqsubseteq, S\})$-formula. Intuitively, an accepting run of the automaton on $(\bar{w}_i)_{i\in\mathbb{N}}$ is at position i_0 in a state containing some subformula ψ if and only if $(\bar{w}_i)_{i\geq i_0} \models \psi$. Obviously the dependence of the transitions of a constraint automaton on the order of the current and next data values is exactly what is needed to allow the automaton to switch from one state to another only if the (possibly negated) atomic constraints contained in the current state are satisfied by the current and the next data values.

Thus, we obtain a constraint automaton $\mathbb{B}$ such that $\mathbb{B}$ accepts $(\bar{w}_i)_{i\in\mathbb{N}}$ if and only if $(\bar{w}_i)_{i\in\mathbb{N}} \models \varphi$. Since the usual product construction for Büchi automata lifts also to constraint automata, we easily construct in polynomial space an automaton $\mathbb{C}$ such that $\mathbb{C}$ accepts a word if and only if both $\mathbb{A}$ and $\mathbb{B}$ accept this word. Thus, the set of all words accepted by $\mathbb{C}$ is non-empty if and only if there is a data word $(\bar{w}_i)_{i\in\mathbb{N}}$ such that $\mathbb{A}$ accepts $(\bar{w}_i)_{i\in\mathbb{N}}$ and $(\bar{w}_i)_{i\in\mathbb{N}} \models \varphi$. Since emptiness is in PSPACE the claim follows.

Case $k \neq \infty$. Now we turn to the case $\mathfrak{T}_k^C$ where $k \neq \infty$. Let C_l be the set of $\preceq$-maximal elements of C, and let φ and $\mathbb{A}$ as before. Without loss of generality we can assume that C_l intersects every infinite branch in $\{1, 2, \ldots, k\}^\omega$(If not, add ci as a new constant for every c in the prefix-closure of C and $i \in \{1, 2, \ldots, k\}$, which only causes a polynomial growth of the input). We claim that $(C, \mathbb{A}, \varphi)$ is a positive instance of $\mathrm{MC}(\mathfrak{T}_k^C)$ if and only if $(C, \mathbb{A}, \psi)$ is a positive instance of $\mathrm{MC}(\mathfrak{T}_\infty^C)$ where $\mathbb{A}$ is seen as a $\mathfrak{T}_\infty^C$-automaton and $\psi = \varphi \wedge \mathsf{G} \bigwedge_{i=1}^n \bigvee_{c\in C_l} (x_i \preceq c \vee c \preceq x_i)$ where $x_1, x_2, \ldots, x_n$ is the set of variables occurring in the constraints of φ. Basically, ψ is φ with the additional condition that the data values occurring in a model form a tree of branching degree k at all constants. It is clear that every witness $(\bar{w}_i)_{i\in\mathbb{N}}$ for the former model checking problem is a witness for the latter.

For the converse assume that $(\bar{w}_i)_{i\in\mathbb{N}}$ is a data word over $\mathfrak{T}_\infty$ accepted by $\mathbb{A}$ satisfying ψ. Note that there is an injective map $g : \mathbb{Q}^* \to \{1, 2\}^*$ preserving $\preceq$ and $\sqsubseteq$ in both directions (cf. Appendix B). Moreover, by definition of ψ we conclude that every value occurring in $(\bar{w}_i)_{i\in\mathbb{N}}$ is either a prefix of one of the constants or of the form $cq_1q_2 \ldots q_n$ for some maximal constant $c \in C_l$. Thus, we can define $\bar{v}_i = (v_i^1, v_i^2, \ldots, v_i^n)$ where $v_i^j = w_i^j$ if $w_i^j \preceq c$ for some $c \in C_l$ and $v_i^j = cg(u)$ if $w_i^j = cu$ for some $c \in C_l$ and $u \neq \varepsilon$. Clearly $(\bar{v}_i)_{i\in\mathbb{N}}$ is a data word over $\mathfrak{T}_k$. Since g preserves $\preceq$, $\sqsubseteq$ and all constants, it is a model of ψ accepted by $\mathbb{A}$ whence it is also a model of φ. □

Remark 5. Demri and Deter [9] conjectured that if the arity k of the tree is part of the input to the satisfiability problem, it is still in PSPACE. Our proof confirms that this branching degree uniform satisfiability problem is PSPACE-complete.

3 Emptiness of Tree Constraint Automata

Recall that every nonempty Büchi automaton has an accepting run which is ultimately periodic. We first prove that a nonempty constraint automaton has an accepting run which ultimately consists of loops that never contract the distances of data values and keep the order type of the data values constant. We then define the notion of the type of a run. It turns out that such a non-contracting loop exists if and only if the automaton has a run realising a type among a certain set. Finally, we provide a PSPACE-algorithm that checks whether an automaton realises a given type. Putting all these together yields our main technical result.

Theorem 6. *Emptiness of $\mathfrak{T}_\infty^C$-constraint automata is in* PSPACE.

3.1 Emptiness and Stretching Loops

We first introduce some notation before defining our notion of stretching loop and characterising emptiness in terms of stretching loops.

From now on a *word* is always an element of $\mathbb{Q}^*$, $\bigsqcap$ ($\sqcap$) denotes the (binary) *greatest common prefix operator*, and we fix a finite tuple of words $C = (c_1, c_2, \ldots, c_m)$ called constants. **We assume that C is closed under prefixes.** Note that closing C under prefixes results only in polynomial growth.

Definition 7. *Let $s_1, \ldots s_n$ be constant symbols and $\sigma = \{ \preceq, \sqsubseteq, s_1, s_2, \ldots, s_n \}$. Given a tuple $\bar{w} = (w_1, w_2, \ldots, w_n)$ of words, the* maximal common ancestor tree *of $\bar{w}$ is the σ-structure*

$$\mathrm{MCAT}(\bar{w}) = (M, \preceq \restriction_{M^2}, \sqsubseteq \restriction_{M^2}, w_1, w_2, \ldots, w_n),$$

where w_i is the interpretation of constant symbol s_i and

$$M = \{ \varepsilon \} \cup \left\{ \textstyle\bigsqcap_{i \in I} w_i \;\middle|\; \emptyset \neq I \subseteq \{ 1, 2, \ldots, n \} \right\}.$$

The (order) type $\mathrm{typ}(\bar{w})$ of $\bar{w}$ *is the σ-isomorphism type of* $\mathrm{MCAT}(\bar{w})$. *We set* $\mathrm{MCAT}_C(\bar{w}) \coloneqq \mathrm{MCAT}(\bar{w}, C)$ *and* $\mathrm{typ}_C(\bar{w}) \coloneqq \mathrm{typ}(\bar{w}, C)$.

Labelling the words from $\bar{w}$ by constant symbols has the following consequence: if $\mathrm{typ}_C(\bar{w}) = \mathrm{typ}_C(\bar{v})$ for $\bar{w} = (w_1, w_2, \ldots, w_n)$ then there is a unique isomorphism h from $\mathrm{MCAT}_C(\bar{w})$ to $\mathrm{MCAT}_C(\bar{v})$ which maps $c \mapsto c$ for every $c \in C$ and $w_i \to v_i$ for w_i the i-th element of $\bar{w}$ and v_i the i-th element of $\bar{v}$.

Definition 8. *For $n \in \mathbb{N}$ we define a relation $\leq_C$ on configurations from $Q \times (\mathbb{Q}^*)^n$ by $(q, \bar{w}) \leq_C (p, \bar{v})$ if $q = p$, $\mathrm{typ}_C(\bar{w}) = \mathrm{typ}_C(\bar{v})$ and the induced isomorphism $h : \mathrm{MCAT}_C(\bar{w}) \to \mathrm{MCAT}_C(\bar{v})$ satisfies for all $d, e \in \mathrm{MCAT}_C(\bar{w})$ if $d \prec e$ then $|h(e)| - |h(d)| \geq |e| - |d|$.*

Intuitively, $(q, \bar{w}) \leq_C (q, \bar{v})$ holds if both data tuples have the same order type and the lengths of intervals in $\mathrm{MCAT}_C(\bar{v})$ seen as a subtree of $\mathbb{Q}^*$ are greater than the lengths of the corresponding intervals in $\mathrm{MCAT}_C(\bar{w})$. In the following sections, we make extensive use of the following properties of $\leq_C$.

Lemma 9. *1. $\leq_C$ is a well-quasi order.*
2. *The (inverse) transition relation $\to$ ($\to^{-1}$) is strongly upwards compatible with respect to $\leq_C$ in the sense of [12], i.e., if $u \to v$ ($u \to^{-1} v$) and $u \leq_C u'$, then there is a v' such that $v \leq_C v'$ and $u' \to v'$ ($u' \to^{-1} v'$).*
3. *Given two configurations $(q, \bar{w})$ and $(q, \bar{v})$ such that $\mathrm{typ}_C(\bar{w}) = \mathrm{typ}_C(\bar{v})$ then there is a configuration $(q, \bar{u})$ such that $(q, \bar{w}) \leq_C (q, \bar{u})$ and $(q, \bar{v}) \leq_C (q, \bar{u})$.*

Definition 10. *A* loop *is a finite run $r = (c_i)_{i \leq n}$ with $c_0 = (q, \bar{w})$, $c_n = (q, \bar{v})$ and $\mathrm{typ}_C(\bar{w}) = \mathrm{typ}_C(\bar{v})$. We say that a loop $r = (c_i)_{i \leq n}$ is* stretching *if $c_0 \leq_C c_n$.*

Lemma 11. *Let $\mathbb{A}$ be a constraint automaton. $\mathbb{A}$ has an accepting run if and only if there are partial runs r_1, r_2 where r_1 starts in an initial configuration and ends in some configuration c whose state is a final state, and where r_2 is a stretching loop starting in c.*

Proof. ($\Rightarrow$). Let $r = (c_i)_{i \in \mathbb{N}}$ be an accepting run. Since r contains infinitely many configurations with a final state and $\leq_C$ is a wqo, we can find numbers $n_1 < n_2$ such that $c_{n_1} \leq_C c_{n_2}$ whence $(c_n)_{n \leq n_1}$, $(c_n)_{n_1 \leq n \leq n_2}$ are the desired runs.
($\Leftarrow$). Assume r_1 is a run from some initial configuration to c_1 whose state is a final state $f \in F$ and r_2 is a stretching loop starting in c_1 and ending in c_2. Since $c_1 \leq_C c_2$, iterated use of strong upwards compatibility (Lemma 9) yields runs r_i from c_{i-1} to c_i such that $c_{i-1} \leq_C c_i$ for all $i \geq 3$. Clearly, the composition of $r_1, r_2, r_3, r_4, \ldots$ is an accepting run. □

3.2 Stretching Loops and Types of Runs

Definition 12. *Let $r = (c_i)_{0 \leq i \leq n}$ be a finite run, with $c_0 = (q, \bar{w})$ and $c_n = (p, \bar{v})$. Setting $\pi = \mathrm{typ}_C(\bar{w}, \bar{v})$, we say r has* type $\mathrm{typ}(r) = (q, \pi, p)$.

Definition 13. *Let $\bar{w}, \bar{v}$ be k-tuples of words such that $\mathrm{typ}_C(\bar{w}) = \mathrm{typ}_C(\bar{v})$ and let h be the induced isomorphism from $\mathrm{MCAT}_C(\bar{w})$ to $\mathrm{MCAT}_C(\bar{v})$. $(\bar{w}, \bar{v})$ is called* contracting *if one of the following holds.*

1. *There is some $d \in \mathrm{MCAT}_C(\bar{w})$ such that $h(d) \prec d$.*
2. *There are $d, e \in \mathrm{MCAT}_C(\bar{w})$ such that $d \prec e$, $h(e) = e$ and $d \prec h(d)$.*

We call a loop r from $(q, \bar{w})$ to $(q, \bar{v})$ contracting if $(\bar{w}, \bar{v})$ is contracting. Otherwise, we call it (and its type) noncontracting.

Remark 14. The type of a loop determines whether it is noncontracting. Let us explain the term 'contracting'. Fix a loop from $(q, \bar{w})$ to $(q, \bar{v})$. The isomorphism $h : \mathrm{MCAT}_C(\bar{w}) \to \mathrm{MCAT}_C(\bar{v})$ relates for every pair $\bigsqcap_{k \in K} w_k \prec \bigsqcap_{l \in L} w_l$ the interval $(\bigsqcap_{k \in K} w_k, \bigsqcap_{l \in L} w_l)$ with the interval $(\bigsqcap_{k \in K} v_k, \bigsqcap_{l \in L} v_l)$. By definition, for every contracting loop there is a pair (K, L) such that (setting $\bigsqcap_{k \in \emptyset} w_k = \varepsilon$)

$$|\textstyle\bigsqcap_{l \in L} w_l| - |\bigsqcap_{k \in K} w_k| > |\bigsqcap_{l \in L} v_l| - |\bigsqcap_{k \in K} v_k|.$$

The technical core of this section shows that if an automaton admits a noncontracting loop then it admits a stretching loop with the same initial and final state. This allows to rephrase the conditions from Lemma 11 in terms of types. The proof of this claim requires some definitions and preparatory lemmas.

Definition 15. *Let u be a word and $m \in \mathbb{N}$. We define the* insertion of an m-gap at u *to be* $\iota_u^m : \mathbb{Q}^* \to \mathbb{Q}^*$ *given by* $\iota_u^m(w) = \begin{cases} w & \text{if } u \not\preceq w, \\ u0^m v & \text{if } w = uv. \end{cases}$

Given a finite run r, the sequence $\iota_u^m(r)$ obtained by applying ι_u^m to each data value of r is the run obtained by insertion of an m-gap at u in r.

For $r = (c_i)_{i \in I}$ and $r' = (d_i)_{i \in I}$ we write $r \leq_C r'$ if $c_i \leq_C d_i$ for all $i \in I$. Note that the insertion of a gap preserves $\preceq, \sqsubseteq$ and $\sqcap$ in both directions.

Lemma 16. *Given a run r and a word u such that u is not a prefix of any constant. The sequence $\iota_u^m(r)$ is indeed a run r' of the same type and $r \leq_C r'$.*

Let $w, v \in \mathbb{Q}^*$. We say w is *incomparable left* of v if $w \sqsubseteq v$ and $w \not\preceq v$. In the same situation we call v *incomparable right* of w.

Lemma 17. *Let $\bar{w}, \bar{v}$ be k-tuples with $\mathrm{typ}(\bar{w}) = \mathrm{typ}(\bar{v})$. If w_i is incomparable left (right) of v_i and $v_i \preceq w_j$, then w_j is incomparable left (right) of v_j and incomparable right (left) of w_i.*

Proof. By type equality, we have that v_i is incomparable left of v_j, whence the same holds for its descendant w_j. From $w_i \sqsubseteq v_i \preceq w_j$ follows $w_i \sqsubseteq w_j$, and $w_i \not\preceq w_j$ as $w_i \not\preceq v_i$. □

Proposition 18. *Let r be a noncontracting loop. There is a stretching loop r' such that $r \leq_C r'$.*

Proof. Let r from $(q, \bar{w})$ to $(q, \bar{v})$ be a noncontracting loop and $h : \mathrm{MCAT}_C(\bar{w}) \to \mathrm{MCAT}_C(\bar{v})$ the induced isomorphism. We iteratively define a sequence $r = r_0 \leq_C r_1 \leq_C \ldots \leq_C r_n$ of runs until r_n is stretching.

We call a pair $(u_1, u_2) \in \mathrm{MCAT}_C(\bar{w})^2$ *problematic* (with respect to r) if $u_1 \preceq u_2$ and $|u_2| - |u_1| > |h(u_2)| - |h(u_1)|$. Recall that in this case u_2 and $h(u_2)$ are not prefix of any constant c from C because h fixes all such elements. Let P_r be the set of all problematic pairs. We split the set of all problematic pairs into three parts, which we handle separately (cf. Figure 1 for an example). Let

$$L_r = \{\, (u_1, u_2) \in P_r \mid u_2 \text{ incomparable left of } h(u_2) \,\},$$
$$R_r = \{\, (u_1, u_2) \in P_r \mid u_2 \text{ incomparable right of } h(u_2) \,\}, \text{ and}$$
$$D_r = \{\, (u_1, u_2) \in P_r \mid u_2 \text{ comparable to } h(u_2) \,\}.$$

L-Step: If L_r is nonempty, choose the $\sqsubseteq$-minimal u_2 such that there is u_1 with $(u_1, u_2) \in L_r$. Now fix u_1 such that $(u_1, u_2) \in L_r$ and $d \coloneqq (|u_2| - |u_1|) - (|h(u_2)| - |h(u_1)|)$ is maximal. Let $\iota = \iota_{h(u_2)}^d$ be the insertion of a d gap at $h(u_2)$ and $r' = \iota(r)$.

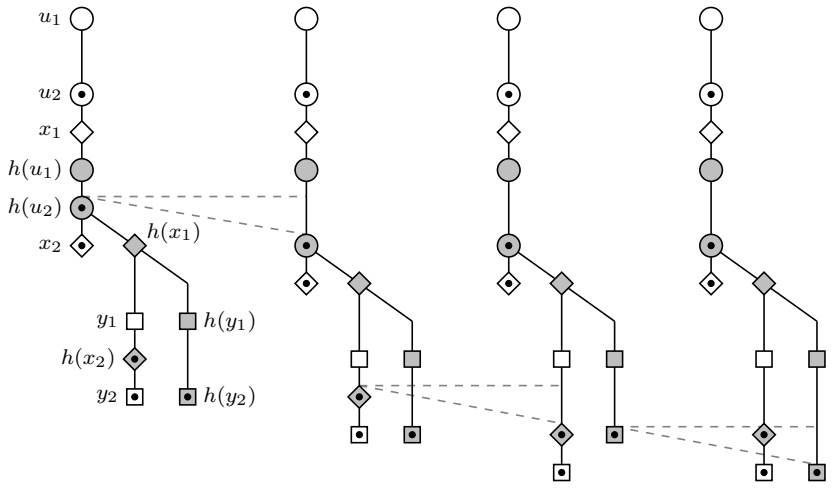

Figure 1. Example for Proposition 18: In the first tree (u_1, u_2) is problematic , insertion of a gap (D-Step) at $h(u_2)$ makes (the pair corresponding to) (x_1, x_2) problematic; insertion of a gap (L-Step) at $h(x_2)$ makes (y_1, y_2) problematic; insertion of a gap (L-Step) at $h(y_2)$ makes the tree stretching.

Denote by $\iota(\bar{w})$ ($\iota(\bar{v})$) the data values of the first (last, respectively) configuration of r'. Let $h' : \mathrm{MCAT}_C(\iota(\bar{w})) \to \mathrm{MCAT}_C(\iota(\bar{v}))$ be the corresponding isomorphism. By definition the set $L_{r'} = \{ (x_1, x_2) \in P_{r'} \mid x_2 \text{ incomparable left of } h'(x_2) \}$ does not contain a pair $(u, \iota(u_2))$ for any $u \in \mathrm{MCAT}_C(\iota(\bar{w}))$. Nevertheless, r' may admit problematic pairs that are not problematic with respect to r. This can happen if there are $x_1, x_2 \in \mathrm{MCAT}_C(\bar{w})$ such that $x_1 \prec h(u_2) \preceq x_2$ holds, but $h(x_1) \prec h(u_2) \preceq h(x_2)$ does not. Then, the distance between $\iota(x_1)$ and $\iota(x_2)$ is greater than the distance between x_1 and x_2 (by d). On the other hand, either both or none of $h'(\iota(x_1))$ and $h'(\iota(x_2))$ are shifted by the insertion of the gap whence their distance is equal to the distance of $h(x_1)$ and $h(x_2)$.

In this case, possibly $(\iota(x_1), \iota(x_2))$ is problematic w.r.t. r' while (x_1, x_2) is not problematic w.r.t r. Application of Lemma 17 shows that then x_2 is incomparable left of $h(x_2)$ and u_2 is incomparable left of x_2 whence the same holds for $\iota(x_2), h'(\iota(x_2)) = \iota(h(x_2))$ and $\iota(u_2)$. Thus, if $(\iota(x_1), \iota(x_2))$ is problematic, then $(\iota(x_1), \iota(x_2) \in L_{r'}$ and $\iota(u_2)$ is strictly incomparable left of $\iota(x_2)$.

Thus, iteration of this step only creates problematic pairs that are more and more to the right with respect to $\mathrm{typ}_C(\bar{w}_n) = \mathrm{typ}_C(\iota(\bar{w}))$. Since $\mathrm{typ}_C(\bar{w}_n)$ is finite, we eventually do not introduce new problematic pairs and obtain a run r_i such that $L_{r_i} = \emptyset$ and $r \leq_C r_i$ because r_i results from insertion of several gaps in r.

R-Step: If $R_r \neq \emptyset$, proceed as in (L-Step) all "left" and "right".

D-Step: If $L_r = R_r = \emptyset$ and r is not stretching, then $D_r \neq \emptyset$. Choose u_2 $\sqsubseteq$-minimal in $\mathrm{MCAT}(\bar{w})$ such that there is some u_1 with $(u_1, u_2) \in D_r$ and choose $u_1 \prec u_2$ in $\mathrm{MCAT}_C(\bar{w})$ such that $d \coloneqq (|u_2| - |u_1|) - (|h(u_1)| - |h(u_2)|)$ is maximal. Since r is not contracting we have $u_2 \preceq h(u_2)$ and $u_1 \preceq h(u_1)$. Assume

$u_2 = h(u_2)$, then $u_1 \prec h(u_1)$ as $(u_1, u_2) \in D$. This contradicts that r is not contracting. Thus $u_2 \prec h(u_2)$. Again, let $\iota = \iota^d_{h(u_2)}$ and $r' = \iota(r)$.

Define $\iota(\bar{w}), \iota(\bar{v})$ and h' as in the L-step. Again there may be a pair (x_1, x_2) which is not problematic with respect to r while $(\iota(x_1), \iota(x_2))$ is problematic with respect to r'. If $R_{r'}$ or $L_{r'}$ are nonempty, we can deal with those problematic intervals using R- or L-steps. This finally leads to a run r_j with $R_{r_j} = L_{r_j} = \emptyset$. Moreover, for every pair (x_1, x_2) such that this pair is not problematic with respect to r but $(\iota(x_1), \iota(x_2))$ is problematic with respect to r', we conclude that x_2 is strictly below u_2 whence $\iota(x_2)$ is strictly below $\iota(u_2)$ w.r.t. $\preceq$. Thus, the endpoints of problematic pairs move downwards (in $\mathrm{typ}_C(\bar{w}, \bar{v}) = \mathrm{typ}_C(\bar{w}', \bar{v}')$) and eventually all problematic pairs are removed. Once r_j is a loop without problematic pair, it is stretching. □

Corollary 19. *The set of words accepted by an automaton $\mathbb{A}$ is nonempty if and only if there are runs r_1 r_2 such that r_2 is a noncontracting loop starting in configuration $(f, \bar{w})$ where f is a final state and r_1 is a run from an initial configuration to some configuration $(f, \bar{v})$ such that* $\mathrm{typ}_C(\bar{w}) = \mathrm{typ}_C(\bar{v})$.

Proof. Due to Lemma 11, only ($\Leftarrow$) requires a proof. Assume that there are runs r_1, r_2 as stated above. By Lemma 9, there is a run $r_2 \leq_C r_2'$ such that $(f, \bar{v}) \leq_C c_0$ for c_0 the initial configuration of r_2'. Note that r_2' is also noncontracting whence by Proposition 18 there is a stretching loop r_2'' such that $r_2' \leq_C r_2''$. Hence this loop starts in some configuration c_1 such that $(f, \bar{v}) \leq_C c_1$. Applying Lemma 9 to r_1 and c_2 we obtain a run r_1' from an initial configuration to c_2. Thus, r_1' and r_2'' match the conditions of Lemma 11 which completes the proof. □

3.3 Emptiness and Computation of Types

In order to turn this characterisation of emptiness in terms of types into an effective algorithm for the emptiness problem the last missing step is to compute whether a given type is realised by some run of a given automaton.

For this purpose, we equip the set of all sets of types with a product operation. Let S, T be sets of types of runs; a type (q, π, p) is in $S \cdot T$ if there are $(q, \pi_1, r) \in S$, $(r, \pi_2, p) \in T$ and tuples $\bar{u}, \bar{v}, \bar{w}$ such that $\mathrm{typ}_C(\bar{u}, \bar{v}) = \pi_1$, $\mathrm{typ}_C(\bar{v}, \bar{w}) = \pi_2$ and $\mathrm{typ}_C(\bar{u}, \bar{w}) = \pi$. Let T_1 denote the set of all types of runs of length 1 (of some fixed automaton $\mathbb{A}$) and $T_1^+ = \bigcup_{n \in \mathbb{N}} (T_1)^n$. By induction on the length, one easily shows that every finite run r of $\mathbb{A}$ satisfies $\mathrm{typ}(r) \in (T_1)^+$. Conversely, for every type $t \in (T_1)^+$ there is also a run of $\mathbb{A}$ of type t. This is due to the fact that gap-insertion preserves types (Lemma 16), $\rightarrow$ is upwards compatible (Lemma 9) and that trees of a given type t_1 with large gaps have, for all order types t, t_2 with $t \in \{ t_1 \} \cdot \{ t_2 \}$, an extension to a tree witnessing this product. The necessary proofs are not very difficult but tedious and lengthy.

We conclude that a type t is in $(T_1)^+$ if and only if t is the type of some run of $\mathbb{A}$. Moreover, types of runs can be represented in polynomial space (in terms of the constants and the dimension of a given automaton) and the product of types can be computed in PSPACE. Thus, we can determine whether an automaton

$\mathbb{A}$ realises a type t by guessing types in T_1 and computing an element of their product until it matches t. This proves the following proposition.

Proposition 20. *There is a* PSPACE*-algorithm that, given a* $\mathfrak{T}^C_\infty$*-constraint automaton* $\mathbb{A}$ *and a type* t*, determines whether there is a run of* $\mathbb{A}$ *of type* t*.*

Together with Corollary 19 we obtain an algorithm proving Theorem 6.

Proof (of Theorem 6). By Corollary 19 it suffices that the algorithm guesses a type (i, π, f) and a noncontracting type (f, π', f) such that i is an initial state, f is a final state, and the order type of the last elements of π coincides with the order type of the first elements of π', and then checks whether these types are realised by actual runs using the previous proposition. □

Acknowledgement

We thank Claudia Carapelle for extremely helpful discussions and proof reading.

References

1. Bojanczyk, M., Torunczyk, S.: Weak MSO+U over infinite trees. In: Proc. of STACS 2012. LIPIcs, vol. 14, pp. 648–660. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2012)
2. Bozzelli, L., Gascon, R.: Branching-time temporal logic extended with qualitative presburger constraints. In: Proc. of LPAR 2006. LNCS, vol. 4246, pp. 197–211. Springer (2006)
3. Bozzelli, L., Pinchinat, S.: Verification of gap-order constraint abstractions of counter systems. Theor. Comput. Sci. 523, 1–36 (2014)
4. Carapelle, C., Kartzow, A., Lohrey, M.: Satisfiability of ECTL* with constraints, under submission
5. Carapelle, C., Feng, S., Kartzow, A., Lohrey, M.: Satisfiability of ECTL* with tree constraints, under sumbission, available at `http://arxiv.org/abs/1306.0814`
6. Carapelle, C., Kartzow, A., Lohrey, M.: Satisfiability of CTL* with constraints. In: Proc. of CONCUR 2013. pp. 455–469 (2013)
7. Cerans, K.: Deciding properties of integral relational automata. In: Proc. of ICALP 1994. pp. 35–46 (1994)
8. Demri, S., Deters, M.: Temporal logics on strings with prefix relation. Research Report LSV-14-13, Laboratoire Spécification et Vérification, ENS Cachan, France (Dec 2014), `http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/PDF/rr-lsv-2014-13.pdf`, 27 pages
9. Demri, S., D'Souza, D.: An automata-theoretic approach to constraint LTL. Inf. Comput. 205(3), 380–415 (2007)
10. Demri, S., Gascon, R.: Verification of qualitative Z constraints. Theor. Comput. Sci. 409(1), 24–40 (2008)
11. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theor. Comput. Sci. 256(1-2), 63–92 (2001)
12. Gascon, R.: An automata-based approach for CTL* with constraints. Electr. Notes Theor. Comput. Sci. 239, 193–211 (2009)
13. Vardi, M.Y., Wolper, P.: Reasoning about infinite computations. Inf. Comput. 115(1), 1–37 (1994)

A Proof of Proposition 1

First we recall the proposition.

Proposition 21. *There is a polynomial time algorithm that computes, on input a* LTL($\{\preceq, \sqsubseteq, S\}$)*-formula* ϕ *an equivalent* LTL($\{\preceq, \sqsubseteq, S\}$)*-formula* ψ *such that* ψ *does not contain terms of the form* $\mathsf{X}^i x$ *with* $i \geq 2$. □

Proof. First, we can replace any occurrence of $\mathsf{X}^i x * \mathsf{X}^j y$ by $\mathsf{X}^{\min(i,j)}(\mathsf{X}^{i-\min(i,j)}) * (\mathsf{X}^{j-\min(i,j)} y)$. Now assume that there is a subformula of the form $\mathsf{X}^i x * y$ (the case $x * \mathsf{X}^j y$ is symmetrical). Introducing fresh variables $y_0, y_1, \ldots, y_{i-1}$ we replace this formula by the formula $x * y_i$ and add the conjunct $\mathsf{G}(y_0 = y \wedge \bigwedge_{j=1}^{i} y_j = \mathsf{X} y_{j-1})$ which is polynomial in i. Obviously, this replacement yields an equivalent formula. Iterating this process for all constraints, we obtain the desired formula ψ. □

B Missing part of Theorem 4

Let $\mathfrak{O} = (\{11, 22\}^* 12, \sqsubseteq)$ where $\sqsubseteq$ denotes the lexicographical order.

Lemma 22. $\mathfrak{O}$ *and* $(\mathbb{Q}, <)$ *are isomorphic.*

Proof. $\mathfrak{O}$ is countable and does not have endpoints because $(11^n 12)_{n\in\mathbb{N}}$ forms a strictly descending sequence such that any element of $\mathfrak{O}$ is minorised by some element of the chain. Analogously, $(22^n 12)_{n\in\mathbb{N}}$ is a strictly increasing sequence majorising every element. Thus, it is left to show that $\sqsubseteq$ is a dense order. Let $w, v \in \mathfrak{O}$ with $w \sqsubseteq v$. Writing $w = w_1 w_2 \ldots w_k$ with $w_i \in \{11, 12, 22\}$ and $v = v_1 v_2 \ldots v_l$ with $v_i \in \{11, 12, 22\}$ let i be minimal such that $w_i \neq v_i$. If $v_i = 12$ then $w_i = 11$ and $w_1 w_2 \ldots w_i (22)^{|w|} 12$ is between w and v. If $v_i = 22$ and $w_i = 11$ or $w_i = 12$ then $w \prec w_1 w_2 \ldots w_{i-1} 22 (11)^{|v|} 12 \prec v$. □

Definition 23. *For* σ *some signature and* σ*-structures* $\mathfrak{A}$ *and* $\mathfrak{B}$ *we say a homomorphism* $h : \mathfrak{A} \to \mathfrak{B}$ *is a* σ-injection *if it is injective and preserves the relations, functions and constants under preimages.*

Lemma 24. *Let* $h : (\mathbb{Q}, <) \to \mathfrak{O}$ *be an isomorphism. The extension* $g : \mathbb{Q}^* \to (\{11, 22\}^* 12)^*$*, given by* $g(q_1 q_2 \ldots q_n) = h(q_1)h(q_2)\ldots h(q_n)$ *is an* $\{\preceq, \sqsubseteq\}$*-injection of* $\mathfrak{T}_\infty = (Q^*, \preceq, \sqsubseteq)$ *into* $\mathfrak{T}_2 = (\{1, 2\}^*, \preceq, \sqsubseteq)$.

Proof. Note that g is injective: if w is in $\mathrm{im}(g)$, then the number of occurrences of 12 where 1 occurs at an odd position determines the length of every preimage v such that $g(v) = w$. It is then a routine check to prove uniqueness of v.

We next show that g preserves $\preceq$ (in both directions). It is obvious from the definition that $w \preceq v$ implies $g(w) \preceq g(v)$. Now assume that $g(w) \preceq g(v)$. Due to the same argument as in the injectivity proof, this implies that $w = w_1 w_2 \ldots w_k$, $v = v_1 v_2 \ldots v_l$, $k \leq l$ and $h(w_i) = h(v_i)$ for every $1 \leq i \leq k$. Since h is injective, it follows that $w_i = v_i$ for all $i \leq k$ which implies $w \preceq v$.

Finally, we have to prove preservation of $\sqsubseteq$. For rational numbers q_1, q_2 we have $q_1 < q_2$ iff $h(q_1) \sqsubseteq h(q_2)$. From this it easily follows that for words

$w, w' \in Q^*$ $w \sqsubseteq w'$ if and only if $w \preceq w'$ or $w = viw_1$ and $w' = vjw_2$ for some $v \in \mathbb{Q}^*$ and some $i < j$ if and only if $g(w) \preceq g(w')$ or $g(w) = g(v)h(i)g(w_1)$ and $g(w') = g(v)h(j)g(w_2)$ with $h(i) \sqsubset h(j)$ if and only if $g(w) \sqsubseteq g(w')$. □

C Missing Proofs Concerning $\leq_C$

In this section we prove Lemma 9. Part 1 is proved in Lemma 25, Part 2 in Lemma 28 and Part 3 in Lemma 29.

C.1 Proof of Part 1

Lemma 25. *$\leq_C$ is a well-quasi order.*

Proof. Obviously, $\leq_C$ is a quasi order.

Any infinite sequence $(\bar{w}^i)_{i\in\mathbb{N}}$ of n-tuples of words induces an infinite sequence $(\bar{w}^i, C)_{i\in\mathbb{N}}$. The latter has an infinite subsequence $(\bar{w}^i, C)_{i\in I}$ such that for all $i, j \in I$ $\mathrm{typ}_C(\bar{w}^i) = \mathrm{typ}_C(\bar{w}^j)$. This implies that $\mathrm{MCAT}_C(\bar{w}^i)$ and $\mathrm{MCAT}_C(\bar{w}^j)$ are isomorphic for all $i, j \in I$ via an isomorphism $\phi_{i,j}$.

For every $i \in I$ we define a map $f_i : \mathrm{MCAT}_C(\bar{w}^i)^2 \to \mathbb{N}$ by $(u, v) \mapsto |u| - |u \sqcap v|$. Fix an $i_0 \in I$ and an enumeration of the domain of f_{i_0}. This induces an enumeration of the domain of f_i for every $i \in I$ by letting $(u, v) \in \mathrm{dom}(f_i)$ be the k-th element if $(\phi_{i,i_0}(u), \phi_{i,i_0}(v))$ is the k-th element of $\mathrm{dom}(f_{i_0})$.

By Dickson's Lemma we find tuples $\bar{w}^j$, $\bar{w}^k$ $(j < k)$ such that for all $(u, v) \in \mathrm{MCAT}_C(\bar{w}^j)$ $f_k(\phi_{j,k}(u), \phi_{j,k}(v)) \geq f_j(u, v)$. From this we immediately conclude that $\bar{w}^j \leq_C \bar{w}^k$. □

C.2 Proof of Part 2

We prepare the proof of strong upwards compatability of the transition relation by formally proving the following intuition: if $\mathrm{MCAT}_C(\bar{w}')$ has larger gaps than $\mathrm{MCAT}_C(\bar{w})$ (seen as subtrees of $\mathbb{Q}^*$), every extension of $\mathrm{MCAT}_C(\bar{w})$ to a bigger tree induces a corresponding extension of $\mathrm{MCAT}_C(\bar{w}')$ to a bigger tree of the same order type.

Definition 26. *For D, E, F sets with $D \subseteq E$, we say $h : E \to F$ extends $g : D \to F$ if $h \restriction_D = g$.*

Lemma 27. *Let $\sigma = \{\preceq, \sqsubseteq, \sqcap\}$ and $\bar{w}, \bar{w}' \in \mathbb{Q}^*$ be tuples such that $\bar{w} \leq_C \bar{w}'$. The isomorphism $h : \mathrm{MCAT}_C(\bar{w}) \to \mathrm{MCAT}_C(\bar{w}')$ extends to a σ-injection $f : (\mathbb{Q}^*, \preceq, \sqsubseteq) \to (\mathbb{Q}^*, \preceq, \sqsubseteq)$.*

Proof. In order to simplify the notation, we assume without loss of generality that $C \subseteq \bar{w}$. We define a family of σ-injections $f_j : \mathbb{Q}^{\leq j} \to \mathfrak{T}_\infty$ such that f_j extends $h \restriction_{M_j}$ where $M_j = \{ w \in \mathrm{MCAT}_C(\bar{w}) \mid |w| \leq \mathbb{Q}^{\leq j} \}$. Let $f_0 : \{\varepsilon\} \to \{\varepsilon\}$. Assume that f_j has been defined and satisfies that for all $\bar{v} \subseteq \bar{w}$ and all $u \in \mathbb{Q}^j$

1. $u \preceq \bigsqcap \bar{v}$ iff $f_j(u) \preceq h(\bigsqcap \bar{v})$ and
2. if $u \preceq \bigsqcap \bar{v}$ then $|\bigsqcap \bar{v}| - |u| \leq |f(\bigsqcap \bar{v})| - |f_j(u)|$.

For each word $u \in \mathbb{Q}^j$, we define the values of f_{j+1} on $u\mathbb{Q}$ according to the following rule. Let $\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_m \subseteq \bar{w}$ be those subsets such that for each i there is some $q_i \in \mathbb{Q}$ with $\bigsqcap \bar{v}_i = uq_i$. We can assume that $q_1 \leq q_2 \leq \cdots \leq q_m$. Note that the second condition on f_j implies that $f_j(u)$ and $h(\bigsqcap \bar{v}_i)$ have distance at least 1 whence there is some $q_i' \in \mathbb{Q}$ such that $f_j(u)q_i' \preceq h(\bigsqcap \bar{v}_i)$. We claim that for all $k, l \leq m$ we have $q_k \leq q_l$ if and only if $q_k' \leq q_l'$.

- If $q_k = q_l$ then $uq_k \preceq \bigsqcap \bar{v}_k \sqcap \bigsqcap \bar{v}_l = \bigsqcap(\bar{v}_k \cup \bar{v}_l)$. Thus, there is some i such that $\bigsqcap \bar{v}_i = \bigsqcap(\bar{v}_k \cup \bar{v}_l)$ and $q_i = q_k = q_l$. Then $f_j(u)q_i' \preceq h(\bigsqcap \bar{v}_i) \preceq h(\bigsqcap \bar{v}_k)$ and analogously for $\bar{v}_l$ whence $q_i' = q_k' = q_l'$.
- If $q_k < q_l$ then $\bigsqcap \bar{v}_k \sqcap \bigsqcap \bar{v}_l = u$. Thus, $u \in \mathrm{MCAT}_C(\bar{w})$ and $f_j(u) = h(u) = h(\bigsqcap \bar{v}_k) \sqcap h(\bigsqcap \bar{v}_l)$. Moreover, $\bigsqcap \bar{v}_k \sqsubset \bigsqcap \bar{v}_l$ whence $h(\bigsqcap \bar{v}_k) \sqsubset h(\bigsqcap \bar{v}_l)$. The only possibility to match both requirements is that $q_k' < q_l'$.

Fixing isomorphisms $g_i : \{\, q \in Q \mid q_i < q < q_{i+1} \,\} \to \left\{\, q \in Q \;\middle|\; q_i' < q < q_{i+1}' \,\right\}$ (with $q_0 = q_0' = -\infty$ and $q_{m+1} = q_{m+1}' = \infty$), we define for every $q \in Q$

$$f_{j+1}(uq) = \begin{cases} h(\bigsqcap v_i) & \text{if } q = q_i, \\ f_j(u)g_{i-1}(q) & \text{otherwise, where } q_i \in \{\, q_1, \ldots, q_m, q_{m+1} \,\} \text{ is minimal with } q < q_i. \end{cases}$$

Assuming that f_j preserves $\preceq, \sqsubseteq$, and $\sqcap$ in both directions, it is not difficult to prove the same result for f_{j+1}. Thus, the limit of $(f_j)_{j \in \mathbb{N}}$ is the desired σ-injection f. □

Proposition 28. *$\to$ and $\to^{-1}$ are strongly upwards compatible with respect to $\leq_C$.*

Proof. Given k-tuples $\bar{w}, \bar{v}, \bar{w}'$ and states q, p such that there is a transition $(q, \bar{w}) \to (p, \bar{v})$ and such that $\bar{w} \leq_C \bar{w}'$ we have to show that there is some $\bar{v} \leq_C \bar{v}'$ and a transition $(q, \bar{w}') \to (p, \bar{v}')$.

Since $\bar{w} \leq_C \bar{w}'$, the isomorphism $h : \mathrm{MCAT}_C(\bar{w}) \to \mathrm{MCAT}_C(\bar{w}')$ extends (by Lemma 27) to a $\{\preceq, \sqsubseteq, \sqcap\}$-injection $\hat{h} : \mathbb{Q}^* \to \mathbb{Q}^*$. Setting $v_i' = \hat{h}(v_i)$ for each $v_i \in \bar{v}$ we obtain with $\bar{v}' = (v_1', \ldots, v_k')$ that $(p, \bar{v}) \leq_C (p, \bar{v}')$ and $(q, \bar{w}') \to (p, \bar{v}')$ as desired.

The argument for $\to^{-1}$ is completely analogous. □

C.3 Proof of Part 3

Recall from Lemma 16 that insertion of an n-gap at some u which is not prefixed by a constant from C preserves the type and leads to a $\leq_C$ larger tuple. Iterated use of this lemma proves Part 3 of Lemma 9, which we restate in the following lemma.

Lemma 29. *Given two configurations $(q, \bar{w})$ and $(q, \bar{v})$ such that $\mathrm{typ}_C(\bar{w}) = \mathrm{typ}_C(\bar{v})$ then there is a configuration $(q, \bar{u})$ such that $(q, \bar{w}) \leq_C (q, \bar{u})$ and $(q, \bar{v}) \leq_C (q, \bar{u})$.*

Proof. Let $d \in \mathbb{N}$ be maximal such that there are $x_1, x_2 \in \mathrm{MCAT}_C(\bar{w})$ with $x_1 \preceq x_2$ and $|x_2| - |x_1| = d$. Inductively, from the $\preceq$-maximal elements to ε we insert a gap of size d at each $y \in \mathrm{MCAT}_C(\bar{v})$ if y is not prefixed by a constant from C. All these iterated insertions result finally in a tuple $\bar{u}$ such that $(q, \bar{v}) \leq_C (q, \bar{u})$ and for all $z_1, z_2 \in \mathrm{MCAT}_C(\bar{u})$ such that $z_1 \preceq z_2$ and z_2 is not prefix of any constant from C, then $|z_2| - |z_1| \geq d$. Thus, by definition of d also $(q, \bar{w}) \leq_C (q, \bar{u})$ holds as desired. □

D Computation of Types

The goal of this section is to prove Proposition 20, i.e., to provide an algorithm that checks whether a given type is realised by one of the runs of a given $\mathfrak{T}^C_\infty$-automaton. For this purpose we first fix an n-dimensional $\mathfrak{T}^C_\infty$-constraint automaton $\mathbb{A}$ with state set Q. We equip the power set of all types with a product operation as follows.

Definition 30. – *Let* $\mathsf{Typs}_{n,C}$ *denote the* set of all types (q, π, p) *where* $q, p \in Q$ *and* $\pi = \mathrm{typ}_C(\bar{w}, \bar{v})$ *where* $\bar{w}$ *and* $\bar{v}$ *are* n*-tuples of words.*

– *We equip the power set* $2^{\mathsf{Typs}_{n,C}}$ *with a* product $\cdot$ *as follows. For* $t = (q_1, \pi_1, p_1), u = (q_2, \pi_2, p_2)$, $v = (q_3, \pi_3, p_3) \in \mathsf{Typs}_{n,C}$ *let* $t \in \{ u \} \cdot \{ v \}$ *if*

1. $q_1 = q_2$, $p_1 = p_3$, $p_2 = q_3$, *and*
2. *there are* n*-tuples* $\bar{x}, \bar{y}, \bar{z}$ *such that* $\mathrm{typ}_C(\bar{x}, \bar{y}) = \pi_2$, $\mathrm{typ}_C(\bar{y}, \bar{z}) = \pi_3$ *and* $\mathrm{typ}_C(\bar{x}, \bar{z}) = \pi_1$.

Generally, for $A, B \subseteq \mathsf{Typs}_{n,C}$ *such that at least one of them is not a singleton, we define* $A \cdot B = \{ t \cdot u \mid t \in A, u \in B \}$.

– *The* set of types of one-step runs $T_1 \subseteq \mathsf{Typs}_{n,C}$ *is given by* $t = (q, \pi, p) \in T_1$ *if there is a transition* (q, β, p) *of* $\mathbb{A}$ *such that* π *satisfies* β.

– *Let* $T_1^1 = T_1$, $T_1^{n+1} = T_1^n T_1$, *and* $T_1^+ = \bigcup_{n \geq 1} T_1^n$.

Remark 31. One easily checks that $t \in T_1$ holds if and only if there is a run of length 1 with type t.

The product operation resembles the composition of types. As a consequence one can connect the runs of $\mathbb{A}$ and T_1^+ as follows.

Lemma 32. *There is a run of* $\mathbb{A}$ *of type* t *if and only if* $t \in T_1^+$.

Before we provide a proof, we show how this lemma can be used to prove 20 which we restate here:

Proposition 33. *There is a* PSPACE*-algorithm that, given an* n*-dimensional* $\mathfrak{T}^C_\infty$*-constraint automaton* $\mathbb{A}$ *and a type* t*, determines whether there is a run of* $\mathbb{A}$ *of type* t.

Proof. Writing $m = \max(|c| : c \in C)$ the algorithm uses polynomial space in terms of $m + n + |\mathbb{A}|$.[4] Given n-tuples $\bar{w}$ and $\bar{v}$, note that $\mathrm{typ}_C(\bar{w}, \bar{v})$ contains at most $4n$ elements that are not constants. Thus, we can represent any type by 2 states and $2n$ words of length at most $m + 4n$. Moreover, It takes logarithmic space in n and $|A|$ to check whether a given type satisfies a specific transition. Finally, it only needs $O(2n(m + (4n \cdot 2n)))$ space to decide whether a given type t is in the product of two types t_1, t_2 (cf. the upcoming Lemma 38).

Thus, an NPSPACE (= PSPACE) algorithm can guess a first type $t_1 \in T_1$ and, having stored a type $t_i \in T_1^i$, it can guess another type $t \in T_1$ and a type t_{i+1} and verify that $t_{i+1} \in \{ t_i \} \cdot \{ t \}$. This procedure is iterated until t_i is the desired type and the algorithm reports that t_i can be realised by some run. □

D.1 Proof of Lemma 32

We finally have to prove the connection between composition of runs and products of their types. One direction is easily shown and contained in the following lemma.

Lemma 34. *For $r = (c_i)_{1 \le i \le n}$ a run (with $n \ge 2$), $\mathrm{typ}(r) \in T_1^{n-1}$.*

Proof. For $n = 2$ the claim follows by definition of $T_1^{2-1} = T_1$. We proceed by induction. Write $c_i = (q_i, w_1^i, \ldots, w_k^i)$. Let $r' = (c_i)_{1 \le i \le n-1}$ and $r_{n-1} = (c_i)_{n-1 \le i \le n}$. By induction hypothesis $\mathrm{typ}(r') = (q_1, \pi, q_{n-1}) \in T_1^{n-2}$ with

$$\pi = \mathrm{typ}_C(w_1^1, w_2^1, \ldots, w_k^1, w_1^{n-1}, \ldots, w_k^{n-1}),$$

and $\mathrm{typ}(r_{n-1}) = (q_{n-1}, \pi_{n-1}, q_n) \in T_1$ with

$$\pi_{n-1} = \mathrm{typ}_C(w_1^{n-1}, \ldots, w_k^{n-1}, w_1^n, \ldots, w_k^n).$$

Thus, the tuples $w_1^1, \ldots, w_k^1$, $w_1^{n-1}, \ldots, w_k^{n-1}$, $w_1^n, \ldots, w_k^n$ witness that

$$(q_1, \pi', q_n) := \mathrm{typ}(r) \in \mathrm{typ}(r') \cdot \mathrm{typ}(r_{n-1}) \subseteq T_1^{n-2} \cdot T_1 = T_1^{n-1},$$

which completes the proof. □

The other direction of Lemma 32 relies on the following intuition.

1. By upwards-compatability and gap-insertion every type realised by some run, is realised by one with large gaps between all pairs of elements except the constants.
2. If two n-tuple $\bar{w}, \bar{v}$ have $2n$-gaps between all pairs of elements from $\mathrm{MCAT}_C(\bar{w}, \bar{v})$ except the constants, then for every type $t \in \mathrm{typ}_C(\bar{w}, \bar{v}) \cdot T_1$ there is a tuple $\bar{u}$ such that $\bar{w}, \bar{v}, \bar{u}$ witness this inclusion.
3. Thus, assuming that all types from T_1^{n-1} are realised by runs, for all $t \in T_1^{n-1} \cdot T_1$ we can realise the appropriate type from T_1^{n-1} with a run r that has large gaps at its last configuration and find a witness for t by realising the appropriate type from T_1 using the values of the last configuration of r.

[4] Assuming any reasonable notion of size of an automaton.

Proving these intuitions is rather tedious and we give the details in the following. Recall that we assume that the set of constants C is closed under prefixes. Let us first make precise what a gap is.

Definition 35. *We say that a tree $T \subseteq \mathbb{Q}^*$ has n-gaps above C if for all $d, e \in T$ with $d \prec e$ such that $e \not\preceq c$ for all $c \in C$ we have $|e| - |d| > n$.*

We can now give a precise version of the first claim.

Lemma 36. *Given a finite run r there is a run r' from $c_1' = (q, \bar{w}')$ to $c_2 = (p', \bar{v}')$ of the same type such that $\mathrm{MCAT}_C(\bar{w}', \bar{v}')$ has $2n$-gaps above C.*

Proof. Let r be a run from $(q, \bar{w})$ to $(q, \bar{v})$ For each $u \in \mathrm{MCAT}_C(\bar{w}, \bar{v})$ (starting with $\preceq$-maximal ones) that is not a constant from C, we insert a gap of size $2n$ at u in r . Since gap insertion preserves types (Lemma 16), the resulting run r' from $(q, \bar{w}')$ to $(p, \bar{v}')$ is of the same type as r and $\mathrm{MCAT}_C(\bar{w}', \bar{v}')$ has $2n$-gaps above C. □

For the second claim we need a technical lemma first and then prove the second intuition to be correct.

Lemma 37. *Let $\sigma = \{ \preceq, \sqsubseteq, \sqcap \}$, $n \in \mathbb{N}$. Let $A \subseteq \mathbb{Q}^*$ be some finite set closed under maximal common prefixes such that $\varepsilon \in A$. Let $B \subseteq A$ and $h : A \to \mathfrak{T}_\infty$ a σ-injection such that $h(A)$ has n-gaps above $h(B)$. Given $D \subseteq \mathbb{Q}^*$ such that*

1. *$|D \setminus A| \leq n$,*
2. *$D \cup A$ is closed under maximal common prefixes, and*
3. *there is no $d \in D$ and $b \in B$ such that $d \preceq b$,*

then h extends to a σ-injection $h_D : A \cup D \to \mathfrak{T}_\infty$.

Proof. The base case $n = 0$ is trivial. Assume that the lemma has been proven for some $n \in \mathbb{N}$. If $|D \setminus A| = n + 1$, let $d \in D \setminus A$ be $\sqsubseteq$-minimal. By induction hypothesis it suffices to extend h to a σ-injection $h' : A \cup \{ d \} \to \mathfrak{T}_\infty$ that has n-gaps above $h(B \cup \{ d \})$. We first define the image of d by a case distinction and prove that the resulting map h' has the desired properties. We distinguish two cases.

1. Assume that there is some $a \in A$ such that $d \preceq a$. Since $\varepsilon \in A$ we find a maximal $w \in A$ such that $w \prec d$. Moreover, $\bar{a} = \bigsqcap \{ a \in A \mid d \preceq a \}$ is well defined and satisfies $w \prec \bar{a}$. Thus, $h(w) \prec h(\bar{a})$ and there is a $q \in Q$ such that $h(w)q \preceq h(\bar{a})$. Let h' be the extension of h to $A \cup \{d\}$ mapping $h'(d) = h(w)q$ and $h'(a) = h(a)$ for all $a \in A$.
2. Otherwise, there is no $a \in A$ with $d \preceq a$. Let again $w \in A$ be maximal with $w \preceq d$ and let $q_d \in \mathbb{Q}$ such that $wq_d \preceq d$. For later use we first establish that

$$\text{there is no } a \in A \text{ with } wq_d \preceq a. \tag{1}$$

   Assuming the contrary let $wq_d \preceq a$. Since $A \cup D$ is closed under maximal common prefixes, we conclude that $wq_d \preceq (a \sqcap d) \in A \cup D$. $(a \sqcap d) \in A$

contradicts the maximality of w. But due to $\sqsubseteq$-minimality of d, $(a \sqcap d) \in D \setminus A$ is only possible if $d = a \sqcap d$ which implies $d \preceq a$ which contradicts our assumption on d.
We define a partition of $\{\, a \in A \mid w \prec a \,\}$ by setting

$$A^- = \{\, a \in A \mid w \prec a \text{ and } a \sqsubseteq d \,\} \text{ and}$$
$$A^+ = \{\, a \in A \mid w \prec a \text{ and } d \sqsubseteq a \,\}.$$

If $A^- \neq \emptyset$ let a^- be its $\sqsubseteq$-maximal element. Since h preserves $\prec$, there is some $q^- \in \mathbb{Q}$ such that $h(w)q^- \preceq h(a^-)$. If $A^- = \emptyset$ set $q^- = -\infty$. Analogously, if $A^+ \neq \emptyset$ let a^+ be its $\sqsubseteq$-minimal element. Since h preserves $\prec$, there is some $q^+ \in \mathbb{Q}$ such that $h(w)q^+ \preceq h(a^+)$. If $A^- = \emptyset$ set $q^+ = \infty$.
If a^- and a^+ are both defined, we conclude with (1) that there are $q_1 < q_d < q_2$ such that $wq_1 \preceq a^-$ and $wq_2 \preceq a^+$. Since h is a σ-injection, we directly conclude that $q^- < q^+$.
Choose $q \in (q^-, q^+)$ arbitrarily and define the map $h' : A \cup \{\, d \,\} \to \mathfrak{T}_\infty$ by $h'(a) = h(a)$ for all $a \in A$ and $h'(d) = h(w)q$.
We prepare the proof that h' is a σ-injection by establishing that

$$\text{for all } p \in (q^-, q^+) \text{ there is no } a \in A \text{ such that } h(w)p \preceq h(a). \tag{2}$$

Heading for a contradiction assume that there was such a and note that $h(a^-) \sqsubset h(a) \sqsubset h(a^+)$ and $h(w) \prec h(a)$. This would imply $a^- \sqsubset a \sqsubset a^+$ and $w \prec a$. But this clearly contradicts the definitions of a^- and a^+ as maximal below d (minimal above d, respectively).

We claim that the resulting map h' is a σ-injection.
Injectivity: Heading for a contradiction, assume that there is an $a \in A$ with $h(a) = h(w)q$ then $h(w) \prec h(a)$ which implies $w \prec a$. But then either $w \prec a \preceq d$ violates the choice of w or $d \preceq a$. In the latter case the third condition on D implies that there is no $b \in B$ with $a \preceq b$. But then $h(w)$ and $h(a)$ need to have an $(n+1)$-gap which is not the case. Thus, we have arrived at a contradiction and conclude that there is no $a \in A$ with $h(a) = h(w)q$ whence h' is injective.
Preservation of $\preceq$: We show that h' preserves $\preceq$ in both directions. Choose some $a \in A$.

- If $a \preceq d$ then by choice of w we have $a \preceq w$ whence $h'(a) = h(a) \preceq h(w) \prec h'(d)$.
- If $h'(a) = h(a) \preceq h'(d) = h(w)q$, then $h(a) \preceq h(w)$ because h' is injective. Thus, $a \preceq w \prec d$ as desired.
- If $d \preceq a$ we are in case one of the definition of h'. Thus, $\bar{a} \preceq a$ whence by definition $h'(d) \preceq h(\bar{a}) \preceq h(a) = h'(a)$.
- If $h'(d) = h(w)q \preceq h(a)$, we conclude with (2) that we are in case one of the definition of h'. Thus, $h(w)q \preceq h(\bar{a}) \preceq h(a)$ implies that $h(w)q \preceq h(a) \sqcap h(\bar{a}) = h(a \sqcap \bar{a})$. Since h is a σ-injection, it follows that $w \prec a \sqcap \bar{a} \preceq \bar{a}$. Since $d \preceq \bar{a}$, we obtain that $a \sqcap \bar{a}$ and d are comparable. By maximality of w, we conclude $d \preceq (a \sqcap \bar{a}) \preceq a$.

Preservation of $\sqsubseteq$: Due to the $\preceq$ preservation, it suffices to prove preservation of $\sqsubset \cap \not\preceq$. Again choose some $a \in A$.

- Assume that $a \sqsubseteq d$ and $a \not\preceq d$. If $a \sqsubseteq w$ we immediately conclude that $h'(a) = h(a) \sqsubseteq h(w) \sqsubseteq h(w)q = h'(d)$. Otherwise, one immediately concludes that $d \sqcap a = w$.
  1. If h' has been defined in case one, we immediately conclude $a \sqcap \bar{a} = w$ and $a \sqsubset \bar{a}$ whence $h(a) \sqcap h(\bar{a}) = h(a \sqcap \bar{a}) = h(w)$ and $h(a) \sqsubset h(\bar{a})$. Since $h(w) \prec h'(d) \preceq h(\bar{a})$, it follows that that $h'(a) = h(a) \sqsubset h(w)$.
  2. Otherwise, h' has been defined in the second case and we conclude that $a \in A^-$ whence $a \sqsubseteq a^-$. This implies that $h'(a) = h(a) \sqsubseteq h(a^-) \sqsubseteq h(w)q = h'(d)$.
- Assume that $d \sqsubseteq a$ and $d \not\preceq a$. First assume that $w \not\preceq a$. Then $d \sqcap a = w \sqcap a \prec w$ whence $w \sqsubseteq a$. Since h is a σ-injection, we obtain $h(w) \sqsubseteq h(a)$, and $h(w) \sqcap h(a) = h(w \sqcap a) \prec h(w)$. Thus, $h(w) \preceq h'(d)$ directly implies $h'(d) \sqsubseteq h(a) = h(a')$. Otherwise, we have $w \preceq a$. Since $d \sqsubseteq a$ we conclude that $w \prec a$.
  1. If h' has been defined in case one, $d \not\preceq a$, $w \prec a$ and maximality of w imply that $w = d \sqcap a = \bar{a} \sqcap a$. Since $\bar{a}$ and d are on a common path, we also have $\bar{a} \sqsubseteq a$. Thus, $h(w) = h(\bar{a} \sqcap a) = h(\bar{a}) \sqcap h(a)$ and $h(\bar{a}) \sqsubseteq h(a)$. Since $h'(d)$ and $h(\bar{a})$ are on a common path, we obtain $h'(d) \sqsubseteq h(a) = h'(a)$.
  2. Otherwise, h' has been defined in case two. Then $w \prec a$ and $d \sqsubseteq a$ imply $a^+ \sqsubseteq a$. We conclude by choice of q that $h'(d) = h(w)q \sqsubseteq h'(a^+) \sqsubseteq h(a)$.

  Since $\sqsubseteq$ is a total order, the backwards preservation of $\sqsubseteq$ follows directly from the forward preservation: assume $h'(x) \sqsubseteq h'(y)$, then forwards preservation and injectivity rules out the case $y \sqsubset x$, whence $x \sqsubseteq y$ because $\sqsubseteq$ is total.

Preservation of $\sqcap$: Finally, note that h' preserves $\sqcap$ in both directions. Let $a \in A$. If a and d are comparable, the claim follows from the preservation of $\preceq$. Otherwise, if a and d are incomparable (with respect to $\preceq$), then we conclude $a \sqcap d \in A$ whence $a \sqcap d = a \sqcap w$. But then also $h'(a)$ and $h'(d)$ are incomparable whence $h'(a) \sqcap h'(d) \preceq h'(w)$ whence by definition of $h'(d)$ we have $h'(a) \sqcap h'(d) = h'(a) \sqcap h'(w) = h(a) \sqcap h(w) = h(a \sqcap w) = h'(a \sqcap w) = h'(a \sqcap d)$. □

Lemma 38. *Let $\bar{w}, \bar{v}$ be n-tuples and $t = (q, \pi, r), t_1 = (q, \pi_1, p), t_2 = (p, \pi_2, r) \in \mathsf{Typs}_{n,C}$ such that $\mathrm{typ}_C(\bar{w}, \bar{v}) = \pi_1$, and $\mathrm{MCAT}_C(\bar{w}, \bar{v})$ has $(2n)$-gaps above C. There is an n-tuple $\bar{u}$ such that $\mathrm{typ}_C(\bar{v}, \bar{u}) = \pi_2$ and $\mathrm{typ}_C(\bar{w}, \bar{u}) = \pi$.*

Proof. By definition of the product, there are k-tuples $\bar{x}, \bar{y}, \bar{z}$ such that $\mathrm{typ}_C(\bar{x}, \bar{y}) = \pi_1$, $\mathrm{typ}_C(\bar{y}, \bar{z}) = \pi_2$ and $\mathrm{typ}_C(\bar{x}, \bar{z}) = \pi$. Fix the isomorphism $h : \mathrm{MCAT}_C(\bar{x}, \bar{y}) \to \mathrm{MCAT}_C(\bar{w}, \bar{v})$. One shows by induction on n that if $\mathrm{MCAT}_C(\bar{x}, \bar{y})$ has $n_1 \in \mathbb{N}$ many leaves and $n_2 \in \mathbb{N}$ many inner nodes then $\mathrm{MCAT}_C(\bar{x}, \bar{y})$ has at most $n_1 + n$ leaves and $n_2 + n$ inner nodes whence $|\mathrm{MCAT}_C(\bar{x}, \bar{y}, \bar{z}) \setminus \mathrm{MCAT}_C(\bar{x}, \bar{y})| \leq 2n$. Thus, h extends by Lemma 37 (setting $A = \mathrm{MCAT}_C(\bar{x}, \bar{y})$, $B = C$, $D = \mathrm{MCAT}_C(\bar{x}, \bar{y}, \bar{z}) \setminus \mathrm{MCAT}_C(\bar{x}, \bar{y})$, and seeing h as an injection $A \to \mathfrak{T}_\infty$) to a $\{\preceq, \sqsubseteq, \sqcap\}$-injection $\hat{h} : \mathrm{MCAT}_C(\bar{x}, \bar{y}, \bar{z}) \to \mathfrak{T}_\infty$ (which is the identity on all all constants from C) such that for $\bar{u} = \hat{h}(\bar{z})$, $\mathrm{typ}_C(\bar{w}, \bar{v}, \bar{u}) = \mathrm{typ}_C(\bar{x}, \bar{y}, \bar{z})$. In particular, $\mathrm{typ}_C(\bar{v}, \bar{u}) = \pi_2$ and $\mathrm{typ}_C(\bar{w}, \bar{u}) = \pi$ as desired. □

Now we are prepared to prove the last direction of Lemma 32

Lemma 39. *For every $t \in T_1^+$ there is a run r of $\mathbb{A}$ with* $\mathrm{typ}(r) = t$.

Proof. As remarked before, for $t \in T_1^1 = T_1$ there is nothing to show. Let $r \in T_1^{n+1}$ and assume the claim is true for all $t \in T_1^n$. Let $t \in t_1 \cdot t_2$ with $t_1 \in T_1^n$ and $t_2 \in T_1$ and let r' be a run of type t_1. Let $c_0 = (q, \bar{w})$ be the first and $c_1 = (p, \bar{v})$ the last configuration of r'. By Lemma 36, we can assume that $\mathrm{MCAT}_C(\bar{w}, \bar{v})$ has $2n$-gaps. Thus, by Lemma 38, there is tuple $\bar{u}$ and a state q' such that $(p, \mathrm{typ}_C(\bar{v}, \bar{u}), q') = t_2$ and $(q, \mathrm{typ}_C(\bar{w}, \bar{u}), q') = t$. Thus, extending r' by configuration $(q', \bar{u})$ results in the desired run r. □