知識工学と人工知能 57－5
（1988. 3. 10）

# ストリングパターンのユニフィケーション

松山 隆司　　張 汝濤

東北大学工学部情報工学科

ストリング処理における基本演算であるパターンマッチングを拡張し、ユニフィケーションを導入することにより、処理機能の向上を図ることができると考えられる。本論文では、定数記号、一文字列変数、任意文字列変数の並びとして定義されるストリングパターンのユニフィケーション・アルゴリズムを提案し、その完全性を証明する。論文では、まず正規ストリングパターンという限定されたストリングパターンに対する完全なアルゴリズムを示し、それに基づいて一般のストリングパターンのユニフィケーション・アルゴリズムが構成できることを示す。この拡張における基本思想は、一般のストリングパターンを制約条件付き正規ストリングパターンとして表現するというもので、ストリング処理における制約条件プログラミングを目指したものとなっている。

# STRING PATTERN UNIFICATION

Takashi MATSUYAMA　Ru-tao ZHANG
Dept. of Information Eng.
Tohoku University
Sendai, Miyagi 980

Pattern matching is an important fundamental function in string processing. The introduction of unification will realize versatile string processing functions. This paper proposes a complete algorithm of string pattern unification, where a string pattern is defined as an ordered sequence of constant symbols, unit-length variables, and arbitrary-length variables. We first confine ourselves to the regular string pattern, which is a subclass of string patterns, and prove the completeness of the unification algorithm. Then based on this simplified algorithm, we formulate a unification algorithm for general string patterns. In this formulation, we describe a general string pattern in terms of a regular string pattern with constraints. Extending this idea, we will be able to formulate the constraint programming in string processing.

## 1．はじめに

ストリングやリスト処理における重要な演算の１つにパターンマッチングがあり、テキストエディタやデータベース検索、自然言語処理、パターンによる関数呼び出しなど様々な分野で利用されている。一方、パターンマッチングの拡張であるユニフィケーションはＰＲＯＬＯＧなどの論理型言語の基本演算機能として、その能力の高さが広く知られている。本論文では、ストリングを対象としたユニフィケーションアルゴリズムを提案し、その完全性を証明する。ストリングに対するユニフィケーションを基本演算として利用することにより、従来パターンマッチングを用いて行われていた各種演算の機能が大幅に向上するものと考えられる。

ストリングパターン（変数を含む文字列）のユニフィケーションとは、与えられた２つのストリングパターン中の変数に相互に文字列を代入することにより、両者を一致させる操作である。通常のユニフィケーションと異なり、ストリングパターンのユニフィケーションでは、任意長の文字列と照合する任意文字列変数があるため、完全なアルゴリズムを考えるのは難しい［１］。ここでは、まず制限されたストリングパターンを対象として、完全なユニフィケーションアルゴリズムを与える。その後、このアルゴリズムに基づいて一般のストリングパターンのユニフィケーションアルゴリズムを構成する。

以下では、まず２において、ストリングパターン及びそのユニフィケーションを定義し、２つのストリングパターンのＭＧＣＣＩＳ（most general complete common instance set）の概念を導入する。３では、正規ストリングパターンという制限されたストリングパターンに対するユニフィケーションアルゴリズムを与え、その完全性を証明する。４では、３のアルゴリズムを拡張することにより、一般のストリングパターンに対するユニフィケーションアルゴリズムが構成できることを示す。

## ２．ストリングパターンのユニフィケーション

### ２．１　ストリングパターン

【定義１】Σを有限アルファベット、Σｖ、Σｕをそれぞれ可算無限個の記号集合とし、Σ、Σｖ、Σｕは互いに素であるとする。Σ、Σｖ、Σｕの要素をそれぞれ定数記号、一文字列変数、任意文字列変数と呼ぶ。ストリングパターンとは、Σ∪Σｖ∪Σｕ中の要素の有限個（１個以上）の並びで、要素の個数をストリングパターンの長さと呼ぶ。特に、Σの要素だけからなるストリングパターンを定数ストリングと呼ぶ。（ストリングパターンの定義を長さ０以上の要素の並びとし、空文字列を導入した場合の議論は５で述べる。）

以下ではａ、ｂ、ｃ等の文字で定数記号を表し、＊ｘ等の＊から始まる文字列で一文字列変数を、＄ｘ等の＄から始まる文字列で任意文字列変数を表す。また、ストリングパターンは、ａ　ｂ　ｃやａ　＄ｘ　ｂのように、要素である定数記号や変数を空白で区切って表現する。

このことから明らかなように、ここで考えるストリングパターンのユニフィケーションアルゴリズムは、（ａ　ｂ　ｃ）や（ａ　＄ｘ　ｂ）等のアトムリストに対してもそのまま適用できる。より一般的には、ストリングパターンのユニフィケーションの問題は、結合則を満たす関数ｆ（ｘ，ｙ）によって作られる項－たとえば ｆ（ａ，ｆ（ｂ，ｃ））＝ｆ（ｆ（ａ，ｂ），ｃ）－に対する意味的ユニフィケーションを考える問題といえる［２］。

【定義２】代入ｆとは全てのストリングパターンの集合Ｐ上の関数で、対＜変数，ストリングパターン＞の有限集合として表現される。

ｆ＝｛＜ｖｉ，ｅｉ＞｜ｖｉ∈Σｕ∪Σｖ，ｅｉ∈Ｐ，
１≦ｉ≦ｎ｝

ここで、ｎは有限の正数、ｅｉをｖｉに対する束縛と呼び、ｆは以下の条件を満たす。

①ｉ≠ｊ⇒ｖｉ≠ｖｊ　（全てのｖｉは異なる）
②ｖｉ∈Σｖの時、ｅｉ∈Σ∪Σｖ　（一文字列変数には長さ１の定数文字列あるいは１つの一文字列変数しか束縛できない）
③ｖｉ∈Σｕの時，ｅｉ∈Ｐ　（任意文字列変数にはどのようなストリングパターンでも束縛できる）

また、次の①、②、③、④を満たすパターンｐに対する代入をｐの改名と呼ぶ。

①∀ｉ，ｅｉ∈Σｖ∪Σｕ　（全ての束縛は変数である）
②ｉ≠ｊ ⇒ ｅｉ≠ｅｊ　（変数が異なれば、対応する束縛も異なる）
③ｖｉ∈Σｕ⇒ｅｉ∈Σｕ（任意文字列変数には任意文字列変数が束縛される）
④∀ｉ，ｅｉ∈Ｖｐ　Ｖｐ：ｐ中の変数の集合

ストリングパターンｐに代入ｆ＝｛＜ｖ１，ｅ１＞，…，＜ｖｎ，ｅｎ＞｝を適用した結果ｆ（ｐ）は、ｐ中の各ｖｉを同時にｅｉ（１≦ｉ≦ｎ）に書き換えて得られたストリングパターンである。 たとえば、代入

ｆ＝｛＜＊ｘ，ａ＞，＜＄ｘ，＄ｙ　＄ｚ＞，＜＄ｙ，ａ　ｂ＞｝

をストリングパターン

ｐ＝ａ　＄ｘ　＊ｘ　ｂ　＄ｙ

に対して適用すると

ｆ（ｐ）＝ａ　＄ｙ　＄ｚ　ａ　ｂ　ａ　ｂ

以下では、ストリングパターンのことを単にパターンと略記する。

【定義３】パターンｐの規定するΣ上の言語Ｌ（ｐ）とは、パターンｐ中の各変数を任意の定数ストリングに書き換えることによって得られる定数ストリングの集合である。ここでの書き換えには次の２種類がある。

①一文字列変数を長さ１の定数ストリングに書き換える。
②任意文字列変数を長さ１以上の定数ストリングに書き換える。

パターン言語と文脈自由言語とは互いに包含関係が成り立たない。たとえば、パターン　＄ｘ　ａ　＄ｘ　によって規定される言語は、｛中央がａで、ａの左右のストリングが等しい定数ストリング｝で、文脈自由文法では表せない。また、正規文法　Ａ－－＞ａ　｜　ａ　Ａ　が表す言語は｛$a^n$｜１≦ｎ｝で、１つのパターンでは表現できない。

【定義４】パターンの間の等価関係と包含関係

①等価関係≡

ｐ１≡ｐ２　iff　∃ｆ　ｐ１＝ｆ（ｐ２）　ｆはｐ２の改名

②包含関係≦

ｐ１≦ｐ２ iff　∃ｆ　ｐ１＝ｆ（ｐ２）　ｆは１つの代入

明らかに、ｐ１≡ｐ２　は　Ｌ（ｐ１）＝Ｌ（ｐ２）を意味し、ｐ１≦ｐ２　であれば　Ｌ（ｐ１）⊆Ｌ（ｐ２）。また、≦は推移律を満たす。

パターンには次のようなクラスがある［３］［４］。

①正規パターン：全ての変数が高々１回しか現れないパターン

②一変数パターン：変数が１種類しかないパターン
③非交差パターン：任意の変数xの最左出現と最右出現の間に他の変数がないパターン
たとえば、
①p1＝a　$x　b　$y　c　$z　d
は正規パターン
②p2＝a　$x　b　$x　c　$x　d
は一変数パターン
③p3＝a　$x　b　$x　c　$z　d
は非交差パターンである。
　Pr、Po、Pn をそれぞれ正規パターン、一変数パターン、非交差パターンの集合とすると、明らかに、Pr⊆Pn、Po⊆Pn。しかし、パターンが表す言語という観点からは、正規パターンが最も一般的である。つまり、一般のパターン（一変数パターン、非交差パターン等）によって規定される言語は、ある正規パターンによって規定される言語の部分集合となる。たとえば、上のp1、p2、p3 に対して、L（p2）⊆L（p3）⊆L（p1）。このことは、４において一般のパターンのユニフィケーションを考えるときに重要な役割を果たす。

## ２．２　ストリングパターンのユニフィケーション

【定義５】２つのパターンp1とp2の共通インスタンスとは、次の①と②を満たすようなパターンp'である。
①p'≦p1　　　　②p'≦p2
【定義６】パターンのユニフィケーションとは、２つのパターンp1とp2が与えられたとき、それぞれに代入を施ことにより、与えられた２つのパターンを同一化する操作である。すなわち、パターンp1とp2に対して
p＝f1（p1）＝f2（p2）　………(1)
を満たすような代入f1、f2および両者の共通インスタンスpを求める操作である。
　一般にp1とp2には、a　$x　bとa　$xのように共通する変数が含まれているが、以下ではそのような共通変数がない場合のユニフィケーションをまず考える。このことは、p1とp2のユニフィケーションを、２つのパターンによってそれぞれ独立に規定された言語L（p1）とL（p2）の積集合を求める操作と考えることを意味する。こう仮定することにより、(1)の代入f1、f2の間に共通変数がなくなり、f＝f1∪f2とすると、明らかにfは１つの代入で、
p＝f（p1）＝f（p2）　………(1)'
この代入fをp1とp2のユニファイアと呼ぶ。（p1とp2に共通変数が含まれる場合のユニフィケーションは４で述べる。）
【定義７】２つのパターンp1とp2の MGCI（most general common instance ）とは、
　　L（p）＝L（p1）∩L（p2）
を満たすようなパターンpである。このとき、ユニファイア　f　（f（p1）＝f（p2）＝p）をp1とp2のMGU（most general unifier）と呼ぶ。
【定理１】（most general common instance の非存在性）一般に、２つのパターンの全ての共通インスタンスの集合は１つのパターンで表すことができない。
【証明】　２つのパターン
p1＝　$x　a　　　p2＝　a　$y
の共通インスタンスであるa　aとa　$z　aは１つのパターンでは表現できない（$zにいかなる空でないパターンを代入してもa　aにはならない）。■
（注意：任意文字列変数に空文字列の束縛を許した場合でも、p1とp2の共通インスタンスaとa　$z　aは１つのパターンで表現できない。）
　この定理から、いくつかのパターンの集合によって、２つのパターンの全ての共通インスタンスの集合（common instance set）を表すことが考えられる。すなわち、２つのパターンp1とp2の共通インスタンスの集合を次の条件を満たすk個のパターン{p'1, ……, p'k}によって表現することを考える。（一般にkは有限とは限らない）
L（p1）∩L（p2）＝L（p'1）∪…L（p'k）
【定義８】２つのパターンp1とp2の完全共通インスタイス集合CCIS（Complete Common Instnce Set）Σ（p1, p2）とは、p1とp2の全ての共通インスタンスからなる集合である。すなわち
Σ（p1, p2）＝{p'｜p'≦p1 and p'≦p2}
　明らかに、Σ（p1, p2）はp1とp2のユニフィケーションの結果を表すが、Σ（p1, p2）中の要素の間に等価あるいは包含関係が成り立つかもしれない。したがって、より簡潔な表現である MGCCIS（most general complete common instance set）を考える。
【定義９】２つのパターンp1とp2の most general complete common instance set（以下MGCCISと略記する）μΣ（p1, p2）とは、次の①, ②, ③を同時に満たすようなパターンの集合である。
①μΣ(p1, p2)⊆Σ(p1, p2)　　(consistency)
②∀p'∈Σ（p1, p2）に対し、
∃p∈μΣ（p1, p2）　p'≦p　　(completeness)
③∀p, p'∈μΣ（p1, p2）に対し、
p≰p'（p≦p'ではない）　　(non-redundancy)
　明らかに、μΣ（p1, p2）とΣ（p1, p2）の表している言語（要素である各パターンの表現している言語の和）は同じである。従って、μΣ（p1, p2）によってp1とp2のユニフィケーションの結果を表すことにすれば、完全で冗長性のない記述が得られる。
　以下では、正規パターンに対象を限定し、MGCCISの性質と、ユニフィケーションアルゴリズムを述べる。

## ３．正規パターンのユニフィケーションアルゴリズム

### ３．１　正規パターンのMGCCISの性質

【定理２】正規パターンp1とp2のμΣ（p1, p2）は正規パターンだけからなる集合である。
【証明】正規パターンでないp∈μΣ（p1, p2）が存在すると仮定する。すなわち、pには
p＝　α　$x　β　$x　γ
のような任意文字列変数$x（一文字列変数の場合も同じ）が含まれるとする。ここで，α、β、γはそれぞれパターンである。
　μΣ（p1, p2）の定義およびp1、p2に共通変数がないことより、
∃f1, f2　f1（p1）＝f2（p2）＝p
　　　　　　＝　α　$x　β　$x　γ
f1、f2における$xを含む束縛の種類によって次の４つの場合に分けて考える（もし、元のp1（あるいはp2）に$xが（ただ１個）存在する場合は、代入f1（あるいはf2）に対<$x, $x>があると考える。）
①f1＝f11∪{<$i1, αe　$x　βb>,

＜＄i2，βe　＄x　γb＞}
f 2＝ f 21∪ {＜＄j1，αe'　＄x　βb'＞，
＜＄j2，βe'　＄x　γb'＞} のとき。

ここで、＄i1、＄i2はｐ1中の変数、＄j1、＄j2はｐ2中の変数を表す（図1(a)）。すなわち、ｐに含まれる前の＄ｘと後ろの＄ｘがそれぞれ異なった変数＄i1（＄j1）、＄i2（＄j2）に対する束縛によって生成されたものである場合。ここで

α＝αb　αe＝αb'　αe'
β＝βb　βm　βe＝βb'　βm'　βe'
γ＝γb　γe＝γb'　γe'

とする。このとき、これまでに現れていない新たな変数＄ｘ'を用いて、新たな代入

f '1＝ f 11∪ {＜＄i1，αe　＄x　βb＞，
＜＄i2，βe　＄x'　γb＞}
f '2＝ f 21∪ {＜＄j1，αe'　＄x　βb'＞，
＜＄j2，βe'　＄x'　γb'＞}

を考えると、

ｐ'1＝ f '1（ｐ1）＝αb　αe　＄x　βb　βm
βe　＄x'　γb　γe
＝α　＄x　β　＄x'　γ
ｐ'2＝ f '2（ｐ2）＝αb'　αe'　＄x　βb'　βm'
βe'　＄x'　γb'　γe'
＝α　＄x　β　＄x'　γ

明らかにｐ'1＝ｐ'2＝ｐ'　かつ　ｐ＝ｆ（ｐ'），
ｆ＝{＜＄x'，＄x＞} となるｐ'が存在する。

② f 1＝ f 11∪ {＜＄i，αe ＄x β ＄x γb＞}
f 2＝ f 21∪ {＜＄j1，αe'　＄x　βb'＞，
＜＄j2，βe'　＄x　γb'＞}

のとき。ここで、＄iはｐ1中の変数で、＄j1と＄j2はｐ2中の変数である（図1(b)）。すなわち，ｐに含まれる2つの＄ｘは、ｆ1では＄iに対する束縛によって生成され、ｆ2では＄j1と＄j2に対する束縛によって生成されたものである場合。ここで

α＝αb　αe＝αb'　αe'
β＝βb'　βm'　βe'
γ＝γb　γe＝γb'　γe'

とする。このとき、これまでに現れていない新たな変数＄ｘ'を用いて、新たな代入

f '1＝ f 11∪ {＜＄i，αe ＄x β ＄x' γb＞}
f '2＝ f 21∪ {＜＄j1，αe'　＄x　βb'＞，
＜＄j2，βe'　＄x'　γb'＞}

を考えると、

ｐ'1＝f '1(ｐ1)＝αb αe ＄x β ＄x' γb γe
＝α　＄x　β　＄x'　γ
ｐ'2＝f '2(ｐ2)＝αb'　αe'　＄x　βb'　βm'　βe'
＄x'　γb'　γe'
＝α　＄x　β　＄x'　γ

明らかに、ｐ'1＝ｐ'2＝ｐ'　かつ　ｐ＝ｆ（ｐ'）、
ｆ＝{＜＄x'，＄x＞} となるｐ'が存在する。

③ f 1＝ f 11∪ {＜＄i1，αe　＄x　βb＞，
＜＄i2，βe　＄x　γb＞}
f 2＝ f 21∪ {＜＄j，αe' ＄x β ＄x γb'＞}

④ f 1＝ f 11∪ {＜＄i，αe ＄x β ＄x γb＞}
f 2＝ f 21∪ {＜＄j，αe' ＄x β ＄x γb'＞}

の場合も同じように、ｐ＝ｆ（ｐ'）かつｆ'1（ｐ1）＝ｆ'2（ｐ2）＝ｐ'を満たすようなｐ'が求められる。

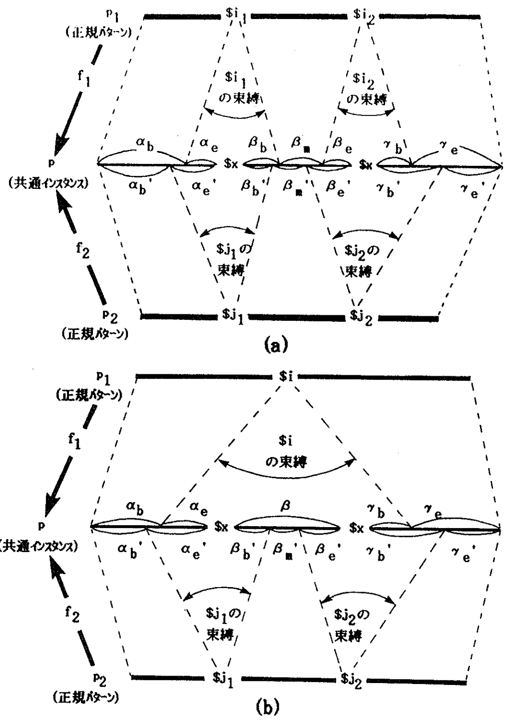


図1　正規パターン $p_1$ と $p_2$ の 共通インスタンス

明らかに　ｐ'∈Σ（ｐ1，ｐ2）であり、
（ａ）ｐ'∉μΣ（ｐ1，ｐ2）ならば、ｐ≦ｐ'（ｐ＝ｆ（ｐ'）から）なので、定義9の条件③によって　ｐ∉μΣ（ｐ1，ｐ2）
（ｂ）ｐ'∈μΣ（ｐ1，ｐ2）ならば、定義9の②より、∃ｐ"∈μΣ（ｐ1，ｐ2）かつｐ'≦ｐ"。また、ｐ≦ｐ'および≦は推移律を満たすことからｐ≦ｐ"となり、ｐ∉μΣ（ｐ1，ｐ2）
（ａ）、（ｂ）いずれの場合でもｐ∉μΣ（ｐ1，ｐ2）になる。これは最初の仮定に矛盾する。■

以上のことから、正規パターンｐ1とｐ2のμΣ（ｐ1，ｐ2）は正規パターンの集合であることが証明された。

以下では正規パターンの完全なユニフィケーションアルゴリズムについて述べる。ここでの完全性とは、（共通変数のない）2つの正規パターンｐ1とｐ2に対して、有限時間で、μΣ（ｐ1，ｐ2）が求められることをいう。

### 3．2　アルゴリズムR

2つの正規パターンｐ1とｐ2があり、それぞれの長さをｍ、ｎとする。ｐ1iをｐ1のｉ番目の要素、ｐ2jをｐ2のｊ番目の要素とする。（1≦ｉ≦ｍ，1≦ｊ≦ｎ）

①ｍ＊ｎの大きさの配列Ａを作る。

```
②for　ｉ＝1　to　ｍ　do
    if　ｐ1iが任意文字列変数
    then　Ａ（ｉ，1），…，Ａ（ｉ，ｎ）に印を付け、
          Ａ（ｉ，1）からＡ（ｉ，2）へ
          ……
          Ａ（ｉ，ｎ-1）からＡ（ｉ，ｎ）へ
          リンクを付ける。
```

elseif　p1iが一文字列変数
then　A（i,1），……，A（i,n）に印を付ける（図2(a)）
③for　j＝1　to　n　do
if　p2jが任意文字列変数
then　A（1，j），…，A（m，j）に印を付け、
A（1，j）からA（2，j）へ
……
A（m-1，j）からA（m，j）へ
リンクを付ける。
elseif　p2jが一文字列変数
then　A（1，j），……，A（m，j）に印を付ける（図2(b)）
④for　i＝1　to　m　do
for　j＝1　to　n　do
if　p1i、p2jが共に定数記号かつp1i=p2j
then　A（i，j）に印を付ける（図2(c)）
⑤for　i＝1　to　m　do
for　j＝1　to　n　do
if　A（i，j）とA（i+1，j+1）に印が存在する
then　A（i，j）からA（i+1，j+1）へリンクを付ける。（図2(d)）
⑥A（1，1）からA（m，n）までの全ての道（連続する有向のリンクの並び）を求める。

以上のアルゴリズムにおいて、各道は2つの正規パターンの共通インスタンスと対応する。もし1つの道もなければ、2つのパターンのユニフィケーションは失敗となる。このアルゴリズムによる2つの正規パターン

p1＝＄x1　b　c　＄y1　f　g　h
p2＝＄x　a　b　＄y　e　f　＄z

のユニフィケーションの例を図3に示す。

【道が表す共通インスタンスの計算法】
道の上の全ての点A（i，j）に対して
①if　p1i、p2jのいずれかが定数記号
then　その定数を採る。
②elseif　p1i、p2jのいずれかが一文字列変数
then　新たな一文字列変数（＊ij）を生成する。
③else　新たな任意文字列変数（＄ij）を生成する。

道に対応する共通インスタンスは、上の①②③で得られた記号をA（1，1）からA（m，n）まで道を辿った順に並べたものである。上の②と③で生成された一文字列変数、任意文字列変数は全て互いに異なり、p1あるいはp2に含まれないものとする。

【道が表すユニファイアの計算法】
①任意文字列変数の束縛
p1iを任意文字列変数とする（p2jが任意文字列変数の場合も同様）。道の上に必ずA（i,j1）からA（i,j2）までの連続した水平リンクの並びがある（極端な場合j1＝j2）。その道が表す共通インスタンスの中で、A（i,j1）からA（i,j2）までに対応する部分パターンをp1iの束縛とする。
②一文字列変数の束縛
p1iを一文字列変数とする（p2jが一文字列変数である場合も同様）。道の上にA（i,j）があり、その道の表す共通インスタンスにおいて、A（i,j）に対応する部分パターン（長さ1）をp1iの束縛とする。

上の①と②で求めた変数の束縛の集合がp1とp2のユニファイアである。

アルゴリズムRで求めた道iに対応する共通インスタ

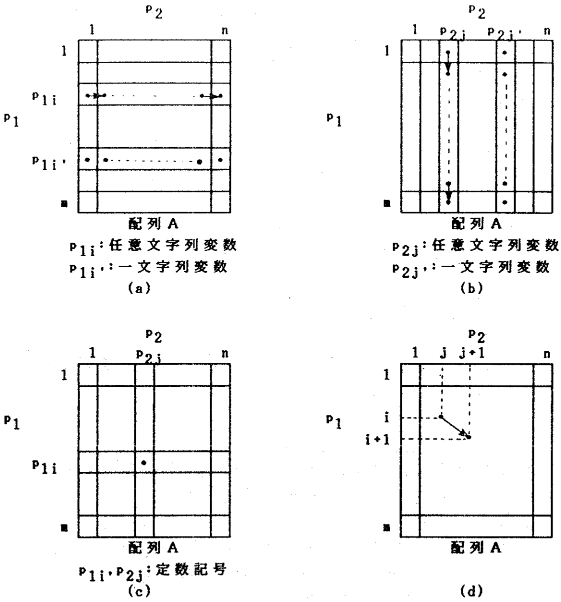


図2　アルゴリズムR（本文参照）

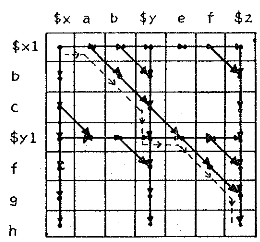


図3　ユニフィケーションの例

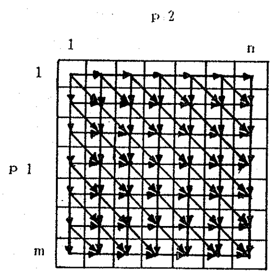


図4　任意文字列変数のみからなるパターンのユニフィケーション

ンス、ユニファイアをri、fiとすると、明らかに
fi（p1）＝fi（p2）＝ri ……(2)
たとえば、図3中の破線で示した道に対応する共通インスタンスとユニファイアはそれぞれ
r=$1 a b c $2 e f g h
f＝{＜$x, $1＞，＜$y, c $2＞，＜$z, g h＞，
＜$x1, $1 a＞，＜$y1, $2 e＞}
となる。

### 3．3　アルゴリズムRの計算量

アルゴリズムRには6つのステップがあり、最も計算量が多いのはステップ6の道の探索過程である。このステップの計算量は道の数に依存する。また、道の数はp1、p2に含まれる任意文字列変数の数に依存する。最悪の場合は、p1もp2も任意文字列変数のみからなるときで、図4に示すように、
N(i,j)をA（1,1）からA（i,j）までの道の数とすると、
N（i,1）＝1（1≦i≦m）　………(3.1)
N（1,j）＝1（1≦j≦n）　………(3.2)

$$N(i,j) = N(i-1,j) + N(i,j-1) + N(i-1,j-1)$$
$$(1 < i \leqq m,\ 1 < j \leqq n) \quad \cdots (3.3)$$

が成り立つ。(3.1)、(3.2)、(3.3)を解くと、

$$N(m,n) = \frac{(m+n)!}{m!n!}$$

また、

$$\frac{(m+n)!}{m!n!} \leqq \frac{(m+n)!}{((m+n)/2)!\ ((m+n)/2)!}$$

$$\fallingdotseq \frac{(m+n)^{m+n}}{((m+n)/2)^{(m+n)/2}\ ((m+n)/2)^{(m+n)/2}}$$

$$= \frac{(m+n)^{m+n}}{((m+n)/2)^{m+n}} \fallingdotseq 2^{m+n}$$

従って、ステップ6(アルゴリズムR)の計算量の最悪値は$O(2^{m+n})$となる。

## 3.4 MGCCISの計算

アルゴリズムRから、$\mu\Sigma$(p1, p2)が求められることは次の定理3、4によって証明される。

【定理3】正規パターンp1とp2に対して、アルゴリズムRで求められた共通インスタンスの集合を{r1, r2, …, rK}とし、p'をp1とp2の任意の共通インスタンスとすると、∃k, 1≦k≦K　p'≦rk …………　(4)

【証明】 p1とp2の任意の共通インスタンスp'が、アルゴリズムRで求めた配列A中の1つの道と対応付けられることを証明すればよい。

p1、p2の長さをそれぞれm、nとする。

まず、p'はp1とp2の共通インスタンスなので、
∃f1, f2　p'=f1(p1)=f2(p2)
ここでf1とf2は代入である。このとき、p'中のpt'…p't+x(0≦x)という部分パターンに対して、p1、p2中の変数の束縛の重なり方を考えると、次の4つの場合がある(図5(a))

①p't…p't+xがp1、p2中の定数ストリングに対応する場合。すなわち、
∃i, j　p1i…p1i+x=p2j…p2j+x=p't…p't+x
かつ　p't, …, p't+xが全て定数記号である。このとき、配列AにはA(i, j)からA(i+x, j+x)への連続する斜めリンクの並びがある。

②p't……p't+xがp1中の定数ストリング、p2中の変数と対応する(変数が一文字列変数の時はx=0)場合。すなわち、
∃i, j　p't……p't+x=p1j……p1i+x,
　　<p2j, …p't…p't+x…>∈f2
かつ　p't, ……, p't+xが全て定数記号である。このとき、配列AにはA(i, j)からA(i+x, j)への連続する縦リンクがある。

③p't……p't+xがp2の定数ストリング、p1の変数と対応する場合。これは②と同様。

④p't……p't+xがp1とp2の変数と対応する場合。すなわち、
∃i, j　<p1i, ……p't……p't+x……>∈f1
　　<p2j, ……p't……p't+x……>∈f2
このとき、配列A中のA(i, j)に印が付けられる。これは、p1j、p21がそれぞれ任意文字列変数あるいは一文字列変数であるからである。

次に、共通インスタンスp'上において、上で考えた4つの場合の部分パターンの並び方を考える。

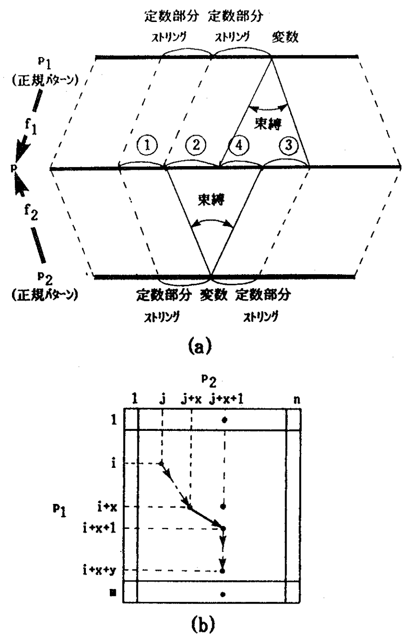


図5　定理3の証明図

①の後ろに②が続く場合。すなわち、図5(b)に示すように、p1中の p1i …p1i+x、p2中のp2j…p2j+x がp'中の定数部分パターンp't…p't+x と対応し、p1中のp1i+x+1…p1i+x+y, p2中の変数p2j+x+1 がp't+x+1…p't+x+yと対応する。p2j+x+1は変数なので、A(1, j+x+1), …, A(m, j+x+1)には印が付けられており、A(i+x, j+x)からA(i+x+1, j+x+1)へのリンクが付けられる。すると、①と②の場合に求まる2つの部分連続リンクが1つの連続リンクになり、この連続リンクがp'中のp't……p't+x+yと対応する。

同様に、①②③④の任意の2つの場合の組み合わせに対して、それぞれの対応する部分連続リンクの間にリンクが付けられ、1つの連続リンクとなり、最後に、p'全体がA中の1つの道と対応する。また、①②③④のどの場合が一番初め、あるいは一番終りにあってもA(1, 1)とA(m, n)は道に入る。

以上のことから、p1とp2の任意の共通インスタンスを配列A中の道に対応付けることができ、その道が表す共通インスタンスをrkとするとp'=f(rk)が成り立つ。ここで、fは{<$ij, p't……p't+x>}と表せ、$ijは上の④の場合におけるp1i、p2jに対応する変数、p't……p't+xはそのときのp'中の部分パターンである。■

この定理から、アルゴリズムRは
①全ての解が求められる。
②有限時間で止まる。(配列Aが有限であるから)
また、このアルゴリズムで求めた全てのrkは
rk∈Σ(p1, p2)　……………　(5)
であるから、{r1, r2, ……, rk}はMGCCISの条件の①と②を満たす(式(4)、式(5)より)。したがって、R={r1, r2, ……, rk}からri≦rjを満たすようなriを除去すれば、残ったものが$\mu\Sigma$(p1, p2)となる。

以下ではこのようなｒｉの除去法について述べる。
【定理４】ｐ1≦ｐ2 iff μΣ（ｐ1，ｐ2）＝{ｐ1}
【証明】⇒　ｐ1≦ｐ2　と仮定する。ｐ1≦ｐ1なので、ｐ1∈Σ（ｐ1，ｐ2）。また、Σ（ｐ1，ｐ2）の定義より、∀ｐ'∈Σ（ｐ1，ｐ2）　ｐ'≦ｐ1
したがって、　μΣ（ｐ1，ｐ2）＝{ｐ1}
⇐　μΣ（ｐ1，ｐ2）＝{ｐ1}と仮定すると、μΣ（ｐ1，ｐ2）の定義より、ｐ1≦ｐ2。■

定理３、４より、ｐ1≦ｐ2　ならば改名を除き　ｐ1∈Ｒ＝{ｒ1，……，ｒｋ}となる。このことから、２つのパターンｐ1、ｐ2に対してアルゴリズムＲを適用して得られたＲ＝{ｒ1，……，ｒｋ}に含まれる任意のｒｉ、ｒｊに対して再びアルゴリズムＲを適用し、その結果をＲ'とすると、（改名を除き）ｒｉ∈Ｒ'（あるいはｒｊ∈Ｒ'）であれば、ｒｉ≦ｒｊ（あるいはｒｊ≦ｒｉ）となり、Ｒからｒｉ（あるいはｒｊ）を除去する。（ｒｉ∈Ｒ'の判定はｒｉ、ｒｊのユニフィケーションに用いた配列中の道の形から容易に判断できる）。このようにして、Ｒからｒｉ≦ｒｊを満たすｒｉを除去することによりμΣ（ｐ1，ｐ2）が求められる。

以上のことから、アルゴリズムＲによって、正規パターンｐ1、ｐ2 のＭＧＣＣＩＳ　μΣ（ｐ1，ｐ2）が有限時間で計算できることが示された。

## ３．５　ＭＧＣＵＳの計算

ここでは、２つの正規パターンの most general complete unifier set を定義し、その求め方を示す。

【定義１０】２つの代入ｆ1とｆ2があるとする。
ｆ1＝{＜ｖ1ｉ，ｅ1ｉ＞｜１≦ｉ≦ｍ}
ｆ2＝{＜ｖ2ｊ，ｅ2ｊ＞｜１≦ｊ≦ｎ}
このとき、ｆ1とｆ2の合成ｆ1∘ｆ2とは，
{＜ｖ2ｊ，ｆ1（ｅ2ｊ）＞｜１≦ｊ≦ｎ}∪
{＜ｖ1ｉ，ｅ1ｉ＞｜１≦ｉ≦ｎ}
から次の①あるいは②を満たすような対を除いたものである。
①＜ｘ，ｘ＞のような対（ｘは変数）
②∃1≦ｊ≦ｎ　ｖ2ｊ＝ｖ1ｉ　を満たす対＜ｖ1ｉ，ｅ1ｉ＞
代入ｆをｆ＝ｆ1∘ｆ2，ｐをパターンとしたとき、
ｆ(ｐ)＝ｆ1(ｆ2(ｐ))≦ｆ2(ｐ)
が成り立つ。

【定義１１】２つのパターンｐ1とｐ2の完全ユニファイア集合、ＵΣ（ｐ1，ｐ2）を
ＵΣ（ｐ1，ｐ2）＝{ｆ｜ｆ（ｐ1）＝ｆ（ｐ2）}
とする。

明らかに、２つのパターンの完全ユニファイア集合の要素は無限個ある可能性がある。また、完全ユニファイア集合には幾つかの冗長なものが含まれるかもしれない。したがって、完全ユニファイア集合より簡潔なＭＧＣＵＳ（most general complete unifier set）を考える。

【定義１２】２つのパターンｐ1とｐ2の Most General Complete Unifier Set（以下ではＭＧＣＵＳと略記する）μＵΣ（ｐ1，ｐ2）は同時に次の①、②、③を満たすようなものである。
①μＵΣ（ｐ1，ｐ2）⊆ＵΣ（ｐ1，ｐ2）
②∀ｆ∈ＵΣ（ｐ1，ｐ2）に対して
∃ｆ'∈μＵΣ（ｐ1，ｐ2）　∃θ　θ∘ｆ'＝ｆ
③∀ｆ1，ｆ2　∈μＵΣ（ｐ1，ｐ2）、ｆ1≒ｆ2に対して
∀θ　θ∘ｆ1≒ｆ2　ｆ1≒θ∘ｆ2

一般のパターンに対して、μＵΣ（ｐ1，ｐ2）が有限かどうかは分からないが、次の定理より正規パターンに対しては、μＵΣ（ｐ1，ｐ2）は有限で、かつアルゴリズムＲで求めたユニファイア集合はＭＧＣＵＳであることが示される。

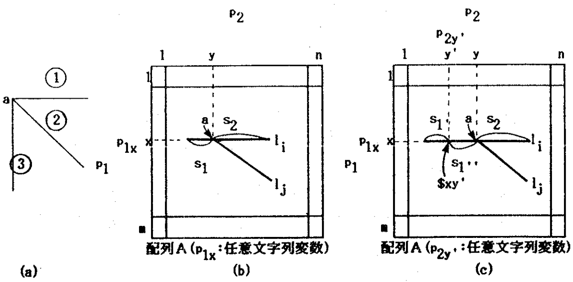


図６　定理５の証明図

【定理５】アルゴリズムＲで求めたユニファイアの集合はｐ1とｐ2のＭＧＣＵＳ　μＵΣ（ｐ1，ｐ2）である。
【証明】２つの正規パターンｐ1とｐ2に対して、アルゴリズムＲで求めたユニファイアの集合をＦ＝{ｆ1，ｆ2，……，ｆｋ}とする。ｆｉ（ｐ1）＝ｆｉ（ｐ2）（１≦ｉ≦ｋ）から、Ｆ⊆ＵΣ（ｐ1，ｐ2）。　また、定理３および代入の合成の定義より∀ｆ'∈ＵΣ（ｐ1，ｐ2）に対して、∃θ　∃ｉ，　θ∘ｆｉ＝ｆ'　。

次に ∀1≦ｉ,ｊ≦ｋ　ｉ≒ｊ ⇒ ∀θ　ｆｉ≒θ∘ｆｊ（あるいはｆｊ≒θ∘ｆｉ）であることを証明する。

ｉ≒ｊなので、アルゴリズムＲで求めたｆｉとｆｊに対応する配列Ａ中の道も異なる。その２つの道をそれぞれｌｉ、ｌｊとする。ｌｉ、ｌｊの作り方から、ｌｉとｌｊは必ずどこかで交わる。この分岐点からの２つの道の出方は図６(a)の①と②、または①と③、②と③の３つの可能性がある。

（ａ）分岐点からの２つの道の出方が①と②である場合（図６(b)）
分岐点ａの位置をＡ（ｘ，ｙ）とすると、パターンｐ1中のｘ番目の記号ｐ1ｘは任意文字列変数である。ａを含むｌｉとｌｊに共通する道のうち、ａおよびａの前の水平部分に対応する共通インスタンスの部分パターンをＳ1、ａの後ろの水平部分に対応する部分パターンをＳ2とする。
（ａ．1）分岐点ａから水平へ行く道をｌｉ，斜めへ行く道をｌｊとする。明らかに、
＜ｐ1ｘ，Ｓ1＞∈ｆｊかつ＜ｐ1ｘ，Ｓ1Ｓ2＞∈ｆｉ。
ｆｉ＝θ∘ｆｊが成り立つためには、θは次の等式を満たさなければならない。
θ（Ｓ1）＝Ｓ1Ｓ2
①Ｓ1に任意文字列変数が含まれていなければ
θ（Ｓ1）≒Ｓ1Ｓ2
となり、ｆｉ＝θ∘ｆｊ　を満たすようなθは存在しない（両辺のパターンの表すストリングの長さが異なる）。
②Ｓ1に任意文字列変数が含まれているとき（この場合、ｐ2中のＳ1と対応する部分にも任意文字列変数がある図６(c)）、θ（Ｓ1）＝Ｓ1Ｓ2 が成り立つためには、Ｓ1中の任意文字列変数に別のパターンを代入しなければならない。今、Ｓ1中の任意文字列変数＄xy'にパターンＳxy'を代入することを考える。すなわち、
＜＄xy'，Ｓxy'＞∈θ　かつ　＄xy'≒Ｓxy'
一方、ｐ2中の＄xy'と対応する任意文字列変数をｐ2y'とすると、ｆｉには＜ｐ2y'，＄xy'＞が含まれ、ｆｊにも

＜ｐ2y'，＄xy'＞が含まれる。すると、
＜ｐ2y'，Sxy'＞　∈　θ∘ｆｊ
＜ｐ2y'，＄xy'＞　∈　ｆｉ
となり、　θ∘ｆｊ≠ｆｉ　。
したがってｆｉ＝θ∘ｆｊを満たすようなθは存在しない。
（ａ.2）分岐点から水平へ行く道をｌｊ、斜めへ行く道をｌｉとする。明らかに、
＜ｐ1x，S1＞∈ｆｉかつ＜ｐ1x，S1S2＞∈ｆｊ
ｆｉ＝θ∘ｆｊ　が成り立つためには、θは次の等式を満たさなければならない。

S1＝θ（S1S2）

上式を満たすθは存在しないから（両辺の長さが異なる）、ｆｉ＝θ∘ｆｊを満たすθは存在しない。

②と③、①と③の場合も同じように、θ∘ｆｊ＝ｆｉを満たすようなθは存在しないことが証明される。従って、アルゴリズムRによって求められたユニファイアの集合｛ｆ1，…，ｆk｝は、２つの正規パターンのMGCUSである。■

定理５から、MGCCISの場合と同じように、２つの正規パターンのMGCUSは有限個であり、アルゴリズムRでそれら全てが求められる。アルゴリズムRで求まる共通インスタンスの集合の中には冗長なものが含まれており、それを除去したものがμΣ（ｐ1，ｐ2）となっていたのに対し、アルゴリズムRで求めたユニファイアの集合はそのままμUΣ（ｐ1，ｐ2）となる。

## ４．一般パターンのユニフィケーション

アルゴリズムRでは、２つのパターンｐ1、ｐ2に次の制限を設けていた。
①ｐ1、ｐ2は共に正規パターンである。
②ｐ1、ｐ2には共通する変数が存在しない。
ここでは、これらの制限を除いた一般のパターンのユニフィケーションについて述べる。

正規パターンのユニフィケーションの結果（MGCCISあるいはMGCUS）は有限集合であったが、一般パターンのMGCCISとMGCUSは有限とは限らない。たとえば、２つのパターン
ｐ1＝＄ｘ　ａ　＄ｘ
ｐ2＝ａ　＄ｙ　ａ　＄ｙ　ａ
の完全共通インスタンス集合 Σ（ｐ1，ｐ2）を考えてみよう。まず、ｐ1がａを中心として左右対称であるから、ｐ2がｐ1と等しいのは、ａ　＄ｙ＝＄ｙ　ａ が成り立つ時だけである。また、ａ　＄ｙ＝＄ｙ　ａ は、＄ｙ＝ａ，ａａ，ａａａ，．．に対してのみ成り立つ。

以上のことから、ｐ1とｐ2の完全共通インスタンスΣ（ｐ1，ｐ2）は｛$a^{3+2n}$｜1≦ｎ｝となり、有限個のパターンで表せない。また、この集合がMGCCISの３つの条件を満たしているから、上の、２つのパターンのMGCCIS　μΣ（ｐ1，ｐ2）も｛$a^{3+2n}$｜1≦ｎ｝となり、無限集合となる。同様に、上の２つのパターンのMGCUSも無限集合となる。

以下では、一般パターンを正規パターンに制約条件を付けたものとして表現し、アルゴリズムRを用いて一般パターンのユニフィケーションを行うことを考える。また、MGCCIS、MGCUSについても制約条件を用いることにより、無限集合を間接的に表現するという方法を用いる。

### ４．１制約付き正規パターンによる一般パターンの表現

２で述べたように、正規パターンが表現しているパターン言語は最も一般的である。逆にいうと、正規パターンに制約を付けたものとして一般パターンを表すことができる。たとえば、パターン
ｐ1＝＄ｘ　ａ　＄ｘ　　　　　ｐ2＝＄ｘ1　ａ　＄ｘ2
に対して、明らかにｐ1≦ｐ2なので、L（ｐ1）⊆L（ｐ2）。すなわち、L（ｐ2）を生成するとき、＄ｘ1に束縛される定数文字列と＄ｘ2に束縛される定数文字列が等しい代入によって生成された定数文字列の集合がL（ｐ1）となる。このことから、正規パターンｐ2に＄ｘ1＝＝＄ｘ2という制約を付けたものとして一般パターンｐ1を表すことにする。

一般パターンを制約条件付き正規パターンに変換するには、
①パターン中の同一変数を互いに異なる新たな変数に書き換える。
②①で導入された同一変数を表す新たな変数の対を"＝＝"で結ぶ。
たとえば，パターン
＄ｘ　ａ　＄ｙ　ｂ　＄ｘ　＄ｙ　ａ　＄ｘ　は
＄ｘ1　ａ　＄ｙ1　ｂ　＄ｘ2　＄ｙ2　ａ　＄ｘ3
where ＄ｘ1==＄ｘ2 ＆ ＄ｘ2==＄ｘ3 ＆ ＄ｙ1==＄ｙ2
と変換される。ここで、whereはそれ以降が制約条件であることを示す特殊記号である。

制約条件を利用することにより、
①｛$a^n$｝＝｛ａ，ａａ，ａａａ，．．｝など、これまで１つのパターンで表現できなかった言語が、制約条件付きパターン ＄ｘ　where　＄ｘ　ａ＝＝ａ　＄ｘ　によって記述できる。
②ｐ1＝ａ　＄ｘとｐ2＝＄ｘ　ａのユニフィケーションといった、ユニファイする２つのパターンに共通変数がある場合、まずｐ1'＝ａ　＄ｘ1とｐ2'＝＄ｘ2　ａのユニフィケーションを行い、その共通インスタンスであるパターンに＄ｘ1＝＝＄ｘ2という制約条件を加えればよいことになる。

### ４．２　一般パターンのユニフィケーション

ここでは、正規パターンのユニフィケーションアルゴリズムRに基いて一般パターンのユニフィケイションアルゴリズムを構築することを考える。

一般パターンｐ1とｐ2に対応する正規パターンをそれぞれｐ'1、ｐ'2とし、ｐ1＝ｐ'1　where　C1、　ｐ2＝ｐ'2　where　C2 とする。ここでC1、C2は＝＝式のAND結合による制約条件を表す。また、ｐ1とｐ2の共通変数に対応するｐ'1、ｐ'2中の変数を＝＝で結んだ式のAND結合による制約条件をC3とする。ｐ1 とｐ2のユニフィケーションの結果をQ、ｐ'1とｐ'2のユニフィケーションの結果をQ'（Q'＝L（ｐ'1）∩L（ｐ'2））とすると、QはQ'の部分集合で、Q'に含まれるパターンの内、制約条件C1、C2、C3を同時に満たすものの集合である。

以上の考察から、一般パターンｐ1とｐ2とのユニフィケーションは次のように行う。
①ｐ1とｐ2を制約付き正規パターンに変換する。
ｐ1＝ｐ'1　where　A11＝＝B11，…，A1m＝＝B1m
ｐ2＝ｐ'2　where　A21＝＝B21，…，A2n＝＝B2n
また、ｐ1、ｐ2の共通変数に対応するｐ'1、ｐ'2中の変

数を==で結び

where A31==B31, …, A3k==B3k

とする。ここでAij、Bijはパターンを表す。

②アルゴリズムRによって，p'1とp'2のユニフィケーションを行う。このときのMGCUSを

$\mu\cup\Sigma$（p'1, p'2）={f1, f2, ……, fk}とする。

③各fi に対して、where の後の全ての==式（A11==B11, …, A1m==B1m, A21==B21, …, A2n==B2n, A31==B31, …, A3k==B3k）の両辺に代入fiを施し、制約条件の簡約化を行う。簡約化は次の手続き simplify によって行う。

【制約条件の簡約化】

まず、代入fi の下で、1つの式A==Bを簡約化する手続き symplify-1 を示す。以下において、m、nをそれぞれパターンA、Bの長さ、AiをAのi番目の要素、BjをBのj番目の要素とする（1≦i≦m，1≦j≦n）。empty は空ストリング、$variable は任意文字列変数、*variable は一文字列変数を表す。

```
simplify-1(A,B,fi)
 begin
   A=fi(A) ; m=length(A)
   B=fi(B) ; n=length(B)
 1> if A=B then return((nil,fi))
 2> elseif A is empty then return("fail")
 3> elseif B is empty then return("fail")
 4> elseif A1=B1
        then simplify-1(A2...Am,B2...Bn,fi)
 5> elseif Am=Bn
      then simplify-1(A1...Am-1,B1...Bn-1,fi)
 6> elseif A1 and B1 are constant and A1≒B1
        then return("fail")
 7> elseif Am and Bn are constant and Am≒Bn
        then return("fail")
 8> elseif A1 is a *variable and B1 is not a
       $variable
          then let f'i= {<A1,B1>}∘fi
             simplify-1(A2...Am,B2...Bn,f'i)
 9> elseif B1 is a *variable and A1 is not a
       $variable
          then let f'i= {<B1,A1>}∘fi
             simplify 1(A2...Am,B2...Bn,f'i)
 10> elseif Am is a *variable and Bn is not a
       $variable
          then let f'i={<Am,Bn>}∘fi
            simplify-1(A1...Am-1,B2...Bn-1,f'i)
 11> elseif Bn is a *variable and Am is not a
       $variable
          then let f'i={<Bn,Am>}∘fi
            simplify-1(A1...Am-1,B1...Bn-1,f'i)
 12> elseif A is a string consisting of only
           one $variable
       then if A1 is a member of B
              then return("fail")
               else let f'i={<A1,B>}∘fi
                  return((nil fi'))
 13> elseif B is a string consisting of only
           one $variable
       then if B1 is a member of A
              then return("fail")
               else let f'i={<B1,A>}∘fi
                  return((nil fi'))
 14> else return(((A,B) fi))
   end
```

次に、simplify-1を用いて、リストとして表現された==式の集合{A1==B1, …, Ar==Br}に対する、代入fiの下での簡約化アルゴリズムsimplifyを与える。

```
simplify(((A1,B1),...,(Ar,Br)),fi)
  begin
    n=1
    m=length((A1,B1),...,(Ar,Br))
    store=()
    loop1
     M=simplify-1(An,Bn,fi)
     if M="fail" then return("fail")
     else f'i=second(M)
     n=n+1
     append first(M) to store
     if fi≒f'i
     then append (An,Bn),...,(Ar,Br) to store
          simplify(store,f'i)
     elseif n<=m then go loop1
               else return((store,f'i))
end
```

【手続き simplify の停止性】

① simplify-1 はk≦min{length（fi(A)），length（fi(B)）}回の再帰適用で止まる。（1> 2> 3> 6> 7> 13> 14> の場合 simplifyはすぐに止まり、他の場合はAとBの長さを1減らす。）

② simplifyの再帰呼び出しの条件fi≠f'i はsimplify-1中で新しい束縛が生成された場合のみに成り立つ。しかし、制約条件の集合を表すリスト((A1,B1),...,(Ar,Br))に現れる変数の個数は有限であり、simplify-1におけるf'iの生成では新たな変数は導入されない。したがって、simplifyの再帰適用で変数の数は減少するのみである。このため、simplifyの再帰適用も有限回しか生じない。

【simplifyの適用結果】

①simplifyを適用した結果が"fail"であれば、fiはp1とp2のユニファイアではない。

②simplifyを適用した結果が(((A'1, B'1), ... , (A'r', B'r')), f'i)であれば、

f'i where A'1==B'1, …, A'r'==B'r'

がp1とp2のユニファイアの候補となる。ここで、ユニファイアに対する==式による制約条件の意味は制約付きパターンの場合と同じである。

③simplifyを適用した結果が(nil, f'i)であれば、f'iはp1とp2のユニファイアである。

上の②と③で得られたユニファイア（候補）からなる集合{fc1, fc2, ……, fck'}がp1とp2のユニファイア（候補）集合である。

以上の議論から、一般パターンのユニフィケーションにおけるユニファイア集合は有限集合で表せるようになる。ただし、ユニファイアにも==式を含む可能性があり、そうしたユニファイアの候補が真のユニファイアであるかどうか（全ての制約条件を満たす解が存在するかどうか）は判定できない。

【共通インスタンスの求め方】
各ユニファイア（候補）ｆci（ｆi あるいは ｆi where Ａ1＝＝Ｂ1，……，Ａr＝＝Ｂrの形式）に対応する共通インスタンスはｆci（ｐ1）あるいはｆci（ｐ2）である。ｆciが制約条件付きのユニファイアであるときは、ｆci（ｐ1）＝ｆi（ｐ1） where Ａ1＝＝Ｂ1，…，Ａr＝＝Ｂr）となり、２つのパターンの共通インスタンスにも＝＝式が含まれる可能性がある。たとえば、パターン＄ｘ ａ ＄ｘと ａ ＄ｙ ａ ＄ｙ ａのユニフィケーション結果は
①ユニファイア｛＜＄ｘ，ａ ＄ｚ＞，＜＄ｙ，＄ｚ＞ where ａ ＄ｚ＝＝＄ｚ ａ｝
②共通インスタンス
ａ ＄ｚ ａ ＄ｚ ａ where ａ ＄ｚ＝＝＄ｚ ａ
となる。

## ５．まとめ

本論文では、ストリングパターンのユニフィケーションについて検討し、正規パターンに対する完全なユニフィケーションアルゴリズムを与え、それに基づいて一般パターンのユニフィケーションアルゴリズムを構成した。その際、制約条件付き正規パターンによる一般パターンの表現と、制約条件の簡約化手続きを与えた。

制約条件を用いることにより、全ての解を求め、有限時間で停止する一般パターンのユニフィケーション・アルゴリズムが構成できたが、得られた共通インスタンスやユニファイアにも制約条件が付けられることになり、今のところ制約条件を満たす解が存在するかどうかを決定する手続きはない。この問題は、ＣＬＰ［５］などの制約条件プログラミング言語に共通した問題であり、より強力な制約条件の簡約化手続きを開発する必要がある。

本論文では、任意文字列変数に空ストリングを束縛できないとしたが、空ストリングが束縛できる変数＄＄ｘを含む拡張パターンｐ＝α ＄＄ｘ βを｛α β，α ＄ｘ β｝のように＄＄ｘが空の場合とそうでない場合を表す２つのパターンの集合として表現することにより、アルゴリズムＲで拡張パターンのユニフィケーションを行うことができる。ここでα、βはパターンを表す。また、アルゴリズムＲの⑤における斜めリンクの設定法を改良することにより任意文字列変数に空ストリングの束縛を許したユニフィケーション・アルゴリズムが容易に構成できる［６］。

本論文で述べたストリングパターンのユニフィケーションを基本演算とすることにより、ストリングを対象とした制約条件プログラミングを実現することができる。我々は、その１つの試みとしてユニフィケーションと制約条件に基づくストリング、リスト処理言語を開発した［６］。この言語については稿を改めて報告する。

最後に本研究を行うにあたり、有益なコメントと熱心な議論をして頂いた伊藤貴康教授に感謝します。また、本研究は、文部省科学研究費一般研究Ａ（60420035）の補助を受けて行われた。

［参考文献］

[1]赤間　清(1986)：文字列領域における一般化と統一化のアルゴリズム，情報処理学会知識工学と人工知能研究会資料，86-AI-45
[2]Plotkin,G.D.(1972): Building-in Equational Theories, Machine Intelligence Vol.7,pp.73-90
[3]Shinohara,T.(1982): Polynomial Time Inference of Pattern Language and its Application, Proc. of 7th IBM Symposium on Mathematical Foundation of Computer Science, pp.193-209
[4]Angluin,D.(1979): Finding Patterns Common to a Set of Strings,Proc. of 11th Annual ACM Symposium on Theory of Computing,PP.130-141
[5]Jaffar,J. and Michaylov,S.: Methodology and Implementation of a CLP System, 4th Int'n. Logic Conf.,pp.196-218
[6]張汝涛(1988)：ユニフィケーションに基づくストリング処理に関する研究、東北大学修士論文