On Constructing Constrained Tree Automata Recognizing Ground Instances of Constrained Terms

Naoki Nishida
Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, 4648603 Nagoya, Japan
nishida@is.nagoya-u.ac.jp

Masahiko Sakai
Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, 4648603 Nagoya, Japan
sakai@is.nagoya-u.ac.jp

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

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.

S. Maneth (Ed.): Workshop on Trends in Tree
Automata and Tree Transducers (TTATT 2013).
EPTCS 134, 2013, pp. 1–10, doi:10.4204/EPTCS.134.1

2 Preliminaries

In this section, we briefly recall the basic notions of terms [3], constraints over predicate logic [6], and constrained tree automata [8].

Throughout the paper, we use $\mathscr{V}$ as a countably infinite set of *variables*. For a *signature* $\mathscr{F}$ (a finite set of function symbols with fixed arities), the set of *terms* over $\mathscr{F}$ and $X \subseteq \mathscr{V}$ is denoted by $\mathscr{T}(\mathscr{F},X)$. The notation f/n represents the function symbol f with the arity n. The set $\mathscr{T}(\mathscr{F},\emptyset)$ of *ground terms* is abbreviated to $\mathscr{T}(\mathscr{F})$. The set of variables appearing in term t is denoted by $\mathscr{V}ar(t)$. 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 $\mathscr{P}os(t)$:

- $\mathscr{P}os(x) = \{\varepsilon\}$ for $x \in \mathscr{V}$, and
- $\mathscr{P}os(f(t_1,\ldots,t_n)) = \{\varepsilon\} \cup \{i.\pi \mid 1 \leq i \leq n,\ \pi \in \mathscr{P}os(t_i)\}$ for $f/n \in \mathscr{F}$.

For terms t,u and a position π of t, the notation $t[u]_\pi$ denotes the term obtained from t by replacing the *subterm* $t|_\pi$ *of* t *at* π with u. For a *substitution* θ, we denote the *range* of θ by $\mathscr{R}an(\theta)$.

Let $\mathscr{G}$ be a signature and $\mathscr{P}$ 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* ϕ over $\mathscr{G}$, $\mathscr{P}$, and $\mathscr{V}$ are defined in BNF as follows:[1]

$$\phi ::= p(t_1,\ldots,t_n) \mid \top \mid \bot \mid (\neg\phi) \mid (\phi \vee \phi) \mid (\phi \wedge \phi)$$

where $p/n \in \mathscr{P}$ and $t_1,\ldots,t_n \in \mathscr{T}(\mathscr{G},\mathscr{V})$. We may omit brackets "(", ")" from formulas as usual. The set of first-order formulas over $\mathscr{G}$, $\mathscr{P}$, and $X \subseteq \mathscr{V}$ is denoted by $\mathscr{F}ol(\mathscr{G},\mathscr{P},X)$. For a formula ϕ, the notation $\mathscr{V}ar(\phi)$ represents the set of variables in ϕ. A variable-free formula is called *closed*. A *structure* for $\mathscr{G}$ and $\mathscr{P}$ is a tuple $\mathscr{S} = (A,\mathscr{I}_\mathscr{G},\mathscr{I}_\mathscr{P})$ such that the *universe* A is a non-empty set of concrete values, $\mathscr{I}_\mathscr{G}$ and $\mathscr{I}_\mathscr{P}$ with types $\mathscr{G} \to \{f \mid f \text{ is an } n\text{-ary function on } A\}$ and $\mathscr{P} \to 2^{A\times\cdots\times A}$ are interpretations for $\mathscr{G}$ and $\mathscr{P}$, resp.:

- $\mathscr{I}_\mathscr{G}(g)(a_1,\ldots,a_n) \in A$ for $g/n \in \mathscr{G}$, and
- $\mathscr{I}_\mathscr{P}(p) \subseteq A^n$ for $p/n \in \mathscr{P}$.

The interpretation of formulas ϕ w.r.t. $\mathscr{S}$, denoted by $\mathscr{S} \models \phi$, is defined as usual. We say that a formula ϕ *holds* w.r.t. $\mathscr{S}$ if $\mathscr{S} \models \phi$. Formulas in $\mathscr{F}ol(\mathscr{G},\mathscr{P},\mathscr{V})$ interpreted by $\mathscr{S}$ are called *constraints* (*w.r.t.* $\mathscr{S}$).

In the following, we use $\mathscr{F}$ and $\mathscr{G}$ for signatures, $\mathscr{P}$ for a set of predicate symbols, and $\mathscr{S} = (A,\mathscr{I}_\mathscr{G},\mathscr{I}_\mathscr{P})$ for a structure for $\mathscr{G}$ and $\mathscr{P}$, without notice. Before formalizing constrained tree automata, we generalize the interpretation of constraints under terms. For a sequence π of natural numbers, the notation $\langle\pi\rangle$ denotes the special variable related to π. We denote the set of such variables by $\langle\mathbb{N}^*\rangle$: $\langle\mathbb{N}^*\rangle = \{\langle\pi\rangle \mid \pi \in \mathbb{N}^*\} \subseteq \mathscr{V}$. A formula ϕ in $\mathscr{F}ol(\mathscr{G},\mathscr{P},\langle\mathbb{N}^*\rangle)$ *holds* w.r.t. a ground term $t \in \mathscr{T}(\mathscr{F}\cup\mathscr{G})$ if $\mathscr{S},t \models \phi$, where $\models$ is inductively defined as follows:

- $\mathscr{S},t \models \top$,
- $\mathscr{S},t \not\models \bot$,
- $\mathscr{S},t \models p(t_1,\ldots,t_n)$ if $\pi \in \mathscr{P}os(t)$ and $t|_\pi \in \mathscr{T}(\mathscr{G})$ for all variables $\langle\pi\rangle \in \bigcup_{i=1}^n \mathscr{V}ar(t_i)$, and $(\mathscr{I}_\mathscr{G}(t_1\theta),\ldots,\mathscr{I}_\mathscr{G}(t_n\theta)) \in \mathscr{I}_\mathscr{P}(p)$, where $p/n \in \mathscr{P}$ and θ is the substitution $\{\langle\pi\rangle \mapsto t|_\pi \mid \langle\pi\rangle \in \bigcup_{i=1}^n \mathscr{V}ar(t_i)\}$,

[1] It 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.

- otherwise, $\mathscr{S},t \not\models p(t_1,\ldots,t_n)$, and
- the relation $\models$ is defined as usual for any Boolean connective.

We may omit $\mathscr{S}$ from "$\mathscr{S},t \models \phi$" if it is clear in context. Note that $t \models \neg\phi$ does not coincide with $t \not\models \phi$ for every ground term t and constraint ϕ (see [8]).

By generalizing *AWEDCs* [3], *constrained tree automata* are defined as follows [8] — note that *AWEDCs* [3] are CTAs.

Definition 1 *A* constrained tree automata *(CTA) over $\mathscr{F}$, $\mathscr{G}$, $\mathscr{P}$, and $\mathscr{S}$ is a tuple $\mathscr{A} = (Q, Q_f, \Delta)$ such that*

- *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*
- *Δ 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 $f/n \in \mathscr{F} \cup \mathscr{G}$, $q_1,\ldots,q_n,q \in Q$ and $\phi \in \mathscr{F}ol(\mathscr{G},\mathscr{P},\langle\mathbb{N}^*\rangle)$.*

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 $\to_{\mathscr{A}}$ *is defined as follows: $t \to_{\mathscr{A}} u$ iff t has no nest of state symbols, $t|_\pi = f(q_1(t_1),\ldots,q_n(t_n))$, $f/n \in \mathscr{F} \cup \mathscr{G}$, $t_1,\ldots,t_n \in \mathscr{T}(\mathscr{F} \cup \mathscr{G})$, $f(q_1,\ldots,q_n) \xrightarrow{\phi} q \in \Delta$, $f(t_1,\ldots,t_n) \models \phi$, and $u = t[q(f(t_1,\ldots,t_n))]_\pi$.*

The terminologies of CTAs are defined analogously to those of *tree automata*, except for determinism and completeness — $\mathscr{A}$ is called

- *deterministic* if for every ground term t, there is at most one state $q \in Q$ such that $t \to^*_{\mathscr{A}} q$, and
- *complete* if for every ground term t, there is at least one state $q \in Q$ such that $t \to^*_{\mathscr{A}} q$.

Note that the above definition of determinism and completeness for CTAs is the same as the definition of the properties for AWEDCs (see [3]).

Example 2 Let $\mathscr{F} = \{\mathsf{f}/2\}$, $\mathscr{G}_{\rm int} = \{\mathsf{s}/1, \mathsf{p}/1, 0/0\}$, $\mathscr{P} = \{=, \neq, \leq, <\}$, and $\mathscr{S}_{\rm int}$ be the structure $(\mathbb{Z}, \mathscr{I}_{\mathscr{G}_{\rm int}}, \mathscr{I}_{\mathscr{P}_{\rm int}})$ for $\mathscr{G}_{\rm int}$ and $\mathscr{P}_{\rm int}$ such that $\mathscr{I}_{\mathscr{G}_{\rm int}}(\mathsf{s})(x) = x+1$, $\mathscr{I}_{\mathscr{G}_{\rm int}}(\mathsf{p})(x) = x-1$, $\mathscr{I}_{\mathscr{G}_{\rm int}}(0) = 0$, and $=, \neq, \leq, <$ are interpreted over integers as usual. Consider a CTA $\mathscr{A}_{\rm int} = (\{q_1,q_2\}, \{q_2\}, \Delta)$ over $\mathscr{F}$, $\mathscr{G}_{\rm int}$, $\mathscr{P}_{\rm int}$, and $\mathscr{S}_{\rm int}$ where

$$\Delta = \{\ 0 \to q_1,\quad \mathsf{s}(q_1) \to q_1,\quad \mathsf{p}(q_1) \to q_1,\quad \mathsf{f}(q_1,q_1) \xrightarrow{\langle 1.1\rangle \leq \mathsf{p}(\langle 2\rangle)} q_2,\quad \mathsf{f}(q_1,q_1) \to q_1\ \}$$

The term $\mathsf{f}(\mathsf{s}(0),\mathsf{s}(0))$ is accepted by $\mathscr{A}_{\rm int}$ — we have that $\mathsf{f}(q_1(\mathsf{s}(0)),q_1(\mathsf{s}(0))) \to_{\mathscr{A}_{\rm int}} q_2$ since both $\mathsf{f}(\mathsf{s}(0),\mathsf{s}(0)) \to^*_{\mathscr{A}_{\rm int}} \mathsf{f}(q_1(\mathsf{s}(0)),q_1(\mathsf{s}(0)))$ and $\mathsf{f}(\mathsf{s}(0),\mathsf{s}(0)) \models \langle 1.1\rangle \leq \mathsf{p}(\langle 2\rangle)$ hold. The term $\mathsf{f}(\mathsf{s}(0),\mathsf{s}(0))$ also transitions into q_1, and thus, $\mathscr{A}_{\rm int}$ is not deterministic. On the other hand, the term $\mathsf{f}(0,\mathsf{s}(0))$ is not accepted by $\mathscr{A}_{\rm int}$ since $\mathsf{f}(0,\mathsf{s}(0)) \not\models \langle 1.1\rangle \leq \mathsf{p}(\langle 2\rangle)$. Note that $\mathscr{A}_{\rm int}$ recognizes the set of instances of the *constrained terms* $\mathsf{f}(\mathsf{s}(x),y)_{[x\leq\mathsf{p}(y)]}$, $\mathsf{f}(\mathsf{p}(x),y)_{[x\leq\mathsf{p}(y)]}$, and $\mathsf{f}(\mathsf{f}(x,z),y)_{[x\leq\mathsf{p}(y)]}$, i.e., $L(\mathscr{A}_{\rm int}) = \{\mathsf{f}(\mathsf{s}(t_1),t_2),\ \mathsf{f}(\mathsf{p}(t_1),t_2),\ \mathsf{f}(\mathsf{f}(t_1,t),t_2) \mid t \in \mathscr{T}(\mathscr{F} \cup \mathscr{G}_{\rm int}),\ t_1,t_2 \in \mathscr{T}(\mathscr{G}_{\rm int}),\ \mathscr{I}_{\mathscr{G}_{\rm int}}(t_1) \leq \mathscr{I}_{\mathscr{G}_{\rm int}}(\mathsf{p}(t_2))\}$ (see Definition 3).

[2] We consider transition rules $l \xrightarrow{\phi} r$ and $l \xrightarrow{\psi} r$ to be equivalent if $\mathscr{V}ar(\phi) = \mathscr{V}ar(\psi)$ and ϕ is semantically equivalent to ψ (i.e., $\phi \leftrightarrow \psi$ is valid w.r.t. $\mathscr{S}$).

Consider the CTA $\mathscr{A}'_{\mathrm{int}} = (\{q_1,q_2\},\{q_2\},\Delta')$ obtained from $\mathscr{A}_{\mathrm{int}}$ by replacing $\langle 1.1\rangle \leq \mathsf{p}(\langle 2\rangle)$ with $\neg(\mathsf{p}(\langle 2\rangle) < \langle 1.1\rangle)$:

$$\Delta' = \{\ \mathsf{0} \to q_1,\quad \mathsf{s}(q_1) \to q_1,\quad \mathsf{p}(q_1) \to q_1,\quad \mathsf{f}(q_1,q_1) \xrightarrow{\neg(\mathsf{p}(\langle 2\rangle) < \langle 1.1\rangle)} q_2,\quad \mathsf{f}(q_1,q_1) \to q_1\ \}$$

The constraints $x \leq \mathsf{p}(y)$ and $\neg(\mathsf{p}(y) < x)$ are semantically equivalent, i.e., for all terms $t_1, t_2 \in \mathscr{T}(\mathscr{G}_{\mathrm{int}})$, $\mathscr{S}_{\mathrm{int}} \models t_1 \leq \mathsf{p}(t_2)$ iff $\mathscr{S}_{\mathrm{int}} \models \neg(\mathsf{p}(t_2) < t_1)$. However, this is not the case for similar constraints over fixed terms, e.g., $\mathsf{f}(\mathsf{0},\mathsf{s}(\mathsf{0})) \models \langle 1.1\rangle \leq \mathsf{p}(\langle 2\rangle)$ does not hold, but $\mathsf{f}(\mathsf{0},\mathsf{s}(\mathsf{0})) \models \neg(\mathsf{p}(\langle 2\rangle) < \langle 1.1\rangle)$ holds. Thus, $\mathsf{f}(\mathsf{0},\mathsf{s}(\mathsf{0}))$ is accepted by $\mathscr{A}'_{\mathrm{int}}$, and hence $L(\mathscr{A}_{\mathrm{int}}) \neq L(\mathscr{A}'_{\mathrm{int}})$.

A CTA $\mathscr{A} = (Q, Q_f, \Delta)$ is called *constraint-complete* [8] if for every ground term $t \in \mathscr{T}(\mathscr{F} \cup \mathscr{G})$ and all transition rules $f(q_1,\ldots,q_n) \xrightarrow{\phi} q \in \Delta$ with $t = f(t_1,\ldots,t_n) \to^*_{\mathscr{A}} f(q_1,\ldots,q_n)$, we have that $\pi \in \mathscr{P}os(t)$ and $t|_\pi \in \mathscr{T}(\mathscr{G})$ for all variables $\langle\pi\rangle$ in ϕ. Note that every CTA can be transformed into an equivalent, deterministic, complete, and constraint-complete CTA [8]. Note also that completeness and constraint-completeness are different notions.

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 3 *A* constrained term *is a pair (t,ϕ), written as $t_{[\phi]}$, of a linear term $t \in \mathscr{T}(\mathscr{F} \cup \mathscr{G}, \mathscr{V})$ and a formula $\phi \in \mathscr{F}ol(\mathscr{G},\mathscr{P},\mathscr{V}ar(t))$. The set of* ground instances *of a constrained term $t_{[\phi]}$, denoted by $G(t_{[\phi]})$, is defined as follows: $G(t_{[\phi]}) = \{t\theta \in \mathscr{T}(\mathscr{F}\cup\mathscr{G}) \mid \mathscr{R}an(\theta|_{FV(\phi)}) \subseteq \mathscr{T}(\mathscr{G}),\ \mathscr{S} \models \phi\theta\}$. 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 $\Box$ and $\blacksquare$ to denote arbitrary interpretable and un-interpretable terms resp. In the following, we denote $\mathscr{T}(\mathscr{F}\cup\mathscr{G}\cup\{\Box,\blacksquare\})$ (the set of patterns) by $\mathscr{T}_{\Box,\blacksquare}$, $\mathscr{T}(\mathscr{G}\cup\{\Box\})$ (the set of interpretable patterns) by $\mathscr{T}_\Box$, and $\mathscr{T}(\mathscr{F}\cup\mathscr{G}\cup\{\Box,\blacksquare\}) \setminus \mathscr{T}(\mathscr{G}\cup\{\Box\})$ (the set of un-interpretable patterns) by $\mathscr{T}_\blacksquare$.

Definition 4 *A pair of a ground term $u \in \mathscr{T}_{\Box,\blacksquare}$ and a formula $\phi \in \mathscr{F}ol(\mathscr{G},\mathscr{P},\langle\mathbb{N}^+\rangle)$ is called a* constrained pattern *if $\pi \in \mathscr{P}os(t)$ and $t|_\pi \in \mathscr{T}_\Box$ for each variable $\langle\pi\rangle$ that occurs in ϕ.*

For a constrained term $t_{[\phi]}$, $(t_{[\phi]})|_\blacksquare$ denotes the set of constrained terms $t\theta_{[\phi\sigma]}$ where

- $\theta = \{x \mapsto \Box \mid x \in \mathscr{V}ar(t) \cap FV(\phi)\} \cup \bigcup_{y\in\mathscr{V}ar(t)\setminus FV(\phi)} \{y \mapsto v \mid v \text{ is either } \Box \text{ or } \blacksquare\}$, *and*
- $\sigma = \{x \mapsto \langle\pi\rangle \mid x \in \mathscr{V}ar(t)\cap FV(\phi),\ t|_\pi = x\}$.[3]

We extend the domain of $(\cdot)|_\blacksquare$ to sets of constrained terms: $T|_\blacksquare = \bigcup_{t_{[\phi]}\in T} (t_{[\phi]})|_\blacksquare$.

[3] π is unique since t is linear.

Roughly speaking, a constrained pattern $u_{[\phi]}$ represents a set of terms obtained by replacing $\Box$ and $\blacksquare$ in u by interpretable and un-interpretable terms, resp., such that the constraint obtained by the corresponding replacement holds.

Example 5 Let $\mathscr{F} = \{\mathsf{g}/2\}$, $\mathscr{G} = \{\mathsf{0}/0, \mathsf{s}/1\}$, and $\mathscr{P} = \{\leq, \geq\}$. Let symbols $\mathsf{0}$, s, $\leq$, and $\geq$ be interpreted by $\mathscr{S}$ as zero function and successor function, less-or-equal relation, and greater-or-equal relation, resp. For $T = \{\mathsf{g}(x,y)_{[x\leq \mathsf{0}]}, \mathsf{g}(\mathsf{s}(x),y)_{[x\geq \mathsf{0}]}\}$,

$$T|_{\Box\blacksquare} = \{\ \mathsf{g}(\Box,\Box)_{[\langle 1\rangle \leq \mathsf{0}]},\ \mathsf{g}(\Box,\blacksquare)_{[\langle 1\rangle \leq \mathsf{0}]},\ \mathsf{g}(\mathsf{s}(\Box),\Box)_{[\langle 1.1\rangle \geq \mathsf{0}]},\ \mathsf{g}(\mathsf{s}(\Box),\blacksquare)_{[\langle 1.1\rangle \geq \mathsf{0}]}\ \}$$

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

Definition 6 *For a set U of constrained patterns, the set $Patterns_{\lhd}(U)$ of proper subterms for constrained patterns in U is defined as follows:*

$$Patterns_{\lhd}(U) = \{u|_\pi \mid u_{[\phi]} \in U,\ \pi \in \mathscr{P}os(u) \setminus \{\varepsilon\},\ u|_\pi \notin \{\Box, \blacksquare\}\}$$

Example 7 For T in Example 5, $Patterns_{\lhd}(T|_{\Box\blacksquare}) = \{\mathsf{s}(\Box)\}$.

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

Definition 8 *A quasi-order $\sqsubseteq$ over terms in $\mathscr{T}_{\Box,\blacksquare}$ is inductively defined as follows:*

- $\Box \sqsubseteq u$ *for* $u \in \mathscr{T}_{\Box}$,
- $\blacksquare \sqsubseteq u$ *for* $u \in \mathscr{T}_{\blacksquare}$, *and*
- $f(u_1,\ldots,u_n) \sqsubseteq f(u'_1,\ldots,u'_n)$ *if* $u_i \sqsubseteq u'_i$ *for all* $1 \leq i \leq n$.

Abusing notations, we also define a quasi-order $\sqsubseteq$ over formulas in $\mathscr{F}ol(\mathscr{G},\mathscr{P},X)$ as follows: $\phi \sqsubseteq \phi'$ iff $FV(\phi) \subseteq FV(\phi')$ and $\phi' \Rightarrow \phi$ is valid w.r.t. $\mathscr{S}$.

A quasi-order $\sqsubseteq$ over constrained patterns is defined as follows: $u_{[\phi]} \sqsubseteq {u'}_{[\phi']}$ iff $u \sqsubseteq u'$ and $\phi \sqsubseteq \phi'$. For constrained patterns $u_{[\phi]}$ and ${u'}_{[\phi']}$, we say that $u_{[\phi]}$ is more general *than ${u'}_{[\phi']}$ (${u'}_{[\phi']}$ is* less general *than $u_{[\phi]}$) if $u_{[\phi]} \sqsubseteq {u'}_{[\phi']}$.*

We use $\doteq$ 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 9 *We define a binary operator $\sqcap$ over $\mathscr{T}_{\Box,\blacksquare}$ inductively as follows:*

- $\Box \sqcap u = u \sqcap \Box = u$ *for* $u \in \mathscr{T}_{\Box}$,
- $\blacksquare \sqcap u = u \sqcap \blacksquare = u$ *for* $u \in \mathscr{T}_{\blacksquare}$, *and*
- $f(u_1,\ldots,u_n) \sqcap f(u'_1,\ldots,u'_n) = f(u_1 \sqcap u'_1,\ldots,u_n \sqcap u'_n)$.

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

Definition 10 *For a set U of constrained patterns, $LessGeneralized(U)$ is the smallest set satisfying the following:*

- $\{\Box_{[\top]}, \blacksquare_{[\top]}\} \cup \{u_{[\top]} \mid u \in Patterns_{\lhd}(U)\} \subseteq LessGeneralized(U)$,
- $\{u_{[\phi']} \mid u_{[\phi]} \in U,\ \phi' \in \{\phi, \neg\phi\},\ \phi'$ *is satisfiable w.r.t.* $\mathscr{S}\} \subseteq LessGeneralized(U)$, *and*

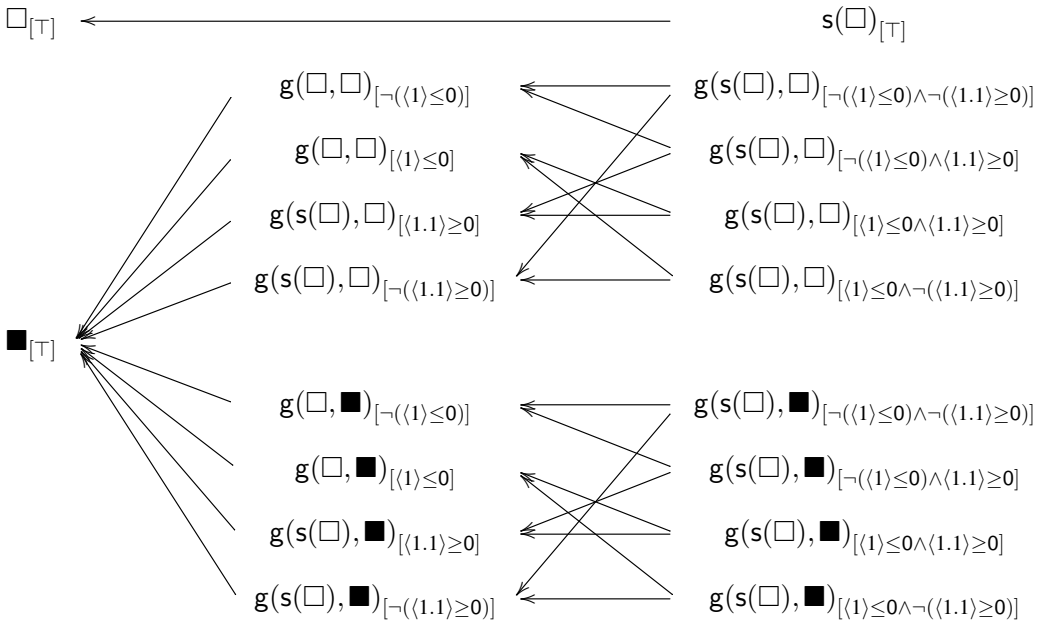
- $(u \sqcap u')_{[\phi \wedge \phi']}$ *if* $u_{[\phi]}$, $u'_{[\phi']} \in \mathit{LessGeneralized}(U)$ *and* $\phi \wedge \phi'$ *is satisfiable w.r.t.* $\mathscr{S}$.

Note that we do not distinguish terms that enjoy $\doteq$*, and that* $\mathit{LessGeneralized}(U)$ *is finite up to* $\doteq$.

Example 11 Consider $(\mathscr{F}, \mathscr{G}, \mathscr{P}, \mathscr{S})$ and T in Example 5. The set $\mathit{LessGeneralized}(T|_{\square\blacksquare})$ contains the following in addition to $T|_{\square\blacksquare}$:

$$\left\{\begin{array}{lll}
\square_{[\top]}, & \blacksquare_{[\top]}, & \mathsf{s}(\square)_{[\top]}, \\
\mathsf{g}(\square,\square)_{[\neg(\langle 1\rangle \le 0)]}, & \mathsf{g}(\square,\blacksquare)_{[\neg(\langle 1\rangle \le 0)]}, & \\
\mathsf{g}(\mathsf{s}(\square),\square)_{[\neg(\langle 1.1\rangle \ge 0)]}, & \mathsf{g}(\mathsf{s}(\square),\blacksquare)_{[\neg(\langle 1.1\rangle \ge 0)]}, & \\
\mathsf{g}(\mathsf{s}(\square),\square)_{[\langle 1\rangle \le 0 \wedge \langle 1.1\rangle \ge 0]}, & \mathsf{g}(\mathsf{s}(\square),\square)_{[\langle 1\rangle \le 0 \wedge \neg(\langle 1.1\rangle \ge 0)]}, & \\
\mathsf{g}(\mathsf{s}(\square),\square)_{[\neg(\langle 1\rangle \le 0) \wedge \langle 1.1\rangle \ge 0]}, & \mathsf{g}(\mathsf{s}(\square),\square)_{[\neg(\langle 1\rangle \le 0) \wedge \neg(\langle 1.1\rangle \ge 0)]}, & \\
\mathsf{g}(\mathsf{s}(\square),\blacksquare)_{[\langle 1\rangle \le 0 \wedge \langle 1.1\rangle \ge 0]}, & \mathsf{g}(\mathsf{s}(\square),\blacksquare)_{[\langle 1\rangle \le 0 \wedge \neg(\langle 1.1\rangle \ge 0)]}, & \\
\mathsf{g}(\mathsf{s}(\square),\blacksquare)_{[\neg(\langle 1\rangle \le 0) \wedge \langle 1.1\rangle \ge 0]}, & \mathsf{g}(\mathsf{s}(\square),\blacksquare)_{[\neg(\langle 1\rangle \le 0) \wedge \neg(\langle 1.1\rangle \ge 0)]} &
\end{array}\right\}$$

The relation between the constrained patterns in $\mathit{LessGeneralized}(T|_{\square\blacksquare})$ w.r.t. $\sqsubseteq$ is illustrated as follows:



where $\longleftarrow$ denotes $\sqsubset$.

Finally, we show a construction of a CTA recognizing $G(T)$. We prepare two kinds of states q_u and $\tilde{q}_u$ for each pattern to distinguish whether the term with the state satisfies the constraint in the corresponding constrained term. In the following, we use $\dot{q}$ to denote q or $\tilde{q}$, and, given an unconstrained pattern u and a set U of constrained patterns, we denote the set of *least general* constrained patterns in U which are more general than u by $\mathit{Least}_{\sqsubseteq u}(U)$:

$$\mathit{Least}_{\sqsubseteq u}(U) = \{u'_{[\phi']} \in U \mid u' \sqsubseteq u,\ (\nexists u''_{[\phi'']} \in U.\ u'' \sqsubseteq u \wedge u'_{[\phi']} \sqsubset u''_{[\phi'']})\}$$

Now we give an intuitive outline of the construction.

Final states For a pattern u capturing an instance of $t_{[\phi]}$ in T, we add the state $\tilde{q}_u$ to Q_f if u also captures an instance of a non-variable proper subterm of $t'_{[\phi']}$ in T; otherwise, we add $\tilde{q}_\square$ and $\tilde{q}_\blacksquare$ where u is in $\mathscr{T}_\square$ and $\mathscr{T}_\blacksquare$, resp.

States In addition to Q_f, for a pattern u capturing an instance of a non-variable proper subterm of $t'_{[\phi']}$ in T, we add q_u to Q; for arbitrary terms in $\mathscr{T}_\Box$ and $\mathscr{T}_\blacksquare$, we add $q_\Box$ and $q_\blacksquare$ to Q, resp.

Transition rules $f(\dot{q}_{u_1},\ldots,\dot{q}_{u_n})$ transitions to a state by considering the following properties of the least general pattern u which is more general than $f(u_1,\ldots,u_n)$:

(a) whether the pattern u also matches a non-variable *proper* subterm of some initial constrained term $t_{[\phi]} \in T$,

(b) whether the pattern u also matches an instance of $t'_{[\phi']}$ in T, and

(c) whether u is in $\mathscr{T}_\Box$.

The state $f(\dot{q}_{u_1},\ldots,\dot{q}_{u_n})$ transitions to is decided as follows:

state	(a)	(b)	(c)	the transition rule is in
q_u	yes	no	—	Δ_{pat}
$\tilde{q}_u$	yes	yes	—	$\tilde{\Delta}_{\mathsf{pat}}$
$q_\Box$	no	no	yes	$\Delta_\Box$
$\tilde{q}_\Box$	no	yes	yes	$\tilde{\Delta}_\Box$
$q_\blacksquare$	no	no	no	$\Delta_\blacksquare$
$\tilde{q}_\blacksquare$	no	yes	no	$\tilde{\Delta}_\blacksquare$

This approach to the construction is formalized as follows.

Definition 12 *For a finite set T of constrained terms, we prepare the set $Labels(T)$ of constrained patterns whose term parts are used as labels for states:*

$$Labels(T) = LessGeneralized(T|_{\Box\blacksquare})$$

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

$$\widetilde{Labels}(T) = \{u_{[\phi]} \in Labels(T) \mid \exists u'_{[\phi']} \in T|_{\Box\blacksquare}.\ u'_{[\phi']} \sqsubseteq u_{[\phi]}\}$$

We prepare the set Q_0 of candidates for states as follows:

$$Q_0 = \{q_u \mid \exists\phi.\ u_{[\phi]} \in Labels(T) \wedge (\exists u' \in Patterns_{\lhd}(T|_{\Box\blacksquare}).\ u' \sqsubseteq u)\}$$

Then, we define a CTA $\mathscr{A} = (Q, Q_f, \Delta)$ such that

$$\begin{aligned}
Q_f &= \{\tilde{q}_u \mid q_u \in Q_0,\ \exists\phi.\ u_{[\phi]} \in \widetilde{Labels}(T)\} \\
&\quad \cup \{\tilde{q}_\Box \mid \exists u_{[\phi]} \in T|_{\Box\blacksquare}.\ u \in \mathscr{T}_\Box \wedge q_u \notin Q_0\} \cup \{\tilde{q}_\blacksquare \mid \exists u_{[\phi]} \in T|_{\Box\blacksquare}.\ u \in \mathscr{T}_\blacksquare \wedge q_u \notin Q_0\} \\
Q &= Q_f \cup \{q_u \in Q_0 \mid \tilde{q}_u \notin Q_f\} \cup \{q_\Box, q_\blacksquare\} \\
\Delta &= \Delta_{\mathsf{pat}} \cup \tilde{\Delta}_{\mathsf{pat}} \cup \Delta_\Box \cup \tilde{\Delta}_\Box \cup \Delta_\blacksquare \cup \tilde{\Delta}_\blacksquare
\end{aligned}$$

where

$$\Delta_{\mathsf{pat}} := \qquad \overbrace{\quad}^{\text{(a)}} \qquad \overbrace{\qquad\qquad}^{\text{(b)}} \qquad \overbrace{\quad}^{\text{(c)}}$$

$$\{f(\dot{q}_{u_1},\ldots,\dot{q}_{u_n}) \xrightarrow{\phi} q_u \mid\ f \in \mathscr{F}\cup\mathscr{G},\ \dot{q}_{u_1},\ldots,\dot{q}_{u_n} \in Q,\ q_u \in Q,$$
$$u_{[\phi]} \in \mathit{Least}_{\sqsubseteq f(u_1,\ldots,u_n)}(\mathit{Labels}(T)),\qquad u_{[\phi]} \notin \widetilde{\mathit{Labels}}(T)\ \}$$

$$\tilde{\Delta}_{\mathsf{pat}} =$$

$$\{f(\dot{q}_{u_1},\ldots,\dot{q}_{u_n}) \xrightarrow{\phi} \tilde{q}_u \mid\ f \in \mathscr{F}\cup\mathscr{G},\ \dot{q}_{u_1},\ldots,\dot{q}_{u_n} \in Q,\ \tilde{q}_u \in Q,$$
$$u_{[\phi]} \in \mathit{Least}_{\sqsubseteq f(u_1,\ldots,u_n)}(\mathit{Labels}(T)),\qquad u_{[\phi]} \in \widetilde{\mathit{Labels}}(T)\ \}$$

$$\Delta_{\square} =$$

$$\{f(\dot{q}_{u_1},\ldots,\dot{q}_{u_n}) \xrightarrow{\phi} q_{\square} \mid f \in \mathscr{F}\cup\mathscr{G},\ \dot{q}_{u_1},\ldots,\dot{q}_{u_n} \in Q,\ q_{\square} \in Q,$$
$$(\exists\, u_{[\phi]} \in \mathit{Least}_{\sqsubseteq f(u_1,\ldots,u_n)}(\mathit{Labels}(T)).\ q_u \notin Q_0 \wedge u_{[\phi]} \notin \widetilde{\mathit{Labels}}(T) \wedge u \in \mathscr{T}_{\square})\}$$

$$\tilde{\Delta}_{\square} =$$

$$\{f(\dot{q}_{u_1},\ldots,\dot{q}_{u_n}) \xrightarrow{\phi} \tilde{q}_{\square} \mid f \in \mathscr{F}\cup\mathscr{G},\ \dot{q}_{u_1},\ldots,\dot{q}_{u_n} \in Q,\ \tilde{q}_{\square} \in Q,$$
$$(\exists\, u_{[\phi]} \in \mathit{Least}_{\sqsubseteq f(u_1,\ldots,u_n)}(\mathit{Labels}(T)).\ q_u \notin Q_0 \wedge u_{[\phi]} \in \widetilde{\mathit{Labels}}(T) \wedge u \in \mathscr{T}_{\square})\}$$

$$\Delta_{\blacksquare} =$$

$$\{f(\dot{q}_{u_1},\ldots,\dot{q}_{u_n}) \xrightarrow{\phi} q_{\blacksquare} \mid f \in \mathscr{F}\cup\mathscr{G},\ \dot{q}_{u_1},\ldots,\dot{q}_{u_n} \in Q,\ q_{\blacksquare} \in Q,$$
$$(\exists\, u_{[\phi]} \in \mathit{Least}_{\sqsubseteq f(u_1,\ldots,u_n)}(\mathit{Labels}(T)).\ q_u \notin Q_0 \wedge u_{[\phi]} \notin \widetilde{\mathit{Labels}}(T) \wedge u \in \mathscr{T}_{\blacksquare})\}$$

$$\tilde{\Delta}_{\blacksquare} =$$

$$\{f(\dot{q}_{u_1},\ldots,\dot{q}_{u_n}) \xrightarrow{\phi} \tilde{q}_{\blacksquare} \mid f \in \mathscr{F}\cup\mathscr{G},\ \dot{q}_{u_1},\ldots,\dot{q}_{u_n} \in Q,\ \tilde{q}_{\blacksquare} \in Q,$$
$$(\exists\, u_{[\phi]} \in \mathit{Least}_{\sqsubseteq f(u_1,\ldots,u_n)}(\mathit{Labels}(T)).\ q_u \notin Q_0 \wedge u_{[\phi]} \in \widetilde{\mathit{Labels}}(T) \wedge u \in \mathscr{T}_{\blacksquare})\}$$

Note that for any pattern $u \notin \{\square,\blacksquare\}$, $q_u \in Q_0$ iff $q_u \in Q$ or $\tilde{q}_u \in Q_f$, and thus, $q_u \in Q_0$ which makes (a) true implicitly holds in the conditions of Δ_{pat} and $\tilde{\Delta}_{\mathsf{pat}}$. Note also that the constructed transition rules are not always optimized via the construction in Definition 12.[4]

Theorem 13 *The CTA $\mathscr{A}$ constructed in Definition 12 is a deterministic, complete, and constraint-complete CTA such that $L(\mathscr{A}) = G(T)$.*

Proof. Due to the construction of constrained patterns used for states, for $u_{[\phi]} \in \mathit{Labels}(T)$, all of the following hold:

- $\pi \in \mathscr{P}os(u)$ and $u|_\pi \in \mathscr{T}_{\square}$ for any variable $\langle\pi\rangle \in FV(\phi)$,
- $\bigvee_{u_{[\psi]}\in\mathit{Labels}(T)} \psi$ is valid w.r.t. $\mathscr{S}$, and
- if $u_{[\phi]}$ is a least general in $\mathit{Labels}(T)$, then $\phi \wedge \psi$ is not satisfiable w.r.t. $\mathscr{S}$ for any least general constrained pattern $u_{[\psi]}$ in $\mathit{Labels}(T)$ such that $\phi \neq \psi$.

It follows from both the first property above and the construction of transition rules that $\mathscr{A}$ is constraint-complete. It follows from the second and third properties above that $\mathscr{A}$ is complete and deterministic. From these properties, every given term transitions to a unique state that keeps the structure of the given term as much as possible in the sense of the patterns to be considered. In constructing transition rules, we add constraints to transition rules so as to transition to a final state if a given initial ground term is an instance of a constrained term in T. For these reasons, it is clear that $L(\mathscr{A}) = G(T)$. □

[4] As in the case of unconstrained tree automata [3], a state q is not accessible if there is no transition rule of the form $l \xrightarrow{\phi} q$. Thus, such a state q and all the transition rules having q can be removed from $\mathscr{A}$.

Example 14 Consider $\mathscr{G}$, $\mathscr{P}$ and $\mathscr{S}$ in Example 5, $\mathscr{F} = \{\mathsf{f}/1\}$, and $T = \{\mathsf{f}(x)_{[x\leq 0]}, \mathsf{f}(\mathsf{s}(x))_{[x\geq 0]}\}$. Then, we have that

$$T|_{\square\!\!\blacksquare} = \{\mathsf{f}(\square)_{[\langle 1\rangle\leq 0]}, \mathsf{f}(\mathsf{s}(\square))_{[\langle 1.1\rangle\geq 0]}\}$$
$$\mathit{Patterns}_{\lhd}(T|_{\square\!\!\blacksquare}) = \{\mathsf{s}(\square)\}$$
$$\mathit{LessGeneralized}(T|_{\square\!\!\blacksquare}) = T|_{\square\!\!\blacksquare} \cup \{\square_{[\top]}, \mathsf{s}(\square)_{[\top]}\} \cup \left\{ \begin{array}{l} \blacksquare_{[\top]}, \mathsf{f}(\square)_{[\neg(\langle 1\rangle\leq 0)]}, \mathsf{f}(\mathsf{s}(\square))_{[\neg(\langle 1.1\rangle\geq 0)]}, \mathsf{f}(\mathsf{s}(\square))_{[\langle 1\rangle\leq 0\wedge\langle 1.1\rangle\geq 0]}, \\ \mathsf{f}(\mathsf{s}(\square))_{[\neg(\langle 1\rangle\leq 0)\wedge\langle 1.1\rangle\geq 0]}, \mathsf{f}(\mathsf{s}(\square))_{[\langle 1\rangle\leq 0\wedge\neg(\langle 1.1\rangle\geq 0)]}, \mathsf{f}(\mathsf{s}(\square))_{[\neg(\langle 1\rangle\leq 0)\wedge\neg(\langle 1.1\rangle\geq 0)]} \end{array} \right\}$$
$$\widetilde{\mathit{Labels}}(T) = T|_{\square\!\!\blacksquare} \cup \{\mathsf{f}(\mathsf{s}(\square))_{[\langle 1\rangle\leq 0\wedge\langle 1.1\rangle\geq 0]}, \mathsf{f}(\mathsf{s}(\square))_{[\langle 1\rangle\leq 0\wedge\neg(\langle 1.1\rangle\geq 0)]}, \mathsf{f}(\mathsf{s}(\square))_{[\neg(\langle 1\rangle\leq 0)\wedge\langle 1.1\rangle\geq 0]}\}$$

The CTA $\mathscr{A} = (\{q_\square, q_{\blacksquare}, q_{\mathsf{s}(\square)}, \tilde{q}_{\blacksquare}\}, \{\tilde{q}_{\blacksquare}\}, \Delta)$ is constructed by Definition 12 with the following transition rules:

$$\Delta = \left\{ \begin{array}{llll} 0 \to q_\square, & \mathsf{s}(q_\square) \to q_{\mathsf{s}(\square)}, & \mathsf{f}(q_\square) \xrightarrow{\langle 1\rangle\leq 0} \tilde{q}_{\blacksquare}, & \mathsf{f}(q_{\mathsf{s}(\square)}) \xrightarrow{\langle 1\rangle\leq 0\wedge\langle 1.1\rangle\geq 0} \tilde{q}_{\blacksquare}, \\ & \mathsf{s}(q_{\blacksquare}) \to q_{\blacksquare}, & \mathsf{f}(q_\square) \xrightarrow{\neg(\langle 1\rangle\leq 0)} q_{\blacksquare}, & \mathsf{f}(q_{\mathsf{s}(\square)}) \xrightarrow{\langle 1\rangle\leq 0\wedge\neg(\langle 1.1\rangle\geq 0)} \tilde{q}_{\blacksquare}, \\ & \mathsf{s}(q_{\mathsf{s}(\square)}) \to q_{\mathsf{s}(\square)}, & \mathsf{f}(q_{\blacksquare}) \to q_{\blacksquare}, & \mathsf{f}(q_{\mathsf{s}(\square)}) \xrightarrow{\neg(\langle 1\rangle\leq 0)\wedge\langle 1.1\rangle\geq 0} \tilde{q}_{\blacksquare}, \\ & \mathsf{s}(\tilde{q}_{\blacksquare}) \to q_{\blacksquare}, & \mathsf{f}(\tilde{q}_{\blacksquare}) \to q_{\blacksquare}, & \mathsf{f}(q_{\mathsf{s}(\square)}) \xrightarrow{\neg(\langle 1\rangle\leq 0)\wedge\neg(\langle 1.1\rangle\geq 0)} q_{\blacksquare} \end{array} \right\}$$

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 possibility to detect a transition rule that is never used: for $\mathsf{f}(q_{\mathsf{s}(\square)}) \xrightarrow{\neg(\langle 1\rangle\leq 0)\wedge\neg(\langle 1.1\rangle\geq 0)} q_{\blacksquare}$ in Example 14, we know that in applying this rule, the first argument of f is always an interpretable term of the form $\mathsf{s}(t)$, and thus, $\langle 1\rangle$ in the constraint can be replaced by $\mathsf{s}(\langle 1.1\rangle)$; 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

[1] Adel Bouhoula & Florent Jacquemard (2008): *Automated Induction with Constrained Tree Automata*. In Alessandro Armando, Peter Baumgartner & Gilles Dowek, editors: *Proceedings of the 4th International Joint Conference on Automated Reasoning, Lecture Notes in Computer Science* 5195, Springer, pp. 539–554, doi:10.1007/978-3-540-71070-7_44.

[2] Adel Bouhoula & Florent Jacquemard (2012): *Sufficient completeness verification for conditional and constrained TRS*. *Journal of Applied Logic* 10(1), pp. 127–143, doi:10.1016/j.jal.2011.09.001.

[3] Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison & Marc Tommasi (2007): *Tree Automata Techniques and Applications*. Available on: `http://www.grappa.univ-lille3.fr/tata`. Release October, 12th 2007.

[4] Laurent Fribourg (1989): *A Strong Restriction of the Inductive Completion Procedure*. Journal of Symbolic Computation 8(3), pp. 253–276, doi:10.1016/S0747-7171(89)80069-0.

[5] Yuki Furuichi, Naoki Nishida, Masahiko Sakai, Keiichirou Kusakari & Toshiki Sakabe (2008): *Approach to Procedural-program Verification Based on Implicit Induction of Constrained Term Rewriting Systems*. IPSJ Transactions on Programming 1(2), pp. 100–121. In Japanese.

[6] Michael Huth & Mark Ryan (2000): *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press.

[7] Naoki Nakabayashi, Naoki Nishida, Keiichirou Kusakari, Toshiki Sakabe & Masahiko Sakai (2011): *Lemma Generation Method in Rewriting Induction for Constrained Term Rewriting Systems*. Computer Software 28(1), pp. 173–189, doi:10.11309/jssst.28.1_173. In Japanese.

[8] Naoki Nishida, Futoshi Nomura, Katsuhisa Kurahashi & Masahiko Sakai (2012): *Constrained Tree Automata and their Closure Properties*. In Keisuke Nakano & Hiroyuki Seki, editors: Proceedings of the 1st International Workshop on Trends in Tree Automata and Tree Transducers, pp. 24–34.

[9] Tsubasa Sakata, Naoki Nishida & Toshiki Sakabe (2011): *On Proving Termination of Constrained Term Rewriting Systems by Eliminating Edges from Dependency Graphs*. In Herbert Kuchen, editor: Proceedings of the 20th International Workshop on Functional and (Constraint) Logic Programming (WFLP 2011), Lecture Notes in Computer Science 6816, Springer, pp. 138–155, doi:10.1007/978-3-642-22531-4_9.

[10] Tsubasa Sakata, Naoki Nishida, Toshiki Sakabe, Masahiko Sakai & Keiichirou Kusakari (2009): *Rewriting Induction for Constrained Term Rewriting Systems*. IPSJ Transactions on Programming 2(2), pp. 80–96. In Japanese.