On Constructing Constrained Tree Automata Recognizing Ground Instances of Constrained Terms

Naoki Nishida 1, Masahiko Sakai 2 and Yasuhiro Nakano 3

Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 4648603, Japan
{nishida@, sakai@, ynakano@trs.cm.}is.nagoya-u.ac.jp

Abstract

An inductive theorem proving method for constrained term rewriting systems, which is based on rewriting induction, needs a decision procedure for reduction-completeness of constrained terms. In addition, the sufficient complete property of constrained term rewriting systems enables us to relax the side conditions of some inference rules in the proving method. These two properties can be reduced to intersection emptiness problems related to sets of ground instances for constrained terms. This paper proposes a method to construct deterministic, complete, and constraint-complete constrained tree automata recognizing ground instances of constrained terms.

1 Introduction

The constrained rewriting in this paper is a computation model that rewrites a term by applying a constrained rewriting rule if the term satisfies the constraint on interpretable domains attached to the rule [5, 1, 10, 9, 2]. Proposed a rewriting induction method on the constrained rewriting [10, 7], a decision procedure of reduction-completeness of terms must be extended for constrained terms, terms admitting constraints attached, in order to apply the method to mechanical inductive theorem proving, where a term is said to be reduction-complete [4] if any ground instance of the term is a redex. If a constrained term rewriting system is terminating and all terms are reduction-complete, the rewrite system is said to be sufficient complete, which is useful to relax the application conditions of inference rules in the above method [10]. These properties are proved by using tree automata with constraints, whose rules contain constraints on interpretable subterms. More precisely, the properties are reducible to the intersection emptiness problem of ground instances of terms satisfying constraints attached to the terms.

This paper proposes a construction method of constrained tree automata that accept ground instances of constrained term (in Section 3). Moreover the obtained tree automata have nice properties: the constraint-completeness [8], completeness and determinacy, where the first property is necessary for proving correctness of the constructed tree automata, and the next two properties contribute avoiding size explosion at the construction of product automata.

2 Preliminaries

In this section, we briefly recall the basic notions of terms [3], constraints over predicate logic [5], and constrained tree automata [8].

Throughout the paper, we use \( V \) as a countably infinite set of variables. For a signature \( F \) (a finite set of function symbols with fixed arities), the set of terms over \( F \) and \( X \subseteq V \) is
denoted by $T(\mathcal{F}, X)$. The notation $p/n$ represents the function symbol $p$ with the arity $n$. The set $T(\mathcal{F}, \emptyset)$ of ground terms is abbreviated to $T(\mathcal{F})$. The set of variables appearing in term $t$ is denoted by $\text{Var}(t)$. A term is called linear if any variable appears in the term at most once. The set of positions for term $t$ is denoted by $\text{Pos}(t)$: $\text{Pos}(x) = \{x\}$ for $x \in \mathcal{V}$, and $\text{Pos}(f(t_1, \ldots, t_n)) = \{x\} \cup \{i \cdot \pi \mid 1 \leq i \leq n, \pi \in \text{Pos}(t_i)\}$ for $f/n \in \mathcal{F}$. For terms $t, u$ and a position $\pi$ of $t$, the notation $t|u,\pi$ denotes the term obtained from $t$ by replacing the subterm $t|\pi$ of $t$ at $\pi$ with $u$. For a substitution $\theta$, we denote the range of $\theta$ by $\text{Ran}(\theta)$.

Let $\mathcal{G}$ be a signature and $\mathcal{P}$ be a set of predicate symbols with fixed arities. The notation $p/n$ represents the predicate symbol $p$ with the arity $n$. First-order (quantifier-free) formulas $\phi$ over $\mathcal{G}$, $\mathcal{P}$, and $\mathcal{V}$ are defined in BNF as follows: $\phi ::= p(t_1, \ldots, t_n) \mid \top \mid \bot \mid (\neg \phi) \mid (\phi \lor \psi) \mid (\phi \land \psi)$, where $p/n \in \mathcal{P}$ and $t_1, \ldots, t_n \in T(\mathcal{G}, \mathcal{V})$. We may omit brackets "(", ")" from formulas as usual. The set of first-order formulas over $\mathcal{G}$, $\mathcal{P}$, and $X \subseteq \mathcal{V}$ is denoted by $\text{Fol}(\mathcal{G}, \mathcal{P}, X)$. For a formula $\phi$, the notation $\text{Var}(\phi)$ represents the set of variables in $\phi$. A variable-free formula is called closed. A structure for $\mathcal{G}$ and $\mathcal{P}$ is a tuple $\mathcal{S} = (A, I_\mathcal{G}, I_\mathcal{P})$ such that the universe $A$ is a non-empty set of concrete values, $I_\mathcal{G}$ and $I_\mathcal{P}$ with types $\mathcal{G} \rightarrow \{f \mid f$ is an $n$-ary function on $A\}$ and $\mathcal{P} \rightarrow 2^{A^{\times \cdot \cdot \cdot \times A}}$ are interpretations for $\mathcal{G}$ and $\mathcal{P}$, resp.: $I_\mathcal{G}(g)(a_1, \ldots, a_n) \in A$ for $g/n \in \mathcal{G}$, and $I_\mathcal{P}(p) \subseteq A^n$ for $p/n \in \mathcal{P}$. The interpretation of formulas $\phi$ w.r.t. $\mathcal{S}$, denoted by $\mathcal{S} \models \phi$, is defined as usual. We say that a formula $\phi$ holds w.r.t. $\mathcal{S}$ if $\mathcal{S} \models \phi$. Formulas in $\text{Fol}(\mathcal{G}, \mathcal{P}, \mathcal{V})$ interpreted by $\mathcal{S}$ are called constraints (w.r.t. $\mathcal{S}$).

In the following, we use $\mathcal{F}, \mathcal{G}$ for signatures, $\mathcal{P}$ for a set of predicate symbols, and $\mathcal{S} = (A, I_\mathcal{G}, I_\mathcal{P})$ for a structure for $\mathcal{G}$ and $\mathcal{P}$, with notice. Before formalizing constrained tree automata, we generalize the interpretation of constraints under terms. For a sequence $\pi$ of natural numbers, the notation $(\pi)$ denotes the special variable related to $\pi$. We denote the set of such variables by $\langle N^* \rangle$: $\langle N^* \rangle = \{\langle \pi \rangle \mid \pi \in N^* \} \subseteq \mathcal{V}$. A formula $\phi$ in $\text{Fol}(\mathcal{G}, \mathcal{P}, \langle N^* \rangle)$ holds w.r.t. $\mathcal{S}$ if $\mathcal{S} \models \phi$, where $\models$ is inductively defined as follows: $\mathcal{S}, t \models \top$; $\mathcal{S}, t \models p(t_1, \ldots, t_n)$ if $\pi \in \text{Pos}(t)$ and $t|\pi \in T(\mathcal{G})$ for all variables $\langle \pi \rangle \in \bigcup_{i=1}^n \text{Var}(t_i)$, and $(I_\mathcal{G}(t_1\theta), \ldots, I_\mathcal{G}(t_n\theta)) \in I_\mathcal{P}(p)$, where $p/n \in \mathcal{P}$ and $\theta$ is the substitution $\{\langle \pi \rangle \mapsto t|\pi \mid (\pi) \in \bigcup_{i=1}^n \text{Var}(t_i)\}$; otherwise $\mathcal{S}, t \not\models p(t_1, \ldots, t_n)$; the relation $\models$ is defined as usual for any Boolean connective. We may omit $\mathcal{S}$ from "$\mathcal{S}, t \models \phi$" if it is clear in context. Note that $\mathcal{S} \models \neg \psi$ does not coincide with $\mathcal{S} \not\models \phi$ for every $\phi$ (see [8]).

By generalizing AWEDCs [8], constrained tree automata are defined as follows [8]. A constrained tree automata (CTA) over $\mathcal{F}$, $\mathcal{G}$, $\mathcal{P}$, and $\mathcal{S}$ is a tuple $\mathcal{A} = (Q, Q_f, \Delta)$ where $Q$ is a finite set of states (unary symbols), $Q_f$ is a finite set of final states (i.e., $Q_f \subseteq Q$), and $\Delta$ is a finite set of constrained transition rules 2 of the form $f(q_1(x_1), \ldots, q_n(x_n)) \xrightarrow{\phi} q(f(x_1, \ldots, x_n)) \in \Delta$ where $f/n \in \mathcal{F} \cup \mathcal{G}$, $q_1, \ldots, q_n, q \in Q$ and $\phi \in \text{Fol}(\mathcal{G}, \mathcal{P}, \langle N^* \rangle)$. We often omit the arguments of states by writing $q$ instead of $q(t)$, and then transition rules are written in the form $f(q_1, \ldots, q_n) \xrightarrow{\phi} q$. We also may omit $\top$ from $f(q_1, \ldots, q_n) \xrightarrow{\top} q$. The move relation $\rightarrow_{\mathcal{A}}$ is defined as follows: $t \rightarrow_{\mathcal{A}} u$ if $t$ has no nest of state symbols, $t|\pi = f(q_1(t_1), \ldots, q_n(t_n)), f/n \in \mathcal{F} \cup \mathcal{G}$, $t_1, \ldots, t_n \in T(\mathcal{F} \cup \mathcal{G}), f(q_1, \ldots, q_n) \xrightarrow{\phi} q \in \Delta$, and $t|\pi \models \phi$, and $u = \{q(f(t_1, \ldots, t_n))\}|_{\pi}$. The terminologies of CTAs are defined analogously to those of tree automata, except for determinism and completeness. $\mathcal{A}$ is called deterministic if for every ground term $t$, there is at most one state $q \in Q$ such that $t \rightarrow_{\mathcal{A}} q$. $\mathcal{A}$ is called complete if for every ground term $t$, there is at least one state $q \in Q$ such that $t \rightarrow_{\mathcal{A}} q$. Note that AWEDCs [8] are CTAs.

1It is possible to allow to introduce quantifiers. To be more precise, introduction of quantifiers to our setting does not affect the results. Though, for the sake of readability, we do not introduce them here.

2We consider transition rules $t \xrightarrow{\phi} r$ and $t \xrightarrow{\psi} r$ to be equivalent if $\text{Var}(\phi) = \text{Var}(\psi)$ and $\phi$ is semantically equivalent to $\psi$ (i.e., $\phi \leftrightarrow \psi$ is valid w.r.t. $\mathcal{S}$).
Example 1. Let $F = \{f/2\}$, $G_{\text{int}} = \{s/1, p/1, 0/0\}$, $P = \{\neg, \neq, \leq, <\}$, and $S_{\text{int}}$ be the structure $(\mathbb{Z}, I_{G_{\text{int}}}, T_{X_{\text{int}}})$ for $G_{\text{int}}$ and $P_{\text{int}}$ such that $I_{G_{\text{int}}}(s)(x) = x + 1$, $I_{G_{\text{int}}}(p)(x) = x - 1$, $I_{G_{\text{int}}}(0) = 0$, and $=, \neq, \leq, <$ are interpreted over integers as usual. Consider a CTA $A_{\text{int}} = \{(q_1, q_2), \{q_2\}, \Delta\}$ over $F$, $G_{\text{int}}$, $P_{\text{int}}$, and $S_{\text{int}}$ where

$$\Delta = \{0 \rightarrow q_1, \ s(q_1) \rightarrow q_1, \ p(q_1) \rightarrow q_1, \ f(q_1, q_1) \overset{(1.1) \leq p(2)}{\rightarrow} q_2, \ f(q_1, q_1) \rightarrow q_1\}.$$  

The term $f(s(0), s(0))$ is not accepted by $A_{\text{int}}$ since both $f(s(0), s(0)) \overset{\ast}{\rightarrow}_{A_{\text{int}}} f(q_1, s(0)), q_1(s(0))$ and $f(s(0), s(0)) \models (1.1) \leq p(2)$ hold, and thus, we have that $f(q_1, s(0)), q_1(s(0)) \not\rightarrow_{A_{\text{int}}} q_2$. The term $f(s(0), s(0))$ also transitions into $q_1$, and thus, $A_{\text{int}}$ is not deterministic. On the other hand, the term $f(0, s(0))$ is not accepted by $A_{\text{int}}$ since $f(0, s(0)) \not\models (1.1) \leq p(2)$. Note that $A_{\text{int}}$ recognizes the set of instances of the constrained terms $f(s(x), y)_{|x \leq p(y)}$, $f(p(x), y)_{|x \leq y}$, and $f(f(x, z), y)_{|x \leq p(y)}$, i.e., $L(A_{\text{int}}) = \{f(s(t_1), t_2), f(p(t_1), t_2), f(f(t_1, t_2), t_2) | t \in T(F \cup G_{\text{int}}), t_1, t_2 \in T(G_{\text{int}}), t \leq I_{G_{\text{int}}}(p(t_2))\}$. Consider the CTA $A'_{\text{int}}$ obtained from $A_{\text{int}}$ by replacing $(1.1) \leq p(2)$ by $\neg p(2) < (1.1)$. The constraints $x \leq p(y)$ and $\neg p(y) < x$ are semantically equivalent, i.e., for all terms $t_1, t_2 \in T(G_{\text{int}})$, $S_{\text{int}} \models t_1 \leq t_2$ iff $S_{\text{int}} \models \neg p(t_2) < t_1$. However, this is not the case for similar constraints over fixed terms, e.g., $f(0, s(0)) \models (1.1) \leq p(2)$ does not hold, but $f(0, s(0)) \models \neg p(2) < (1.1)$ holds. Thus, $f(0, s(0))$ is accepted by $A'_{\text{int}}$, and hence $L(A_{\text{int}}) \neq L(A'_{\text{int}})$. 

A CTA $A = (Q, Q_f, \Delta)$ is called constraint-complete [3] if for every ground term $t \in T(F \cup G)$ and all transition rules $f(q_1, \ldots, q_n) \overset{\varnothing}{\rightarrow} q \in \Delta$ with $t = f(t_1, \ldots, t_n) \overset{\varnothing}{\rightarrow} f(q_1, \ldots, q_n)$, we have that $\pi \in P_{\text{int}}(t)$ and $t_\pi \in T(G_{\text{int}})$ for all variables $\pi$ in $\varnothing$. Note that every CTA can be transformed into a deterministic, complete, and constraint-complete CTA [3]. Note also that completeness and constraint-completeness are different.

### 3 Recognizing Ground Instances of Constrained Terms

This section defines constrained terms and their instances, and then proposes a method for constructing a CTA recognizing the set of ground instances for a given set of constrained terms. The method is based on the construction of a tree automaton recognizing the set of ground instances for unconstrained terms, which is complete and deterministic [3, Exercise 1.9]. Accessibility of states is in general undecidable, and thus, it is difficult to get rid of inaccessible states which affect the intersection emptiness problem. In this sense, it is worth developing a construction method that introduces inaccessible states as little as possible.

**Definition 2.** A constrained term is a pair $(t, \phi)$, written as $t_{[\phi]}$, of a linear term $t \in T(F \cup G, V)$ and a formula $\phi \in \text{Fol}(G, P, \text{Var}(t))$. The set of ground instances of a constrained term $t_{[\phi]}$, denoted by $G(t_{[\phi]})$, is defined as follows: $G(t_{[\phi]}) = \{t \phi \in T(F \cup G) | \text{Ran}(t_{[\phi]}(V)) \subseteq T(G), S \models \phi \}$. The argument of $G$ is naturally extended to sets $G(T) = \bigcup_{t_{[\phi]} \in T} G(t_{[\phi]})$. 

To deal with constrained terms, we consider constrained patterns. We introduce wildcard symbols $\square$ and $\blacksquare$ to denote arbitrary interpretable and un-interpretable terms resp.

**Definition 3.** Let $t_{[\phi]}$ be a constrained term. Then $\text{Rep}_{\blacksquare}(t_{[\phi]})$ is the set of terms $t\phi_{[\varnothing]}$ where

- $\varnothing = \{x \mapsto \square \ | \ x \in \text{Var}(t) \cap FV(\phi)\} \cup \bigcup_{y \in \text{Var}(t) \setminus FV(\phi)} \{y \mapsto v \ | \ v \text{ is either } \square \text{ or } \blacksquare\}$, and

- $\sigma = \{x \mapsto (\pi) \ | \ x \in \text{Var}(t) \cap FV(\phi), t_\pi = x\}$.\(^3\)

We extend the domain of $\text{Rep}_{\blacksquare}$ to sets of constrained terms: $\text{Rep}_{\blacksquare}(T) = \bigcup_{t_{[\phi]} \in T} \text{Rep}_{\blacksquare}(t_{[\phi]})$. A pair of a ground term $t \in T(F \cup G \cup \{\square, \blacksquare\})$ and a formula $\phi \in \text{Fol}(G, P, \langle \mathbb{N}^+ \rangle)$ is called a

\(^3\) $\pi$ is unique since $t$ is linear.
constrained pattern if \( \pi \in \mathcal{P}os(t) \) and \( t|_{\pi} = \square \) for each variable \( \langle \pi \rangle \) that occurs in \( \phi \).

Note that \( \text{Rep}_{\pi}(t|_{\pi}) \) is a set of constrained patterns. Roughly speaking, a constrained pattern \( u|_{\phi} \) represents a set of terms obtained by replacing \( \square \) and \( \blacksquare \) in \( u \) by interpretable and uninterpretable terms, resp., such that the constraint obtained by the corresponding replacement holds.

**Example 4.** Let \( \mathcal{F} = \{ \text{g}/2 \} \), \( \mathcal{G} = \{ 0/0, s/1 \} \), and \( \mathcal{P} = \{ \leq, \geq \} \). Let symbols \( 0, s, \leq, \) and \( \geq \) be interpreted by \( \mathcal{S} \) as zero function and successor function, less-or-equal relation, and greater-or-equal relation, resp. For \( \mathcal{T} = \{ g(x, y)|_{x \leq 0}, g(s(x), y)|_{y \geq 0} \} \),

\[
\text{Rep}_{\pi}(\mathcal{T}) = \{ g(\square, \square)|_{(1) \leq 0}, g(\square, \blacksquare)|_{(1) \leq 0}, g(s(\square), \square)|_{(1.1) \geq 0}, g(s(\square), \blacksquare)|_{(1.1) \geq 0} \}
\]

Next, we define a function to augment their subterms to constrained patterns.

**Definition 5.** For a set \( U \) of constrained patterns, the set \( \text{Subp}(U) \) is defined as follows:

\[
\text{Subp}(U) = \{ u|_{\pi} | u|_{\phi} \in U, \pi \in \mathcal{P}os(u) \setminus \{ \varepsilon \} \}
\]

**Example 6.** For \( T \) in Example 4, \( \text{Subp}(\text{Rep}_{\pi}(T)) = \{ \square, \blacksquare, s(\square) \} \).

We define a quasi-order over constrained patterns that represents an approximation relation.

**Definition 7.** A quasi-order \( \sqsubseteq \) over terms in \( T(\mathcal{F} \cup \mathcal{G} \cup \{ \square, \blacksquare \}) \) is inductively defined as follows:

\[
\square \sqsubseteq u \text{ for } u \in T(\mathcal{G} \cup \{ \square \}); \blacksquare \sqsubseteq u \text{ for } u \in T(\mathcal{F} \cup \mathcal{G} \cup \{ \square, \blacksquare \}) \setminus T(\mathcal{G} \cup \{ \square \}); f(u_1, \ldots, u_n) \sqsubseteq f(u_1', \ldots, u_n') \text{ if } u_i \sqsubseteq u_i' \text{ for all } 1 \leq i \leq n.
\]

Abusing notations, we also define a quasi-order \( \sqsubseteq \) over formulas in \( \text{Fol}(\mathcal{G}, \mathcal{P}, X) \) as follows:

\[
\phi \sqsubseteq \phi' \text{ if } FV(\phi) \subseteq FV(\phi') \text{ and } \phi \Rightarrow \phi' \text{ is valid w.r.t. } \mathcal{S}.
\]

A quasi-order \( \sqsubseteq \) over constrained patterns is defined as follows: \( u|_{\phi} \sqsubseteq u'|_{\phi'} \text{ if } u \sqsubseteq u' \text{ and } \phi \sqsubseteq \phi' \). We use \( \simeq \) for equality part of \( \sqsubseteq \), and \( \sqsubset \) for strict part of \( \sqsubseteq \).

Next, to compute more concrete patterns, we define an operation \( \sqcap \) for constrained patterns.

**Definition 8.** We define a binary operator \( \sqcap \) over terms in \( T(\mathcal{F} \cup \mathcal{G} \cup \{ \square, \blacksquare \}) \) inductively as follows:

\[
\square \sqcap u = u \sqcap \square = u \text{ for } u \in T(\mathcal{G} \cup \{ \square \}); \blacksquare \sqcap u = u \sqcap \blacksquare = u \text{ for } u \in T(\mathcal{F} \cup \mathcal{G} \cup \{ \square, \blacksquare \}) \setminus T(\mathcal{G} \cup \{ \square \}); f(u_1, \ldots, u_n) \sqcap f(u_1', \ldots, u_n') = f(u_1 \sqcap u_1', \ldots, u_n \sqcap u_n').
\]

In the following, we define the set of constrained patterns used for labels of states.

**Definition 9.** For a set \( U \) of constrained patterns, \( \text{Inst}(U) \) is the smallest set satisfying the following:

- \( \{ \square|_{T}, \blacksquare|_{T} \} \cup \{ u|_{T} | u \in \text{Subp}(U) \} \subseteq \text{Inst}(U) \),
- \( \{ u|_{\phi'} | u|_{\phi} \in U, \phi' \leq \phi, \phi' \text{ is satisfiable w.r.t. } \mathcal{S} \} \subseteq \text{Inst}(U) \), and
- \( \{ u \sqcap u' | u|_{\phi} \sqcap u'|_{\phi'} \leq \phi \text{ is satisfiable w.r.t. } \mathcal{S} \} \subseteq \text{Inst}(U) \).

We use \( \simeq \) for equality terms that enjoy \( \simeq \), and \( \text{Inst}(U) \) is finite up to \( \simeq \).

**Example 10.** Consider \((\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{S})\) and \( T \) in Example 4. The set \( \text{Inst}(\text{Rep}_{\pi}(T)) \) contains the following in addition to \( \text{Rep}_{\pi}(T) \):

\[
\begin{align*}
\square|_{T}, \quad & g(s(\square), \square)|_{(1.1) \geq 0}, \quad g(s(\square), \blacksquare)|_{(1.1) \geq 0}, \\
g(s(\square), \blacksquare)|_{(1.1) \geq 0}, \quad & g(s(\square), \square)|_{(1.1) \geq 0}, \quad g(s(\square), \blacksquare)|_{(1.1) \geq 0}, \\
g(s(\square), \square)|_{(1.1) \geq 0}, \quad & g(s(\square), \blacksquare)|_{(1.1) \geq 0}, \quad g(s(\square), \square)|_{(1.1) \geq 0}, \quad g(s(\square), \blacksquare)|_{(1.1) \geq 0}.
\end{align*}
\]
Finally, we show a construction of a CTA recognizing $G(T)$. We prepare two kinds of states $q_u$ and $\bar{q}_u$ for each pattern to distinguish whether the term with the state satisfies the constraint in the corresponding constrained term. In the following, we use $q$ to denote $q$ or $\bar{q}$, and we denote the set of maximal constrained patterns in $U$ w.r.t. $u$ by $Max_u(U)$: $\text{Max}_u(U) = \{u'_[\varphi'] \in U \mid u' \subseteq u, (\exists u''[\alpha''] \in U. u'' \subseteq u \land u'_[\varphi'] \subseteq u''[\alpha''])\}$.

**Definition 11.** For a finite set $T$ of constrained terms, we prepare the set $\text{Lab}(T)$ of constrained patterns whose term parts are used as labels for states:

$$\text{Lab}(T) = \text{Inst}(\text{Rep}_T(T))$$

The subset $\text{Lab}_0(T)$ of constrained patterns that match elements of $T$ is defined as follows:

$$\text{Lab}_0(T) = \{u_{[\varphi]} \in \text{Lab}(T) \mid \exists u'[\varphi'] \in \text{Rep}_T(T). u'[\varphi'] \subseteq u_{[\varphi]}\}$$

Then, we define a CTA $A = (Q, Q_f, \Delta)$ as follows:

$$Q_f := \{\bar{q}_u \mid u_{[\varphi]} \in \text{Lab}_0(T), \exists u' \in \text{Subp}(\text{Rep}_T(T)) \setminus \{\Box \top, \Box \bot\}, u' \subseteq u\}$$

$$\cup \{\Box \top \mid \exists u_{[\varphi]} \in \text{Rep}_T(T). u \in T(G \cup \{\Box\}) \land (\exists u' \in \text{Subp}(\text{Rep}_T(T)) \setminus \{\Box \top, \Box \bot\}, u' \subseteq u\}$$

$$\cup \{\bar{q}_u \mid u_{[\varphi]} \in \text{Rep}_T(T). u \notin T(G \cup \{\Box\}) \land (\exists u' \in \text{Subp}(\text{Rep}_T(T)) \setminus \{\Box \top, \Box \bot\}, u' \subseteq u\}$$

$$Q := Q_f \cup \{q_u, q_{\Box\top}\} \cup \{q_u \mid u_{[\varphi]} \in \text{Lab}(T), \exists u' \in \text{Subp}(\text{Rep}_T(T)) \setminus \{\Box \top, \Box \bot\}, u' \subseteq u\}$$

$$\Delta := \{f(\bar{q}_{u_1}, \ldots, \bar{q}_{u_n}) \Rightarrow q_u \mid q_{u_1}, \ldots, q_{u_n} \in Q, q_u \in \text{Lab}_0(T) \land u_{[\varphi]} \in \text{Max}_{f(u_1, \ldots, u_n)}(\text{Lab}(T))\}$$

$$\cup \{f(\bar{q}_{u_1}, \ldots, \bar{q}_{u_n}) \Rightarrow \bar{q}_u \mid q_{u_1}, \ldots, q_{u_n} \in Q, \exists u_{[\varphi]} \in \text{Lab}_0(T), u_{[\varphi]} \notin Q_f\}$$

$$\cup \{f(q_{u_1}, \ldots, q_{u_n}) \Rightarrow \bar{q}_u \mid q_{u_1}, \ldots, q_{u_n} \in Q, \exists u_{[\varphi]} \in \text{Max}_{f(u_1, \ldots, u_n)}(\text{Lab}(T))\}$$

$$\cup \{f(q_{u_1}, \ldots, q_{u_n}) \Rightarrow q_u \mid q_{u_1}, \ldots, q_{u_n} \in Q, \exists u_{[\varphi]} \in \text{Lab}_0(T) \land u_{[\varphi]} \notin Q_f\}$$

Note that the constructed transition rules are not always optimized.

**Theorem 12.** The CTA $A$ constructed in Definition 11 is a deterministic, complete, and constraint-complete CTA such that $L(A) = G(T)$.

**Example 13.** Consider $G$, $P$ and $S$ in Example 4, $F = \{f(1)\}$, and $T = \{f(x)_{[x \leq 0]}, f(s(x))_{[x \geq 0]}\}$. Then, we have that

$$\text{Rep}_T(T) = \{\Box \top\}_{[1 \leq 0]}, f(s(\Box \top))_{[-1, 1]} \geq 0\}$$

$$\text{Subp}(\text{Rep}_T(T)) = \{\Box \bot, s(\Box \bot)\}$$

$$\text{Inst}(\text{Rep}_T(T)) = \{\Box \bot, s(\Box \bot)\}$$

$$\text{Lab}_0(T) = \{f(s(\Box \bot))_{[-1, 1]} \geq 0, f(s(\Box \top))_{[-1, 1]} \geq 0\}$$

The CTA $A = \{(q_0, q_{\Box\bot}, q_{\Box\top}, \bar{q}_{\top}, \bar{q}_u), \{\bar{q}_u\}, \Delta\}$ is constructed by Definition 11 with the following transition rules:

$$\Delta = \begin{cases} 0 \rightarrow q_0, & s(q_0) \rightarrow q_{\Box\top}, \ f(q_{\Box\bot}) \rightarrow \bar{q}_u, \\ s(q_{\Box\bot}) \rightarrow q_{\Box\bot}, \ f(q_{\Box\bot}) \rightarrow \bar{q}_u, \\ s(q_{\Box\top}) \rightarrow q_{\Box\top}, \ f(q_{\Box\top}) \rightarrow \bar{q}_u, \\ s(\bar{q}_u) \rightarrow \bar{q}_u, \ f(\bar{q}_u) \rightarrow \bar{q}_u, \\ s(\bar{q}_u) \rightarrow \bar{q}_u, \ f(\bar{q}_u) \rightarrow \bar{q}_u, \\ s(q_{\Box\top}) \rightarrow q_{\Box\top}, \ f(q_{\Box\top}) \rightarrow \bar{q}_u, \\ s(q_{\Box\bot}) \rightarrow q_{\Box\bot}, \ f(q_{\Box\bot}) \rightarrow \bar{q}_u, \\ s(\bar{q}_{\top}) \rightarrow \bar{q}_{\top}, \ f(\bar{q}_{\top}) \rightarrow \bar{q}_{\top}, \\ s(\bar{q}_{\bot}) \rightarrow \bar{q}_{\bot}, \ f(\bar{q}_{\bot}) \rightarrow \bar{q}_{\bot}, \\ s(q_{\Box\bot}) \rightarrow q_{\Box\bot}, \ f(q_{\Box\bot}) \rightarrow \bar{q}_u, \\ s(q_{\Box\top}) \rightarrow q_{\Box\top}, \ f(q_{\Box\top}) \rightarrow \bar{q}_u, \\ s(\bar{q}_{\top}) \rightarrow \bar{q}_{\top}, \ f(\bar{q}_{\top}) \rightarrow \bar{q}_{\top}, \\ s(\bar{q}_{\bot}) \rightarrow \bar{q}_{\bot}, \ f(\bar{q}_{\bot}) \rightarrow \bar{q}_{\bot}, \\ s(q_{\Box\bot}) \rightarrow q_{\Box\bot}, \ f(q_{\Box\bot}) \rightarrow \bar{q}_u, \\ s(q_{\Box\top}) \rightarrow q_{\Box\top}, \ f(q_{\Box\top}) \rightarrow \bar{q}_u, \\ s(\bar{q}_{\top}) \rightarrow \bar{q}_{\top}, \ f(\bar{q}_{\top}) \rightarrow \bar{q}_{\top}, \\ s(\bar{q}_{\bot}) \rightarrow \bar{q}_{\bot}, \ f(\bar{q}_{\bot}) \rightarrow \bar{q}_{\bot} \end{cases}$$
4 Conclusion

In this paper, we proposed a construction method of deterministic, complete, and constraint-complete CTAs recognizing ground instances of constrained terms. For the lack of space, we did not describe how to apply it to the verification of reduction-completeness and sufficient completeness, while we have already worked for some examples.

Unlike the case of tree automata, for a state, it is in general undecidable whether there exists a term reachable to the state, and thus, the intersection emptiness problem of CTAs is undecidable in general (see the case of AWEDC [3, Theorem 4.2.10]). For this reason, we will use a trivial sufficient condition that the set of final states of product automata is empty. Surprisingly, this is sometimes useful for product automata. To make this approach more powerful, we need to develop a method to find states that are not reachable from any ground term, e.g., there is a room for finding a transition rule that is never used: for $f(q_{\Box}(\langle 1 \rangle \leq 0 \land \langle 1.1 \rangle \geq 0) \rightarrow q_{\Box}$ in Example 13, we know that in applying this rule, the first argument of $f$ is always an interpretable term of the form $s(t)$, and thus, $\langle 1 \rangle$ in the constraint can be replaced by $s(\langle 1.1 \rangle)$; then we can notice that the constraint is unsatisfiable, and thus, this transition rule is never used. Formalizing this observation is one of our future work.

References