ICOT Technical Memorandum: TM-0087

TM-0087

Knuth-Bendix Algorithm for Thue System Based on Kachinuki Ordering

Ko Sakai

December, 1984

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato ku Tokyo 108 Japan

(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

Knuth-Bendix Algorithm for Thue System Based on Kachinuki Ordering

Kō Sakai

ICOT Research Center
Institute for New Generation Computer Technology
Tokyo, Japan

ABSTRACT

The "kachinuki" odering method, which well-orders strings, is defined. An application of the Knuth-Bendix algorithm based on "kachinuki" ordering is presented for Thue systems. Finite complete rewriting systems for the Jantzen monoid and the Greendlinger group are mechanically generated by the algorithm.

1. Introduction

The word problem of an equational theory involves determining whether the equality of two arbitrary terms can be deduced from a given set of equations. Even for Thue systems, which are regarded as specific equational theories, such problems are not, in general, decidable. However, there are many concrete Thue systems whose word problem is known to be decidable.

A standard approach to solve the word problem of a given equational theory is to construct a complete rewriting system and use it as a decision procedure. The Knuth-Bendix algorithm [5] is well known as a mechanical method to obtain finite complete term-rewriting systems.

Otto presented two finite complete (string-)rewriting systems [6]: one for the Jantzen monoid (more precisely, a monoid isomorphic with the Jantzen monoid) and the other for the Greendlinger group. However, since there is no Knuth-Bendix ordering that orients the rewriting rules properly, the Knuth-Bendix algorithm based on such an ordering cannot generate Otto's rewriting systems.

In this paper it will be shown that the Knuth-Bendix algorithm based on a "kachinuki" ordering (defined below) can generate Otto's rewriting systems mechanically. Consequently, Theorem 3 and Theorem 9 in [6], which show the systems to be complete, are obtained as corollaries of Theorem 3.5 in this paper.

2. Rewriting system

Definition 2.1

Let Σ be an alphabet, i.e., a set of letters. A *(string-)rewriting system* R over Σ is a subset of $\Sigma^* \times \Sigma^*$. Let $\Rightarrow$ be the relation on Σ^* defined as follows:

$u \Rightarrow v$ if and only if there exist x and y in Σ^* and (l, r) in R such that $u = xly$ and $v = xry$

The *derivation* $\overset{*}{\Rightarrow}$ is the reflexive and transitive closure of $\Rightarrow$ and the *Thue congruence* $\overset{*}{\Leftrightarrow}$ is the reflexive, transitive, and commutative closure.

A pair (l, r) in a rewriting system is called a rewriting rule and is denoted by $l \rightarrow r$.

In the literature, a rewriting system is often called a Thue system, especially when the issue is Thue congruence (two-way rewriting) rather than derivation (one-way rewriting). In what follows, a Thue system formally means a rewriting system. Using this term indicates, following tradition, that we are principally concerned with Thue congruence. The above notation $r \rightarrow l$ will not be used for what is referred to as a Thue system.

A string-rewriting system (or a Thue system) can also be viewed as a term-rewriting system (or an equational theory) where letters are considered to be unary function symbols.

The word problem of a Thue system R involves the determination of whether $t_1 \overset{*}{\Leftrightarrow} t_2$ for two arbitrary strings t_1 and t_2.

Definition 2.2

Let R be a rewriting system. R is said to be *confluent* if, for any two derivations $t \overset{*}{\Rightarrow} t_1$ and $t \overset{*}{\Rightarrow} t_2$, there exists a string u such that $t_1 \overset{*}{\Rightarrow} u$ and $t_2 \overset{*}{\Rightarrow} u$.

Definition 2.3

A rewriting system is said to *terminate* if there exists no infinite derivation $t_1 \Rightarrow t_2 \Rightarrow \cdots$.

Definition 2.4

A string t is said to be *irreducible* if there exists no string u such that $t \Rightarrow u$.

Definition 2.5

A terminating and confluent rewriting system is called *complete*.

Let R be a terminating rewriting system. For every string t, there exists an irreducible string u such that $t \overset{*}{\Rightarrow} u$. Moreover, R is confluent if and only if the irreducible string u is unique. In this case, the string u is called the *normal* form of t and $t_1 \overset{*}{\Leftrightarrow} t_2$ if and only if t_1 and t_2 have the same normal. Therefore the word problem is decidable for a finite complete rewriting system.

Knuth and Bendix devised a mechanical method to generate finite complete term-rewriting systems for finite equational theories [5]. If the Knuth-Bendix algorithm is applied to a Thue system E viewed as an equational theory, it generates a finite complete string-rewriting system R with the same Thue congruence (and therefore the same word problem) as E.

Definition 2.6

Let $\prec$ be a partial ordering on Σ^*. $\prec$ is said to have the *replacement property* if $l \prec r$ implies $xly \prec xry$ for any l, r, x, and y.

Definition 2.7

Let $l_1 \rightarrow r_1$ and $l_2 \rightarrow r_2$ be rewriting rules. A *critical pair* is defined, if l_1 and l_2 overlap:

(1) If l_j is a substring of l_i $(i \neq j)$ i.e. $l_i = u l_j v$ for some u and v, then $(r_i, u r_j v)$ is a critical pair.

(2) If a postfix of l_i is a prefix of l_j, $(i \neq j)$ i.e., there exists $u \neq \epsilon$ (ϵ denotes the empty string) such that $l_i = vu$ and $l_j = uw$ for some v and w, then $(r_i w, v r_j)$ is a critical pair.

Now we will define below the Knuth-Bendix algorithm modified for Thue systems. Assume that a well-founded ordering $\prec$ with the replacement property is defined on Σ^*.

Knuth-Bendix Algorithm

Step 0: Set E to the initally given finite Thue system. Set R to empty. Go to **Step 1**.

Step 1: If E is empty, the current value of R is the desired rewriting system. Otherwise, go to **Step 2**.

Step 2: Remove a pair (t, u) from E, and find irreducible strings t_1 and u_1 such that $t \xrightarrow{*} t_1$, $u \xrightarrow{*} u_1$ with respect to R. If $t_1 = u_1$, go to Step 1. If $t_1 \succ u_1$ or $t_1 \prec u_1$, go to **Step 3**. Otherwise, stop; the procedure is unsuccessful.

Step 3: We can assume $t_1 \prec u_1$ without loss of generality. Remove all the rewriting rules $l \rightarrow r$ from R such that either l or r is reducible by the rewriting rule $t_1 \rightarrow u_1$, and append (l, r) to E instead. Append the new rule $t_1 \rightarrow u_1$ to R. Construct all the critical pairs generated between $t_1 \rightarrow u_1$ and each rule in R and append them to E. Go to **Step 1**.

3. Ordering of strings

Knuth and Bendix [5] presented a well-founded ordering method for terms with the property corresponding to the replacement property in the case of string. (For the string version of the Knuth-Bendix ordering, see [6].) We will define another well-founded ordering method for strings.

Definition 3.1

Let $\prec$ be an arbitrary partial ordering on Σ^*, and σ be a new letter which is not contained in Σ. We will define a partial ordering $\prec_\sigma$ on $(\Sigma \bigcup \{\sigma\})^*$. Let $|t|$ denote the number of occurences of σ in t. If $|t| < |u|$, then $t \prec_\sigma u$. If $|t| = |u| = n$, let

$$t = t_0 \sigma t_1 \sigma \ldots \sigma t_n, \quad u = u_0 \sigma u_1 \sigma \ldots \sigma u_n,$$

where each t_i and u_j are in Σ^*. In this case, $t \prec_\sigma u$ if there exist an i $(0 \leq i \leq n)$ such that

$$t_n = u_n, \ \ldots, t_{i+1} = u_{i+1}, \ t_i \prec u_i.$$

It is easy to verify the following lemma.

Lemma 3.2

(1) $\prec_\sigma$ is a partial ordering.
(2) If $\prec$ is well-founded, then $\prec_\sigma$ is well-founded.
(3) If $\prec$ is total, then $\prec_\sigma$ is total.
(4) If $\prec$ has the replacement property, then $\prec_\sigma$ has the replacement property.
(5) $(\Sigma^*, \prec)$ is the initial segment of $((\Sigma \bigcup \{\sigma\})^*, \prec_\sigma)$ given by σ, i.e. the identity map from Σ^* into $\{w \in (\Sigma \bigcup \{\sigma\})^* \mid w \prec_\sigma \sigma\}$ is an order-isomorphism.

Definition 3.3

Let $<$ be a well-ordering of Σ. For each σ in Σ, the ordering $\prec^\sigma$ on $\{\tau \mid \tau \leq \sigma\}^*$ is defined by transfinite induction:

$$\prec^\sigma = (\bigcup\{\prec^\tau \mid \tau < \sigma\})_\sigma$$

The *kachinuki* ordering on Σ^* is

$$\prec = \bigcup\{\prec^\sigma \mid \sigma \in \Sigma\}.$$

Example 3.4

Let $\Sigma = \{\mathbf{a}, \mathbf{b}\}$ and $\mathbf{a} < \mathbf{b}$. Then the kachinuki ordering $\prec$ orders Σ^* thus:

$$\epsilon \prec \mathbf{a} \prec \mathbf{aa} \prec \cdots \prec \mathbf{b} \prec \mathbf{ba} \prec \mathbf{baa} \prec \cdots \prec \mathbf{ab} \prec \mathbf{aba} \prec \mathbf{abaa} \prec \cdots \prec \mathbf{bb} \prec \mathbf{bba} \prec \mathbf{bbaa} \prec \cdots.$$

Theorem 3.5

Let $<$ be a well-ordering of Σ. Then, for each $\sigma \in \Sigma$, $\prec^\sigma$ well-orders $(\{\tau \in \Sigma \mid \tau < \sigma\})^*$, and the kachinuki ordering $\prec$ well-orders Σ^*. Moreover, they have the replacement property. If $\rho < \sigma$, then $((\{\tau \mid \tau \leq \rho\})^*, \prec^\rho)$ is an initial segment of $((\{\tau \mid \tau \leq \sigma\})^*, \prec^\sigma)$.

Proof :

The proof is by transfinite induction. Assume that the above conditions hold for any τ such that $\tau < \sigma$. It almost immediately follows from the assumption and (5) of Lemma 3.2 that $\prec' = \bigcup\{\prec^\tau \mid \tau < \sigma\}$ is a total ordering with the replacement property. Assume that $\prec'$ is not well-founded, i.e., there exists an infinite descending sequence $w_1 \succ' w_2 \succ' \cdots$ in $\{\tau \mid \tau < \sigma\}^*$. Let ρ be such that $w_1 \in \{\tau \mid \tau \leq \rho\}^*$ and $\rho < \sigma$. Clearly, $(\{\tau \mid \tau \leq \rho\}, \prec^\rho)$ is $(\{\tau \mid \tau < \sigma\}, \prec')$ itself or its intial segment. Therefore, the sequence is also in $(\{\tau \mid \tau \leq \rho\}, \prec^\rho)$, which is well-ordered by the assumption, and the contradiction follows. Now, the claims of the theorem are immediate consequences of Lemma 3.2.

Theorem (Kachinuki algorithm)

Let $\prec$ be the kachinuki ordering on Σ^* defined from a well-ordering $<$ of Σ. Then $t \prec u$ if and only if one of the following hold:

(1) $t = \epsilon$ and $u \neq \epsilon$,

(2) $t = \sigma t'$, $u = \sigma u'$ for some $\sigma \in \Sigma$ and $t' \prec u'$,

(3) $t = \sigma t'$, $u = \tau u'$ for some $\sigma, \tau \in \Sigma$ such that $\sigma > \tau$ and $t \preceq u'$,

(4) $t = \sigma t'$, $u = \tau u'$ for some $\sigma, \tau \in \Sigma$ such that $\sigma < \tau$ and $t' \prec u$.

Proof :

By induction on (the length of t) $\times$ (the length of u).

The above theorem provides an algorithm for comparing two strings with respect to kachinuki ordering without look-ahead. The name "kachinuki" comes from the order of the bouts in team matches in Japanese judo, because of resemblance to this algorithm. It is easily shown as a corollary of the above theorem that kachinuki ordering is actually a string version of recursive path ordering [2] (or its generalization [3, 8]). Thus, we can obtain another proof of Theorem 3.5 because recursive path ordering defines a well-founded ordering on terms.

Theorem 3.6

Let α be the order type [9] of a well-ordering $<$ of Σ and α^* denote the order type of the kachinuki ordering on Σ^* defined from $<$. Then $0^*=1$, $(n+1)^*=\omega^{\omega^n}$ for all $n<\omega$ and $\alpha^*=\omega^{\omega^\alpha}$ for all $\alpha\geq\omega$.

Proof :

If the order type of Σ is 0, then $\Sigma=\emptyset$. Hence, $\Sigma^*=\{\epsilon\}$, and therefore, $0^*=1$. By the definition of kachinuki ordering,

$$1^*=\lim_{n\to\omega}(1+1^2+\cdots+1^n)=\omega=\omega^{\omega^0},$$
$$(\alpha+1)^*=\lim_{n\to\omega}(\alpha^*+(\alpha^*)^2+\cdots+(\alpha^*)^n)=\lim_{n\to\omega}(\alpha^*)^n=(\alpha^*)^\omega \quad \text{if} \quad \alpha\geq 1,$$
$$\alpha^*=\lim_{\beta\to\alpha}\beta^* \quad \text{if} \quad \alpha(\neq 0) \text{ is a limit ordinal.}$$

The proof is completed by transfinite induction, because if α^* can be expressed by ω^{ω^β}, then

$$(\alpha+1)^*=(\alpha^*)^\omega=(\omega^{\omega^\beta})^\omega=\omega^{\omega^\beta\times\omega}=\omega^{\omega^{\beta+1}}.$$

4. Applications

In this section we report on the automated generation of rewriting systems for the Jantzen monoid and the Greendlinger group.

Definition 4.1

The *Jantzen monoid* is the monoid presented by the Thue system

$$J=\{(\mathbf{abbaab},\epsilon)\}$$

over $\{\mathbf{a},\mathbf{b}\}$.

Otto showed the Jantzen monoid is isomorphic to the monoid presented by

$$J_1=\{(\mathbf{a\bar{a}},\epsilon),(\mathbf{\bar{a}a},\epsilon),(\mathbf{x\bar{x}},\epsilon),(\mathbf{\bar{x}x},\epsilon),(\mathbf{\bar{a}xa},\mathbf{\bar{x}\bar{x}})\}$$

over $\{\mathbf{a},\mathbf{\bar{a}},\mathbf{x},\mathbf{\bar{x}}\}$, and gave a rewriting system for J_1 with a proof that it is complete [6]. We will show that the same rewriting system can be obtained by the Knuth-Bendix algorithm based on kachinuki ordering. The following is the process by which the rewriting system is generated, where the total ordering on the alphabet is specified as $\mathbf{x}<\mathbf{\bar{x}}<\mathbf{a}<\mathbf{\bar{a}}$.

$\mathbf{a\bar{a}}=\epsilon$
$\mathbf{\bar{a}a}=\epsilon$
$\mathbf{x\bar{x}}=\epsilon$
$\mathbf{\bar{x}x}=\epsilon$
$\mathbf{\bar{a}xa}=\mathbf{\bar{x}\bar{x}}$

1: $\mathbf{a}\bar{\mathbf{a}} \to \epsilon \Leftarrow 0$

2: $\bar{\mathbf{a}}\mathbf{a} \to \epsilon \Leftarrow 0$

3: $\mathbf{x}\bar{\mathbf{x}} \to \epsilon \Leftarrow 0$

4: $\bar{\mathbf{x}}\mathbf{x} \to \epsilon \Leftarrow 0$

5: $\bar{\mathbf{a}}\mathbf{x}\mathbf{a} \to \bar{\mathbf{x}}\bar{\mathbf{x}} \Leftarrow 0$

6: $\mathbf{a}\bar{\mathbf{x}}\bar{\mathbf{x}} \to \mathbf{x}\mathbf{a} \Leftarrow 5/1$

7: $\mathbf{a}\bar{\mathbf{x}} \to \mathbf{x}\mathbf{a}\mathbf{x} \Leftarrow 4/6$

delete 9

delete 6

8: $\bar{\mathbf{a}}\mathbf{x} \to \bar{\mathbf{x}}\bar{\mathbf{x}}\bar{\mathbf{a}} \Leftarrow 1/5$

delete 5

9: $\bar{\mathbf{x}}\bar{\mathbf{x}}\bar{\mathbf{a}}\bar{\mathbf{x}} \to \bar{\mathbf{a}} \Leftarrow 3/8$

10: $\bar{\mathbf{x}}\bar{\mathbf{a}}\bar{\mathbf{x}} \to \mathbf{x}\bar{\mathbf{a}} \Leftarrow 9/3$

delete 9

11: $\bar{\mathbf{a}}\bar{\mathbf{x}} \to \mathbf{x}\mathbf{x}\bar{\mathbf{a}} \Leftarrow 10/3$

delete 10

12: $\mathbf{a}\mathbf{x}\mathbf{x}\bar{\mathbf{a}} \to \bar{\mathbf{x}} \Leftarrow 11/1$

13: $\mathbf{a}\mathbf{x}\mathbf{x} \to \bar{\mathbf{x}}\mathbf{a} \Leftarrow 2/12$

delete 12

1: $\mathbf{a}\bar{\mathbf{a}} \to \epsilon$

2: $\bar{\mathbf{a}}\mathbf{a} \to \epsilon$

3: $\mathbf{x}\bar{\mathbf{x}} \to \epsilon$

4: $\bar{\mathbf{x}}\mathbf{x} \to \epsilon$

7: $\mathbf{a}\bar{\mathbf{x}} \to \mathbf{x}\mathbf{a}\mathbf{x}$

8: $\bar{\mathbf{a}}\mathbf{x} \to \bar{\mathbf{x}}\bar{\mathbf{x}}\bar{\mathbf{a}}$

11: $\bar{\mathbf{a}}\bar{\mathbf{x}} \to \mathbf{x}\mathbf{x}\bar{\mathbf{a}}$

13: $\mathbf{a}\mathbf{x}\mathbf{x} \to \bar{\mathbf{x}}\mathbf{a}$

The equations above the first horizontal line show the given Thue system. The generated rules are between the horizontal lines. The symbol $\Leftarrow 0$ means that the rule was obtained from one of the initially given pairs. The symbol $\Leftarrow n/m$ means that the equation was obtained from a critical pair generated by the previous rules n and m. The line "delete n" shows that rule n was removed at **Step 3** of the Knuth-Bendix algorithm because the left or right side of the rule was reducible by the newly-obtained rule. The set of rules under the second horizontal line is the resulting complete rewriting system. This agrees with the presentation by Otto [6].

Definition 4.2

The *Greendlinger group* is the group with three generators $\mathbf{a}, \mathbf{b}, \mathbf{c}$ satisfying the equation $\mathbf{abc} = \mathbf{cba}$. It is defined by the Thue system

$$G = \{(\mathbf{a}\bar{\mathbf{a}}, \epsilon), (\bar{\mathbf{a}}\mathbf{a}, \epsilon), (\mathbf{b}\bar{\mathbf{b}}, \epsilon), (\bar{\mathbf{b}}\mathbf{b}, \epsilon), (\mathbf{c}\bar{\mathbf{c}}, \epsilon), (\bar{\mathbf{c}}\mathbf{c}, \epsilon), (\mathbf{abc}, \mathbf{cba})\}.$$

It has been demonstrated that there is no Knuth-Bendix ordering for G such that the Knuth-Bendix algorithm terminates [4]. However, using kachinuki ordering, the Knuth-Bendix algorithm does terminate.

$\mathbf{a\bar{a}} = \epsilon$
$\mathbf{\bar{a}a} = \epsilon$
$\mathbf{b\bar{b}} = \epsilon$
$\mathbf{\bar{b}b} = \epsilon$
$\mathbf{c\bar{c}} = \epsilon$
$\mathbf{\bar{c}c} = \epsilon$
$\mathbf{abc} = \mathbf{cba}$

1: $\mathbf{a\bar{a}} \rightarrow \epsilon \Leftarrow 0$
2: $\mathbf{\bar{a}a} \rightarrow \epsilon \Leftarrow 0$
3: $\mathbf{b\bar{b}} \rightarrow \epsilon \Leftarrow 0$
4: $\mathbf{\bar{b}b} \rightarrow \epsilon \Leftarrow 0$
5: $\mathbf{c\bar{c}} \rightarrow \epsilon \Leftarrow 0$
6: $\mathbf{\bar{c}c} \rightarrow \epsilon \Leftarrow 0$
7: $\mathbf{abc} \rightarrow \mathbf{cba} \Leftarrow 0$
8: $\mathbf{cba\bar{c}} \rightarrow \mathbf{ab} \Leftarrow 5/7$
9: $\mathbf{ba\bar{c}} \rightarrow \mathbf{\bar{c}ab} \Leftarrow 8/6$
delete 8
10: $\mathbf{a\bar{c}} \rightarrow \mathbf{\bar{b}\bar{c}ab} \Leftarrow 9/4$
delete 9
11: $\mathbf{\bar{a}cba} \rightarrow \mathbf{bc} \Leftarrow 7/2$
12: $\mathbf{\bar{a}cb} \rightarrow \mathbf{bc\bar{a}} \Leftarrow 1/11$
delete 11
13: $\mathbf{bc\bar{a}\bar{b}} \rightarrow \mathbf{\bar{a}c} \Leftarrow 3/12$
14: $\mathbf{c\bar{a}\bar{b}} \rightarrow \mathbf{\bar{b}\bar{a}c} \Leftarrow 13/4$
delete 13
15: $\mathbf{\bar{a}\bar{b}} \rightarrow \mathbf{\bar{c}\bar{b}\bar{a}c} \Leftarrow 14/6$
delete 14

1: $\mathbf{a\bar{a}} \rightarrow \epsilon$
2: $\mathbf{\bar{a}a} \rightarrow \epsilon$
3: $\mathbf{b\bar{b}} \rightarrow \epsilon$
4: $\mathbf{\bar{b}b} \rightarrow \epsilon$
5: $\mathbf{c\bar{c}} \rightarrow \epsilon$
6: $\mathbf{\bar{c}c} \rightarrow \epsilon$
7: $\mathbf{abc} \rightarrow \mathbf{cba}$
10: $\mathbf{a\bar{c}} \rightarrow \mathbf{\bar{b}\bar{c}ab}$
12: $\mathbf{\bar{a}cb} \rightarrow \mathbf{bc\bar{a}}$

15: $\bar{\mathbf{a}}\bar{\mathbf{b}} \to \bar{\mathbf{c}}\bar{\mathbf{b}}\bar{\mathbf{a}}\mathbf{c}$

The resulting complete rewriting system is the same as Otto's, because we adjusted the ordering of the alphabet thus:

$$\mathbf{b} < \mathbf{c} < \bar{\mathbf{b}} < \bar{\mathbf{c}} < \mathbf{a} < \bar{\mathbf{a}}$$

so as to get the same result. However, we could have derived another complete rewriting system if we had imposed a different ordering.

Let us consider again the Jantzen monoid. It is easy, but not trivial, to verify that J and J_1 are isomorphic. Moreover, since the above rewriting system for the Jantzen monoid does not include the letter **b**, we have to know how to represent the letter **b** in J_1 if we want to solve the original word problem for J. Without knowing that **x** represents **ab**, it may take considerable time to solve these problems.

In this sense, the infinite rewriting system obtained by Potts [7] for the Jantzen monoid is more straightforward, because the modification made by him involves no more than adding two new letters to represent special strings.

Even based on kachinuki ordering, the Knuth-Bendix algorithm does not terminate for the original presentation J of the Jantzen monoid.

Theorem 4.3

There is no finite complete rewriting system for J such that the rewriting rules are oriented according to kachinuki ordering.

Proof :

We will omit the details, since the proof is analogous to Otto's Theorem 1 [6], which claims the same conclusion for the Knuth-Bendix ordering. The key point of the proof is that $u \prec v$ for every u and v such that $|u|_a < |v|_a$ and $|u|_b < |v|_b$, where $|u|_a$ and $|u|_b$ denote the number of occurences of the letter **a** and **b** in u, respectively.

Nevertheless, if we add new letters **x** and **y** to represent **ab** and **ba** respectively, the Knuth-Bendix algorithm can generate a finite rewriting system. The ordering of the alphabet is specified as $\mathbf{x} < \mathbf{y} < \mathbf{a} < \mathbf{b}$.

$\mathbf{abbaab} = \epsilon$
$\mathbf{x} = \mathbf{ab}$
$\mathbf{y} = \mathbf{ba}$

1: $\mathbf{ab} \to \mathbf{x} \Leftarrow 0$
2: $\mathbf{ba} \to \mathbf{y} \Leftarrow 0$
3: $\mathbf{ay} \to \mathbf{xa} \Leftarrow 2/1$
4: $\mathbf{bx} \to \mathbf{yb} \Leftarrow 1/2$
5: $\mathbf{xyx} \to \epsilon \Leftarrow 0$
6: $\mathbf{yx} \to \mathbf{xy} \Leftarrow 5/5$
delete 5
7: $\mathbf{xxy} \to \epsilon \Leftarrow 5$

8: **axy→xax** ⇐6/3

9: **xaxx→a** ⇐6/8

10: **yyby→b** ⇐7/4

11: **yby→xxb** ⇐10/7

delete 10

12: **axxb→ε** ⇐11/3

13: **aaxx→xaxa** ⇐9/9

14: **yaxx→xyya** ⇐13/2

15: **axx→xya** ⇐14/7

delete 14

delete 13

delete 12

delete 9

16: **by→xxxxb** ⇐11/7

delete 11

1: **ab→x**

2: **ba→y**

3: **ay→xa**

4: **bx→yb**

6: **yx→xy**

7: **xxy→ε**

8: **axy→xax**

15: **axx→xya**

16: **by→xxxxb**

Thus, we can mechanically obtain a complete rewriting system for another monoid whose isomorphism to the Jantzen monoid is as straightforward as Potts'. Moreover, the resulting rewriting system is finite.

5. Concluding Remarks

In this paper, we reported on an application of the Knuth-Bendix algorithm based on kachinuki ordering; a very powerful tool for constructing finite complete rewriting systems.

Besides Knuth-Bendix ordering, multiset ordering [1] and lexicographic ordering are used as ordering methods for strings consisting of elements of a given partially ordered set.

As suggested by its name, however, multiset ordering is ordering for multiple sets, i.e., collections of elements that may have multiple occurences of identical elements without regard to the order of occurences. Therefore, in multiset ordering, we cannot compare two strings consisting of an identical number of the same letters arranged in different orders.

On the other hand, lexicographic ordering does not produce, in general, a well-founded ordering even if the alphabet is well-founded. If we want the ordering to be well-founded, we have to use length as a criterion, which introduces too strong a constraint on ordering.

Since kachinuki ordering is free of these disadvantages, we believe that it has a very wide range of application. For example, a new ordering method for terms is obtained by using kachinuki ordering instead of multiset ordering in the definition of recursive path ordering [2].

ACKNOWLEDGEMENTS

The author wishes to express his thanks to K. Fuchi, Director, and T. Yokoi, Third Laboratory Chief, of the Institute for New Generation Computer Technology (ICOT) for providing the opportunity to conduct this research. Thanks are also due to Mr. K. Yuyama of Hitachi Ltd. and Prof. M. Takahashi of Tokyo Institute of Technology for their helpful suggestions.

REFERENCES

[1] Dershowitz, N. and Manna, Z.: *Proving termination with multiset orderings*, Comm. ACM 22 (1979) 465-467.

[2] Dershowitz, N.: *Orderings for term-rewriting systems*, Theoretical Computer Science 17 (1982) 279-301.

[3] Kamin, S. and Levi, J-J.: *Two generalizations of the recursive path orderings*, Unpublished note, Department of Computer Science, university of Illinois, Urbana, IL (1980).

[4] Kemmerich, S.: *Unendlinche reduktionssysteme*, Dissertation, TH Aachen (1983).

[5] Knuth, D. E., Bendix, P. B.: *Simple word problems in universal algebras*, Computational problems in abstract algebra, J. Leech (ed), Pergamon Press, Oxford, (1970) 263-297. also in: Automated Reasoning 2 (Siekmann and Wrightson eds.), Springer (1983).

[6] Otto, F.: *Finite complete rewriting systems for the Jantzen monoid and the Greendlinger group*, Theoretical Computer Science 32 (1984) 249-260.

[7] Potts, D.: *Remarks on an example of Jantzen*, Theoret. Comput. Sci. 29 (1984) 277-284.

[8] Sakai, K.: *An ordering method for term rewriting systems*, Technical Report 062, ICOT (1984).

[9] Jech, T.: *Set theory*, Academic Press (1978).