RMXにおける関数形式アドレスおよびデバッグ支援機能の実装

北囿　達也† 　青山　陽亮† 　遠山元道††

† 慶應義塾大学理工学部情報工学科　〒 223–8522 神奈川県横浜市港北区日吉 3–14–1
E-mail: †{zonop,bluemountain}@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし　ルールベースメール配送システム RMX において，従来のアドレス表記方法では配送ルールやサブドメインを全て'.' で区切り並記するため，パラメータと配送ルールの関係を始めとするメールアドレスの構造が理解し辛いという問題があった．そこで本論文では関数的なメールアドレスの記述方法を提案し，メールアドレスの構造をより直感的に理解できるようにした．
また，RMX にはメールアドレスの誤表記等があると，送信者の予期しないメールアドレスにまでメールが送られてしまう可能性があった．情報の漏洩となり得るこの問題を防ぐため，送信先アドレスをあらかじめ確認する事が出来る，デバッグコマンドを提案した．

キーワード　RMX，電子メール

Implementation of Functional Form Address and Debug Support Features in RMX

Tatsuya KITAZONO†, Yosuke AOYAMA†, and Motomichi TOYAMA††

† ††Department of Information and Computer Science ,
Keio University
Hiyoshi 3–14–1, Kouhoku-ku, Yokohama-shi, Kanagawa, 223–8522 Japan
E-mail: †{zonop,bluemountain}@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

1. はじめに

RMX(Rule-based e-Mail Exchange system) はユーザが設定したルールとメールアドレスを基に，データベースのアクセスを行い，得られたユーザ集合に対してメールを配信するメール転送エージェントである．従来のメールアドレス表記方法はパラメータ，配送ルール，ドメインを全て'.' で区切り並記するため，パラメータと配送ルールの関係や配送ルールとドメインの区別が理解し辛いという問題があった．ユーザにとって理解し辛い点はシステムを導入する上での妨げになると考えられ，可能な限り取り除かれるべきである．そこで本論文では新たなアドレス表記方法として，中括弧'{}' を用いた関数形式アドレスを提案した．これにより，パラメータと配送ルールの関係が一目で分かる様になり，より直感的に構造を理解できるアドレスを実現した．関数形式アドレスは従来形式アドレスで表記される全てのアドレスを表現できる上，従来形式アドレスでは制約により不可能だった配送ルール間のユニオンを実現できるという点から，記述力においても従来形式アドレスより優れている．

また，RMX は 1 つのメールアドレスで配送ルールに基づいた多数のメールアドレスへメールが送信されるため，メールアドレスの誤表記等があると送信者の予期しないメールアドレスにまでメールが送信される可能性があった．メールの内容が無関係の人間に公開されるべきではないものであった場合，重大な機密漏洩に繋がる恐れがある．この問題を解決するため，本論文ではデバッグ支援機能として，3 種類のデバッグコマンドを提案した．1 つ目の count は送信されるアドレスの数を返し，2 つ目の expand は送信されるアドレスの一覧を返す．3 つ目の explain は適用された配送ルールごとにパラメータと送信先アドレスを表記した上で，最終的な送信先アドレス一覧を返す．これらは従来のアドレスの先頭にコマンドを加えるだけで利用でき，送信者以外にメールが送信される事なく，送信先アドレスやその数をあらかじめ知る事が出来る．その際にメールアドレスがあらゆる人に知られてしまう事がない様，デバッグコマンドの使用には制限を設けている．

以下，本稿の構成を示す．まず 2 章で RMX の概要を述べる．3 章で関数形式アドレス，4 章でデバッグコマンドについて述べ，5 章で関数形式アドレスの評価を行い，6 章で結論を述べる．

2. RMX

Rule-based e-Mail eXchange(RMX) は遠山研究室が提案している電子メール配信方式である．一般的にメール配信に利用されているメーリングリスト (ML) ではメーリングリストを代表するアドレスにメールが送信されるとメーリングリストに所属するメンバー全員に対してメールが配信される．

$$< ML \text{ アドレス } >:=< ML \text{ 名 } > @ < \text{ドメイン } >$$

メーリングリストではメンバーのメールアドレスの更新など管理者の作業負担が必要となる．例えば大学でのメーリングリストの利用を考える．学生がメールアドレスを変更した場合にはクラス，研究室，プロジェクトなど所属する全てのメーリングリストでアドレスの変更を行わなければならない．

それに対して RMX では下記のような記述により複数の送信先を指定する．

$$< RMX \text{ のメール配信先指定 } >:=< \text{パラメータ } > @$$
$$< \text{配送ルール名 } > . < \text{サブドメイン } > . < \text{ドメイン } >$$

RMX のメールアドレスは以上のように配送範囲記述部分とドメイン部分が”.”の記号で区分されている．配送範囲記述部は一つ以上のパラメータの組み合わせで構成され，@記号によってパラメータと配送ルール名に分けられる．サブドメインは後述する設定ファイルの名前に相当する．RMX はこのような配送範囲記述を受け取る．そして，指定された配送ルールとそのパラメータに基づきデータベースに問い合わせを行い実際の送信先アドレスを得る．最終的に得られたメールアドレスに基づき配送が行われる．図 1

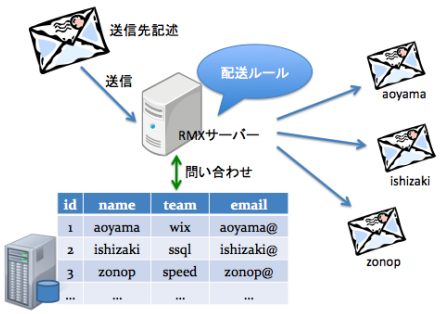

図 1　RMX におけるメール配信の流れ

RMX ではメール配信に必要な情報は全て利用組織のデータベース中で管理されている．そのため，従来のメーリングリストのように一つ一つ登録アドレスを更新する必要はない．また，送信者は配送ルールの記述により柔軟に送信範囲を指定できるため，メーリングリストのようにいくつものグループを用意する必要はない．例えば大学でメーリングリストを作成する場合を想定すると，研究室，学年，学科などの連絡を多く取るであろうグループ毎に，メーリングリストを用意しなければならない．

2.1 配送ルール

配送ルールとは配送範囲記述とそれに基づき送信先メールアドレスを得るクエリを関連付けるルールである．配送ルールは以下のように定義する．

配送ルール名
Type:パラメータの型
query: 送信先メールアドレスを得るためのクエリ

query 部分は SQL によって記述される．RMX は記述された配送ルール名に対応するクエリにパラメータを挿入し，問い合わせを行うことで送信先メールアドレスの集合を得る．このような配送ルールを用いることにより，利用者は簡潔な記述で配送範囲を指定することができる．以下に配送ルール定義の例を示す．

```
grade
gradeType= integer
grade[1]= select s.address from student s
where s.grade = $1 ;
```

上記の例では学年を integer 型で受け取り，それに基づいてメール配信を行う grade ルールを定義している．メールの宛先が 4@grade.example.edu の場合，図 2 のように query 部分で利用者のメールアドレスと学年が格納されている表 student から，学年が 4 年の学生のメールアドレスを得るクエリを記述している．

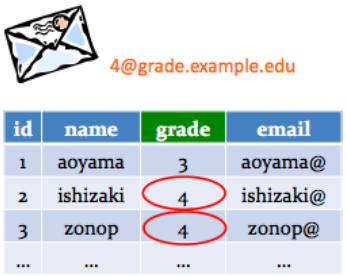

図 2　grade ルールと配送例

2.2 複数の配送ルールの組み合わせ

ここでは，複数の配送ルールを組み合わせてメールを送る際に，RMX で使用可能な演算子について説明する．演算子を Envelope-To フィールドの配送ルールに使用することにより，ユーザに対してより詳細な配送範囲の指定を可能とする．図 3 では，演算子の詳細を表にしている．

演算子	ターゲット	意味	使用場所	優先順位
.	ルール	積集合	○@○	低
+	パラメータ	和集合	○@×	↕
-		多相	○@×	高

図 3　RMX の演算子

2.2.1 積 集 合

Syntax: $< par_1 >$ . ... . $< par_n >$ @ $< name_1 >$ . ... . $< name_n >$. $< subdomain >$. $< domain >$

Semantics: $name_1\ (par_1)\ \cap\ \ldots\ \cap\ name_n\ (par_n)$

“.” は，Envelope-To フィールドに複数の配送ルールを使用したい時に用いられる演算子である．指定された複数の配送ルールの結果の積集合を取り，最終的に得られた結果に対して配送を行う．この演算子を使用するときは，“.” で区分されたパラメータと配送ルール名の数を一致させ，@を挟んでパラメータとそれに対応する配送ルール名を順番に並べる．

例：

toyama.1@lab.grade.example.edu

2.2.2 和 集 合

Syntax: $< par_1 >$ + ... + $< par_n >$ @ $< name_1 >$. $< subdomain >$. $< domain >$

Semantics: $name_1\ (par_1)\ \cup\ \ldots\ \cup\ name_1\ (par_n)$

“+” は，ある配送ルールに対して論理和を得たい複数のパラメータを指定するときに用いられる．与えられた複数のパラメータをそれぞれ配送ルールのクエリに代入し，得られた結果の和集合を取る．そして，最終的に得られた結果に対して配送を行う．この演算子は，パラメータにのみ有効で，複数の配送ルールに対して “+” を適用するということはできない．

例：

3+4@grade.example.edu

2.2.3 多相 (ポリモルフィック)

Syntax: $< par_1 >$ − ... − $< par_n >$ @ $< name_1 >$. $< subdomain >$. $< domain >$

Semantics: $name_1\ (par_1,\ ...,\ par_n)$

“-” は，ポリモルフィックな考え方を導入した演算子である．“-” によって与えられたパラメータの数により配送ルールから呼び出すクエリが異なる．

例：

kitazono-tatsuya@name.example.edu

以上の 3 つの演算子は組み合わせて使用することもでき，ユーザが配送範囲を指定する際の手助けを行う．メールアドレスは以下の形式を取る．

$ListOpe := -$
$UnionOpe := +$
$InterOpe := .$
$Arg :=$ string | integer
$ListPara := Arg \mid Arg\ ListOpe\ ListPara$
$UnionPara := ListPara \mid ListPara\ UnionOpe\ UnionPara$
$InterPara := UnionPara \mid UnionPara\ InterOpe\ InterPara$
$RuleList :=$ subdomain | rule $InterOpe\ RuleList$
$Address := InterPara$ @ $RuleList$. domain

例：

kitazono-tatsuya+aoyama.toyama@name.lab.example.edu

2.3 サブドメイン設定ファイル

サブドメイン設定ファイルでは，データベースへの接続や，使用するルールの設定を管理する．ファイル名をサブドメイン名.properties とし，.properties の前をサブドメイン名として使用する．例えば，sample.properties というファイルを作成すれば，サブドメイン sample とすることでこの設定ファイルを参照し，同様に sample1.properties というファイルを作成し，サブドメインを sample1 とすることでこちらの設定ファイルを参照する．サブドメイン設定ファイルでは以下の項目においてデータベース情報を設定する．

- dbDriver = ＜データベースドライバ＞
- dbUrl = ＜接続するデータベースの URL ＞
- dbId = ＜データベース上のユーザ ID ＞
- dbPassword = ＜ユーザ ID に対応するパスワード＞

使用するルールの設定もサブドメイン設定ファイルで行う．

3. 関数形式アドレス

従来のアドレス表記方法に基づいて，配送ルールが 3 種類あった場合のメールアドレスについて考える．

例：

zonop+toyama.3-5.basket@name.grade.club.lab.example.edu

この例では name，grade，club が配送ルールであり，lab がサブドメインである．ドメイン部において，それら全てが’.’ で区切り並記されているため，どこまでが配送ルールなのか，どのパラメータと対応しているのかが理解し辛い構造となっている．また，パラメータ部には複数の演算子が用いられており，その優先順位も分かりにくい．この様なユーザにとって理解し辛い要素は，システムを導入する上で大きな妨げとなる可能性がある．この問題を解決するため，本論文では関数形式アドレスを提案する．

関数形式アドレスは以下の形式を取る．

$ListOpe := -$
$UnionOpe := +$
$RuleOpe := . \mid +$
$Arg :=$ string | integer
$Para := Arg \mid Arg\ ListOpe\ Para$
$ParaList := Para \mid Para\ UnionOpe\ ParaList$
$Exp :=$ rule { $ParaList$ }

$ExpList := Exp \mid Exp\ RuleOpe\ ExpList$
$Address := ExpList$ @ subdomain . domain

学年が 3 年，または 4 年で，名前が kitazono である学生へメールを送る場合，2 つの形式ではそれぞれどのように表記するのか比較する．サブドメインは lab とする．

従来形式：

3+4.kitazono@grade.name.lab.example.edu

関数形式：

grade{3+4}.name{kitazono}@lab.example.edu

従来形式ではパラメータと配送ルールが離れて記述されていたので，どの配送ルールに対してどのパラメータが対応しているのか分かりにくかったのに対し，関数形式では中括弧で括る事でその関係をより明示的に表現している．これならば一目で対応する配送ルールとパラメータを理解する事が出来る．また，従来形式では配送ルールが増えるに連れて@マーク以降のドメイン部も随時書き換える必要があった．しかし関数形式であれば配送ルールはパラメータ部に記述するため，ドメイン部はサブドメインが同じである限り統一される．

3.1 ルール間のユニオン

前述した通り，論理和を得る演算子 “+” はパラメータにのみ有効で，複数の配送ルールに対して “+” を適用するということはできない．これはメールアドレスのドメイン部にはラテン文字，数字，’-’，’.’ 以外の文字を表記してはならないという制約があるためである．このため，例えば学年が 4 年である，または名前が kitazono である学生に対してメールを送ろうとした場合，従来形式であれば以下のようなアドレスとなる．

従来形式：

4.kitazono@grade+name.lab.example.edu

grade と name という 2 つの配送ルールに対して論理和を適用しようとしているのだが，上記の制約のためドメイン部に’+’ を用いる事は出来ず，この表現は実現不可能である．しかし関数形式アドレスは配送ルールもパラメータ部に記述するので，この表現が可能となる．

関数形式：

grade{4}+name{kitazono}@lab.example.edu

また，配送ルール間の積集合をとる’.’ と和集合をとる’+’ が同時に使用された場合には，積集合をとる’.’ の方が優先順位が高いものとする．例えば以下の様なメールアドレスが作成された場合，grade と club の積集合をとった後に，その結果と name との和集合をとる．

例：

name{kitazono}+grade{4}.club{basket}@lab.example.edu

4. デバッグコマンド

デバッグコマンドは以下の形式のように作成したアドレスの前に#でデバッグコマンドを挟んで用いる．従来形式，関数形式のアドレス共に使用する事が出来る．

従来形式：#デバッグコマンド# $<$ *para* $>$ @ $<$ *rule* $>$. $<$ *subdomain* $>$. $<$ *domain* $>$
関数形式：#デバッグコマンド# $<$ *rule* $>$ {$<$ *para* $>$}@ $<$ *subdomain* $>$. $<$ *domain* $>$

デバッグコマンドは以下の 3 種類がある．いずれもメールは送信アドレスに対してのみ送られ，メール本文が各コマンドに応じた内容に編集されている．

- count
  送信先アドレスの数を返す．
- expand
  最終的な送信先アドレスの一覧を返す．
- explain
  適用されたルールごとのパラメータと送信先アドレス，及び最終的な送信先アドレスを返す．

例として explain コマンドを使用した場合のメール本文を図 4 に示す．送信したメールアドレスは以下の通りである．testz がサブドメインとする．

例：

#explain#4.basket@grade.club.testz.db.ics.keio.ac.jp

```
Hi. This is the mail server at mail0.db.ics.keio.ac.jp.
This is Debug System. About the following address.

<4.basket@grade.club.testz.db.ics.keio.ac.jp>

Function : explain

[1] grade ( 4 )

zonop@db.ics.keio.ac.jp
sample@keio.jp

[2] club ( basket )

zonop@db.ics.keio.ac.jp
toyama@ics.keio.ac.jp


Result Addresses :
zonop@db.ics.keio.ac.jp
```

図 4　expalin コマンドによる返信メール本文

例のメールアドレスでは grade と club の 2 つの配送ルールが用いられているので，それぞれの配送ルールに対するパラメータと，その条件に当てはまるメールアドレスを表記している．更にそれらの積集合を求め，最終的な送信先アドレスを Result Address として表記する．

上記の例を見れば分かる通り，デバッグコマンドの中でも expand や explain はデータベース上のメールアドレスを直接表示する．送信先の数を返す count でさえも，メールアドレスの条件を変えて行けば送信対象となるメールアドレスがあるかどうか調べる事が出来る．RMX は現在，企業など限られた組織内で限定的に使われるという前提で開発されているので，組織内のメールにメールアドレスが表示されるのは問題ないと考えられるが，外部の人間が悪用すると簡単に機密情報を取得できてしまう．この問題を解決するため，デバッグコマンドを使用するには前述したサブドメイン設定ファイルにおいて送信元アドレスが登録されている必要がある．以下の形式に沿って記述する．

debugUser = address1,address2, ...

登録されていないメールアドレスからデバッグコマンドの使用要求が来た場合，RMX システムは使用する権限がない旨を返信する．

5. 評　　価

本研究では関数形式アドレスにおける評価を行う．架空の学校における学生データベースを用意し，それには学生の名前，学年，部活，性別などの属性があるものとする．被験者 11 名には用意されたデータベースと配送ルールがあった時に，どのようなメールアドレスにすれば希望通りの相手にメールが送信できるか，という問題形式でメールアドレスをパソコン上で作成してもらい，作成にかかった時間や正答率から評価を行う．配送ルールの数によって難易度は変化すると考えられるので，問題は配送ルール数で分類する．またこの時，被験者にはあらかじめ RMX の概要と各形式におけるメールアドレスの作成方法について説明しておく．更に例題を各形式で解いてもらい，すでにメールアドレスの作成には慣れている状態で実験を行うものとする．これは初見で実験を行うと慣れによる偏りが発生すると考えられるので，等しい条件下で比較を行うためである．

問題例：

- サッカー部かテニス部に所属している人 (従来形式)
- 1 年生から 3 年生の中で，佐藤という名前の人 (関数形式)

解答例：

- soccer+tennis@club.lab.sample.edu
- grade{1-3}.name{sato}@lab.sample.edu

5.1 作成時間による評価

被験者には問題群の中から各形式，各配送ルール数ごとに問題をランダムで複数選出し，連続で解答してもらった．作成時間は完全に正解となるメールアドレスが作成されるまでにかかった時間とし，その結果から配送ルール数ごとに作成時間の平均を取り，グラフにまとめたものを図 5 に示す．

図 5 を見ると，いずれの配送ルール数においても関数形式アドレスの方が作成時間が短くなっている事が分かる．この要因としてまず，関数形式アドレスはパラメータと配送ルールをペアで記述していけるという点が挙げられる．実際に被験者の様子を見ていると，関数形式の場合はペアごとにアドレスを記述してから次のペアへ，という流れで進めるので比較的迷いなく記述できていたのに対し，従来形式はパラメータと配送ルールを離して記述するため，確認の手間がかかっている被験者が多く見られた．従来形式で配送ルールとパラメータを交互に記述する被験者もいたが，その場合はカーソルの移動が手間となり，その分時間がかかってしまっていた．同じデータベースが対象であれば，関数形式ではドメイン部が全て同じになるという点も有効に働いたようである．

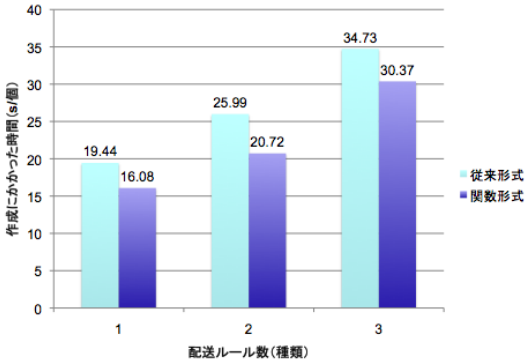

図 5　従来形式と関数形式によるアドレス作成時間比較

5.2 正答率による評価

同様の実験を時間制限を設けた状態で行い，どのような正答率となるか実験した．先ほどの実験における各形式，各配送ルール数の作成平均時間を制限時間とし，アドレスが正しくなかった場合と時間内に作成できなかった場合を誤答として扱った．また，出題形式と問題の分類方法は先ほどの実験と同じとする．なお，日常生活において時間制限が設けられた上でメールアドレスを作成する状況は非常に考えにくいものである．しかし無条件でメールアドレスの正答率を計測した場合，時間をかけて確認すればほぼ確実に正解となるメールアドレスを作成する事が出来てしまい，結果を比較する事が出来ない．本研究はよりユーザに理解しやすいメールアドレス記述方法として関数形式アドレスを提案し，その差を数値として表すべく実験を行っているので，差をより明確にするために設けた条件であると考えていただきたい．結果を図 6 に示す．正答率は百分率で表記する．

配送ルール数	従来形式	関数形式
1	93.9	93.9
2	87.9	90.9
3	75.8	84.8

図 6　従来形式と関数形式によるアドレス作成正答率比較

配送ルール数が 1 つの場合は形式による違いは見られない．誤答の内容も単なるパラメータの書き間違えやタイピングミス等，形式の違いが要因であるものではなかった．しかし配送

ルールの数が増えるに連れて，正答率には明確な差が見られるようになった．特に従来形式ではパラメータと配送ルールの順番を間違えたり，配送ルールに気を取られてサブドメインを記入し忘れる等，アドレスの構造が要因であると考えられる誤答が頻繁に見られた．正答率においても，関数形式アドレスの方が優勢であると言える．

6. 結　　論

本研究では RMX における関数形式アドレスおよびデバッグ支援機能の実装を行った．関数形式アドレスを実装する事により，パラメータと配送ルールの関係を始めとした構造が理解し辛かった従来形式のアドレスをより理解しやすくし，ユーザにとって扱いやすいものである事を証明した．また，従来形式では実現不可能だった配送ルール間のユニオンを可能にした．デバッグ支援機能の実装では 3 種類のデバッグコマンドを実装した．count は送信先アドレスの数，expand は送信先アドレスの一覧，explain は適用されたルールごとのパラメータと送信先アドレス，及び最終的な送信先アドレスを返し，あらかじめ送信されるメールアドレスを知る事が出来るようになった．

文　　献

[1] 高畑理, 藤沼健太郎, 石橋玲, 遠山元道. ”Magic Mirror Mailing：個人情報データベースを利用する柔軟なメイル配送システム”, 情報処理学会データベースシステム研究報告 Pages: 123-128 July 2001

[2] Kim Hanki, Sang-Gyu Shin, Motomichi Toyama. ”A Rule-Based Mailing System for an Organization”, International Workshop on INformation Processing over Evolving Networks, June 2006

[3] 原田哲志, 慎　祥揆, 遠山元道. “ RMX における電子メール送受信範囲管理方式の提案”, DBWS2007

[4] 青山陽亮, 遠山元道. “ RMX におけるポリモルフィックルールとメール本文編集機能の導入”, DEIM2010