Information Processing Letters 138 (2018) 61–66

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

The shortest kinship description problem ☆

Chao Xu [a,*], Qian Zhang [b]

[a] Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, United States of America
[b] School of Civil and Environmental Engineering, Cornell University, Ithaca, NY 14853, United States of America

ARTICLE INFO

Article history:
Received 16 October 2017
Received in revised form 25 May 2018
Accepted 8 June 2018
Available online 15 June 2018
Communicated by Kun-Mao Chao

To the memories of Jiaqi Zhao (1994–2016)

Keywords:
Algorithms
Rewriting system
Kinship
Shortest path
Monoid

ABSTRACT

We consider a problem in descriptive kinship systems, namely finding the shortest sequence of terms that describes the kinship between a person and his/her relatives. The problem reduces to finding the minimum weight path in a labeled graph where the label of the path comes from a regular language. The running time of the algorithm is $O(n^3 + s)$, where n and s are the input size and the output size of the algorithm, respectively.

1. Introduction

Consider a person who is inexperienced with (consanguineous) kin terms in a particular language (e.g. Chinese). She tries to form a description of her relation to a kin. A desirable description should be concise, and only use terms she knows. This captures the shortest kinship description problem (**SKDP**). The scenario sounds like a small hurdle exclusive to new language learners. But in reality, this can be a frustrating problem even for native speakers. The Chinese kinship terminology is very complicated. There are few applications built to address the complication. LessLoop Limited developed an application to calculate the correct Chinese kin term. The application gathered over 100,000 downloads and a wide media coverage [1]. Mi Calculator implemented a similar feature before Chinese New Year 2017. It helped users with addressing their visiting relatives in a correct manner [2]. Yet, both applications give up when the relationship is too complicated: neither application can report a description involving more than *one* term. For example, "my maternal granddaughter's maternal granddaughter" is a relation that cannot be described with a single Chinese term. This prompts the investigation in this paper.

A *kin type* is an abstract concept of the kinship between two people. The concept of mother is a kin type. A monoid is an algebraic structure with a binary associative operation, called product, and an identity element. The kinship monoid is an algebraic model of kin types. The monoid has four primitives: `f`, `m`, `s` and `d` representing father, mother, son and daughter, respectively (we will use typewriter font to indicate the 4 elements). The product of the elements represents the composition of the relations. For example, `fsm` would be "(ego's) father's son's mother". Some products represent the same kin type. For example, `fs` and `ms` both represent brother. A *kin term* is a name people use to refer to a kin type. For instance, "paternal grandfather" for `ff`, "brother" for `fs` and "wife" for `sm`. Kin terms depend

☆ Chao is supported in part as a State Farm Companies Foundation Doctoral Scholar.

* Corresponding author.
E-mail addresses: chaoxu3@illinois.edu (C. Xu), qz283@cornell.edu (Q. Zhang).

https://doi.org/10.1016/j.ipl.2018.06.005
0020-0190/

on the language, dialect, and culture [14]. The kin terms also compose by taking product of their representing kin types. The goal is to have a concise expression of a kin type through composing available kin terms.

Related work Modeling aspects of kin types and kin terms as a monoid has a long history (see [12] for a historical overview). Murdock used a similar formalism of the kinship monoid, but had four extra primitives [10]. Morgan outlined six major kinship systems [9]. Our algorithm handles the Sudanese system, the most complicated one. For non-Sudanese patterns, Boyd analyzed Arunta, Kariera and Ambrym kinship algebraically [6]. Read analyzed the American kinship and found the underlying space is $\mathbb{Z}^2$ [12].

For a monoid M, a subset S and $x \in M$, the *submonoid membership problem* asks for a product of some elements in S that equals x. The optimization version minimizes the number of elements in the product. It is called the *submonoid membership optimization problem* (**SMOP**). The shortest kinship description problem is a special case of **SMOP**. In general, **SMOP** is undecidable even for simple algebraic structures [11]. Hence, it is unclear whether an algorithm for the shortest kinship description problem exists.

Our contribution We give the first algorithm that runs in polynomial time with respect to the input size and linear time with respect to the output size. It is also the first algorithm that can handle the case where the output is more than a single term.

2. Preliminaries

2.1. Rewriting systems

We assume standard knowledge on regular and context-free languages. For an introduction, see [13]. We adapt the notations for rewriting systems in [5]. We review a few standard notions. Let X be a set of strings, X^* is the *Kleene star* operation: the smallest superset of X that contains the empty string and is closed under string concatenation. If X is set of symbols, X^* is the set of all strings using symbols in X. For a non-terminal A in a context-free grammar, $\mathbb{L}(A)$ is the set of all strings generated by A. For a regular expression R, $\mathbb{L}(R)$ is the set of strings matched by R. A context-free grammar is a *Chomsky normal form* if all rules are of the form $A \to BC$ or $A \to a$ for some non-terminals A, B and C, and terminal a. Each context-free language has a context-free grammar in Chomsky normal form. A *(string) rewriting system* R on X is a relation on X^*. A *(rewriting) rule* is an element in a rewriting system. Instead of (a, b), we write $a \to b$ to emphasis it is a rule. If $a \to b \in R$, $w = uav$, and $w' = ubv$ for some strings u and v, we write $w \to_R w'$. w' is the result after *applying* rule $a \to b$ to w. $w \leftrightarrow_R w'$ if either $w \to_R w'$ or $w' \to_R w$. $w \xrightarrow{*}_R w'$ if $w = w'$ or there exists w'' such that $w \to_R w''$ and $w'' \xrightarrow{*}_R w'$. w *derives* w' if $w \xrightarrow{*}_R w'$. $w \leftrightarrow^*_R w'$ if $w = w'$ or there exists w'' such that $w \leftrightarrow_R w''$ and $w'' \leftrightarrow^*_R w'$. $\leftrightarrow^*_R$ is an equivalence relation, and we denote the equivalence class containing w as $[w]_R = \{ w' \mid w' \leftrightarrow^*_R w \}$.

A string w is R *normal* with respect to a rewriting system R, if $w \xrightarrow{*}_R w'$ implies $w = w'$. A normal string y is called a *normal form* of a string x if $x \xrightarrow{*}_R y$. R is *convergent* if every string has a unique normal form. The unique normal form of w is denoted as $\underline{w}$ when there is no ambiguity about the rewriting system. A rule $a \to b$ is *length reducing* if $|a| > |b|$, *length preserving* if $|a| = |b|$, and *length non-increasing* if it is either length reducing or length preserving.

For a rewriting system R on alphabet X and $L \subset X^*$, the set of strings can be derived from some string in L is $\Delta^*_R(L) = \{ w' \mid w \in L, w \xrightarrow{*}_R w' \}$. The set of strings that derive some string in L is $\nabla^*_R(L) = \{ w \mid w' \in L, w \xrightarrow{*}_R w' \}$. If R is a convergent rewriting system, $w \leftrightarrow^*_R w'$ if and only if $\underline{w} = \underline{w'}$ [3, Theorem 2.1.9].

2.2. The kinship monoid and problem definition

A presentation is an ordered pair $\langle X \mid R \rangle$. X is the *generating set* or *alphabet*, and $R \subset X^* \times X^*$ are the *relators*. We write $S = \langle X \mid R \rangle$ if S is a monoid isomorphic to $X^*/\leftrightarrow^*_R$ (i.e. the monoid on $\{ [x]_R \mid x \in X^* \}$ where the product is defined as $[x]_R[y]_R = [xy]_R$). For more introduction on presentation, see [7].

We proceed to define the kinship monoid. The alphabet of the kinship monoid is $\Sigma = \{ \mathtt{f}, \mathtt{m}, \mathtt{s}, \mathtt{d} \}$. A *word* is another name for strings in Σ^*. A sign string for a word $w \in \Sigma^*$, denoted $\mathrm{sgn}(w)$, is the string obtained by replacing each $\mathtt{f}$ and $\mathtt{m}$ with $+$, and each $\mathtt{s}$ and $\mathtt{d}$ with $-$.

We define a set of relators Γ. For all $a, b, c \in \Sigma$,

1. (abc, c) is in Γ if $\mathrm{sgn}(abc) \in \{ +-+, -+- \}$, (generation cancellation)
2. (abc, a) is in Γ if $\mathrm{sgn}(abc) = --+$, and $\{a, c\} = \{\mathtt{s}, \mathtt{f}\}$ or $\{a, c\} = \{\mathtt{d}, \mathtt{m}\}$, (Child's child's parent is child if gender agrees)
3. (ac, bc) is in Γ if $\mathrm{sgn}(ac) = \mathrm{sgn}(bc) \in \{ -+, +- \}$. (siblings/spouse)

Γ is also a rewriting system on Σ. We will use $[w]$ to denote $[w]_\Gamma$. The *kinship monoid* is $K = \langle \Sigma \mid \Gamma \rangle$.

Problem 1 *(**SMOP**). Let M be a monoid, given $X \subseteq M$ and $y \in M$ with cost function $\$: X \to \mathbb{N}$. Find $x_1, \ldots, x_n \in X$ such that $y = x_1 \cdot \ldots \cdot x_n$ and $\sum_i \$(x_i)$ is minimized, or return no solution exists.*

The shortest kinship description problem is **SMOP** on K, the kinship monoid. X is the kin type of the allowed kin terms. Each kin type $x \in K$ is given as an element in $w \in \Sigma^*$, such that x is the product of w, when w is seen as a sequence of elements in K. We also want a concise output. Instead of outputting the sequence of elements in X, we order the element in X, and output the sequence of indices. Hence we obtain the *shortest kinship description problem (**SKDP**).*

Problem 2 *(Shortest kinship description problem). Let $K = \langle \Sigma \mid \Gamma \rangle$ be the kinship monoid. $W = w_1, \ldots, w_k$ is a sequence of elements in Σ^* and $w \in \Sigma^*$ is the target ele-*