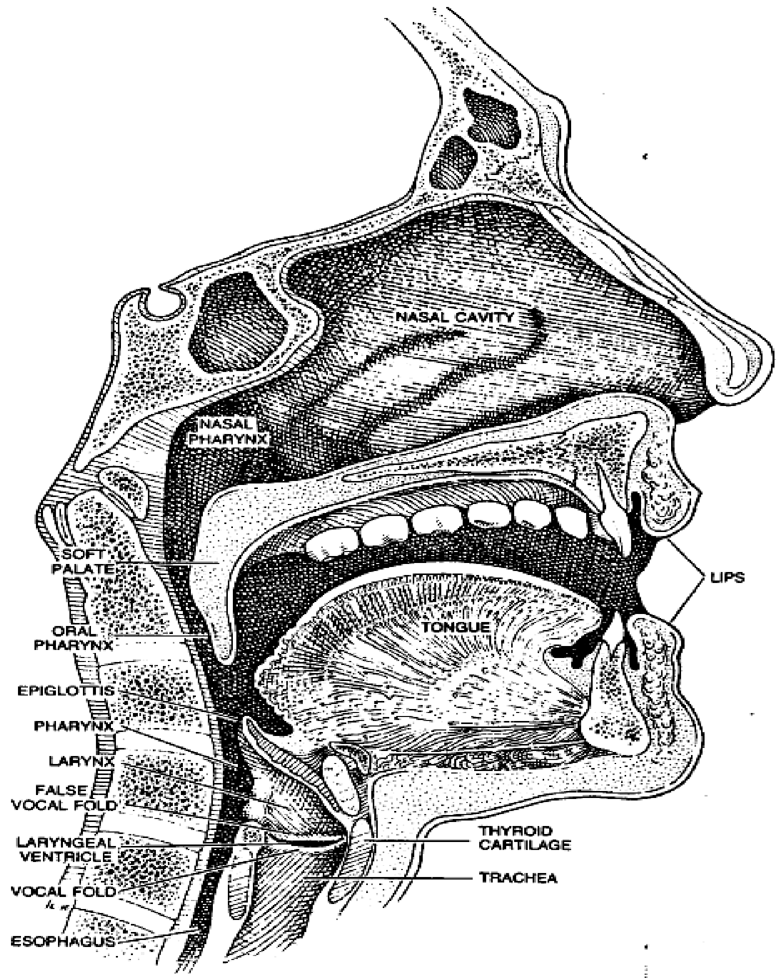
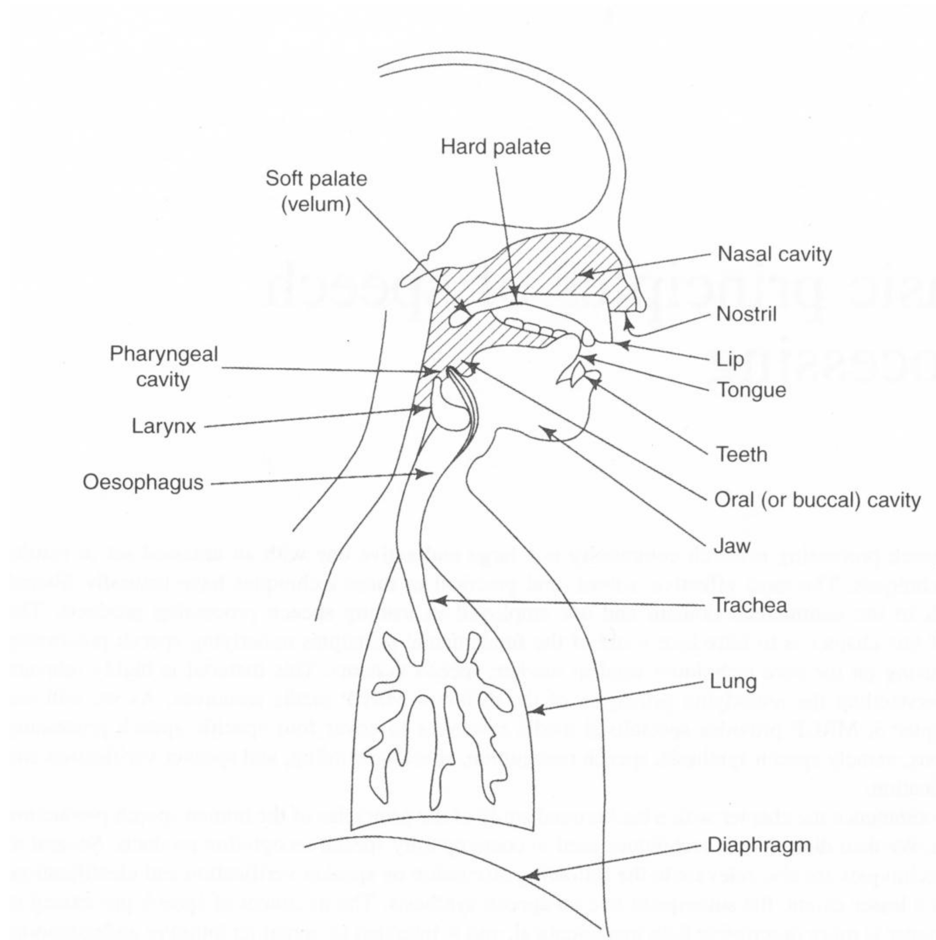
ApplyingTerm Rewriting to Speech Recognition of Numbers

R. Shostak - November, 2012

Term rewriting is an important area of the branch of artificial intelligence dealing with automated theorem proving. In this talk, we discuss its application to speech recognition grammar generation in Vocera.

The Human Speech Machine





Phonemes – the Atoms of Speech

ɪː READ	ɪ SIT	ʊ BOOK	uː TOO	ɪə HERE	eɪ DAY	John & Sarah Free Materials 1996
e MEN	ə AMERICA	ɜː WORD	ɔː SORT	ʊə TOUR	ɔɪ BOY	əʊ GO
æ CAT	ʌ BUT	ɑː PART	ɒ NOT	eə WEAR	ɑɪ MY	ɑʊ HOW

p PIG	b BED	t TIME	d DO	ʧ CHURCH	ʤ JUDGE	k KILO	g GO
f FIVE	v VERY	θ THINK	ð THE	s SIX	z ZOO	ʃ SHORT	ʒ CASUAL
m MILK	n NO	ŋ SING	h HELLO	l LIVE	r READ	w WINDOW	j YES

a sunny day

/s/ /u/ /n/ /ɪ:/

a rainy day

/r/ /eɪ/ /n/ /ɪ:/

Number of Phonemes in Different Languages

!Xóõ	112
English	44
Mandarin	28
French	24
German	24
Japanese	22
Hawaiian	13
Rotokas	11

Rotokas - World's Simplest Language

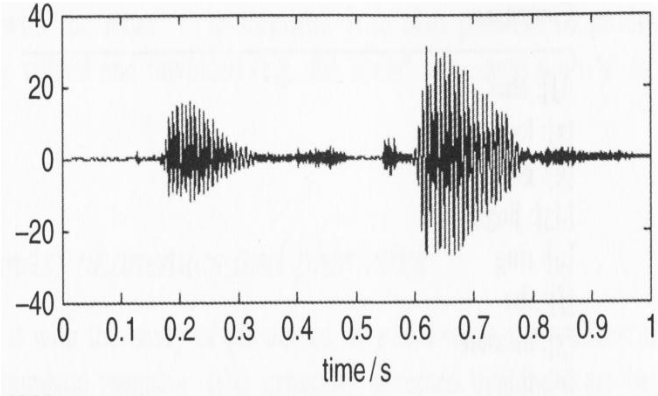
- 12 Letters
- 11 Phonemes
- 4000 Speakers

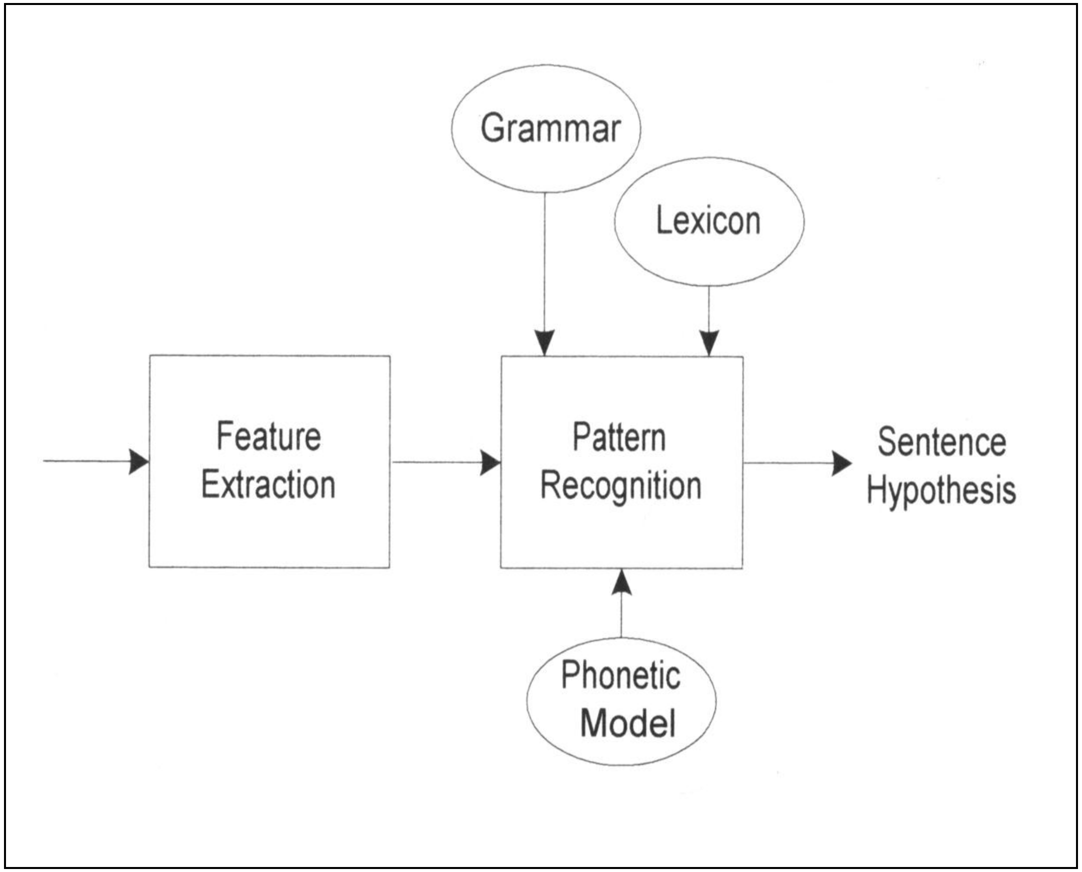
Osireitoarei avukava iava ururupavira toupasiveira.

"The old woman's eyes are shut."

Speech Recognition System

Input Waveform





Recognition Grammar

Modified BNF Format:

Weather ::= a **Condition** day

Condition ::= sunny | rainy

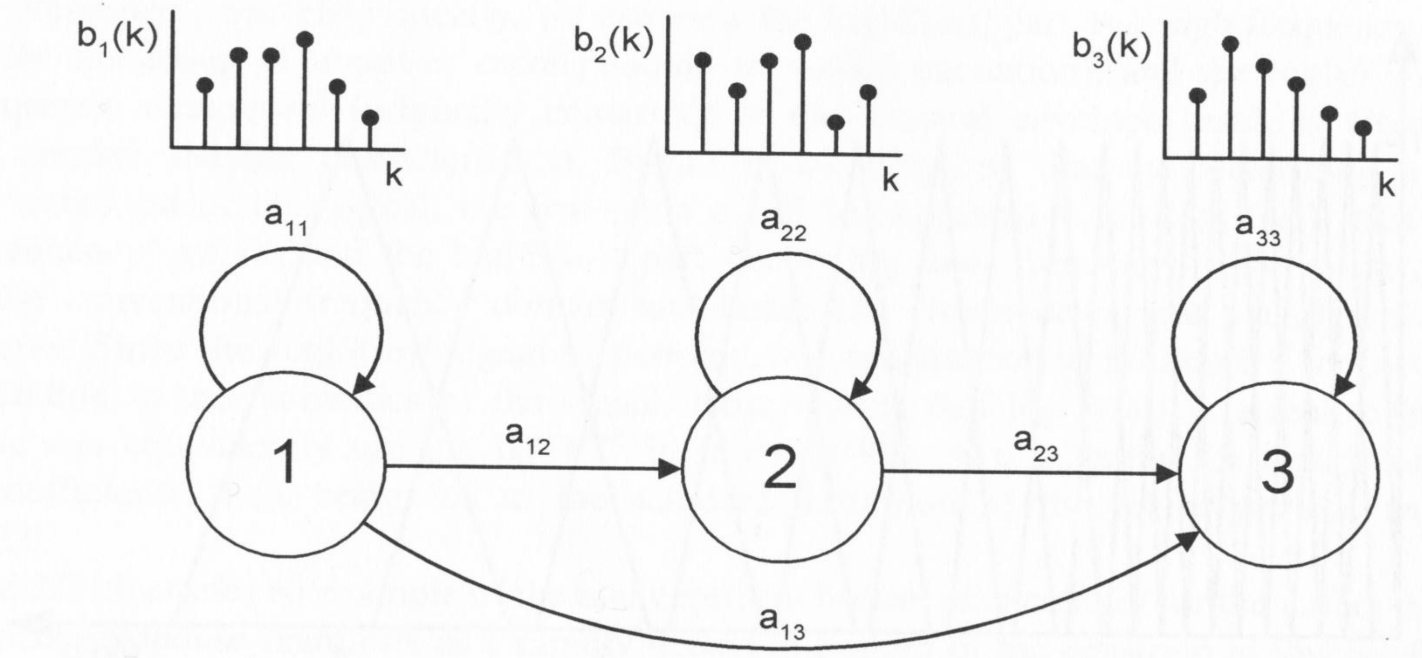
SRGS XML- Format:

```
<rule id=weather>
   a
   <one-of>
     <item> sunny </item>
     <item> rainy  </item>
   </one-of>
    day
</rule>
```

Lexicon (Dictionary)

a	/Λ/
day	/d/
rainy	/r/ /eɪ/ /n/ /ɪ:/
sunny	/s/ /u/ /n/ /ɪ:/
this	/ ð / /ɪ / /s/
is	/ɪ /

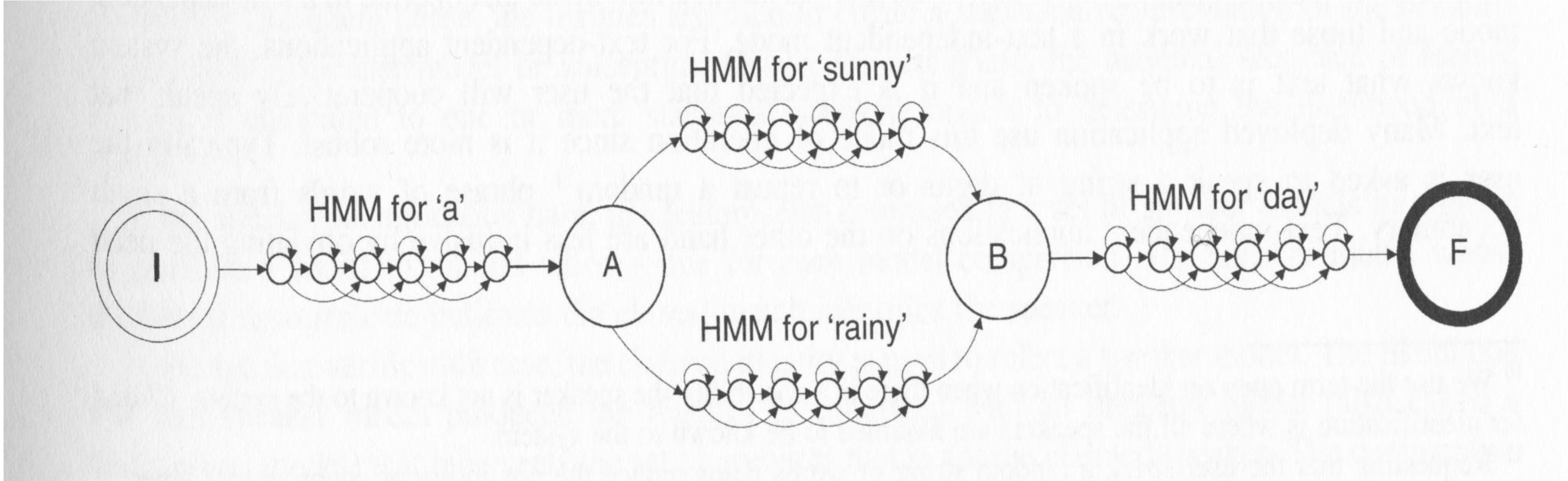
A Three-State Hidden Markov Model



a_{ij} - probability of transition from state i to state j

$b_i(k)$ - probability that observed symbol k is emitted from state i

HMM's for "A Sunny/Rainy Day"



Vocera Command Grammar Fragment

Command ::= **CallCommand** | **MessageCommand**

CallCommand ::= **Call** [**Name** | **PhoneNumber**]

Call ::= call | get me

Name ::= doctor smith
| nurse betty
| home
| captain james kirk
| a cardiologist
| room 2007

How do you pronounce “2007” in English?

- two thousand seven
- twenty oh seven
- two oh oh seven
- two double oh seven
- two zero zero seven
- two nought nought seven
- two double nought seven
- two double zero seven

Vocera Command Grammar Fragment

Command ::= **CallCommand** | **MessageCommand**

CallCommand ::= **Call** [**Name** | **PhoneNumber**]

Call ::= call | get me

Name ::= doctor smith
| nurse betty
| home
| captain james kirk
| a cardiologist
| room [two thousand seven | two double oh seven …]

Pronounciations Depend Strongly on the Language and the Context

- French
  - deux mille sept
  - deux zéro zéro sept
  - vingt zéro sept   (for a phone extension)
- Mandarin
  - 二千零七
  - 二零零七
- Japanese
  - 二千七
  - 弐阡漆

The Problem

- Find a way to quickly compute the set of all natural pronunciations for a given string of digits
- Find a way to compute a context-free grammar that generates pronunciations for all digit strings of a given length

The Solution

- Number Generating Term Rewriting Systems (NGTRS)
  - Efficient to compute
  - Intutitive, easy to verify by inspection
  - Easily adapted to different languages

Example of an NGTRS

S(dw) → T(d) S(w)
S(d) → T(d)
T(0) → zero
T(0) → oh
T(1) → one
T(2) → two
…
T(9) → nine

Variables range over strings of digits, and come in different "flavors":

d represents a single digit (0 – 9)

n represents a single non-zero digit

w represents a non-empty string of digits

$\mathbf{w^n}$ represents a string of n digits

$\mathbf{X^d}$ represents a digit other than d

Constants are the members of:

- The set D = {0, …, 9} of digits
- A set P of *pronunciation symbols* such as "nine", "oh", "twenty"

Operators consist of:

- A set of uninterpreted function symbols that map $D^+ \rightarrow P^+$

  S is the *start symbol*
- Concatenation (represented implicitly by juxtiposition)

Simple NGTRS Example

$S(dw) \rightarrow T(d)\ S(w)$

$S(d) \rightarrow T(d)$

$T(0) \rightarrow$ zero

$T(0) \rightarrow$ oh

$T(1) \rightarrow$ one

$T(2) \rightarrow$ two

…

$T(9) \rightarrow$ nine

Given a natural number q represented by a string $q_1 q_2 \ldots q_k$ of digits, we say that a string p of pronunciation symbols is a *pronunciation* of q iff

$$S(q_1 q_2 \ldots q_k) \rightarrow^{+} p$$

Simple NGTRS Example

S(dw)	→	T(d) S(w)
S(d)	→	T(d)
T(0)	→	zero
T(0)	→	oh
T(1)	→	one
T(2)	→	two
…		
T(9)	→	nine

S(2007)

→T(2) S(007)

→ two S(007)

→ two T(0) S(07)

→ two oh S(07)

→ two oh T(0) S(7)

→ two oh oh T(7)

→ two oh oh seven

Example using "double"

S(dw)	→	S(d) S(w)
S(wd)	→	S(w) S(d)
S(dd)	→	double T(d)
S(d)	→	T(d)
T(0)	→	zero
T(0)	→	oh
T(1)	→	one
T(2)	→	two
…		
T(9)	→	nine

S(2007)

→ S(2) S(007)

→ S(2) S(00) S(7)

→ S(2) double T(0) S(7)

→ S(2) double oh S(7)

…

→ two double oh seven

Constraints:

- Left-hand sides of rules must consist of a single function symbol term
- Every variable on the left-hand side must occur exactly once on the right-hand side.

- ; Room Number Generation Rules for North American English

- ; For numbers with > 10 digits, we demand that they be spoken digit by digit.
- S(y11) => G(y11)
- G(d w) => T(d) G(w)
- G(d) => T(d)

- ; For numbers with <= 10 digits, we support natural pronunciation.
- S(x10) => L(x10)
- L(w) => T(w)
- L(w 0 0 0) => T(w) "thousand"
- L(w 0 0) => T(w) "hundred"

- T(w d1 d2) => T(w) T(d1 d2)
- T(d1 d2 d3 w) => T(d1 d2) T(d3 w)
- T(d1 d2) => T(d1) T(d2)
- T(1 0) => "ten"
- T(1 1) => "eleven"
- T(1 2) => "twelve"
- T(1 3) => "thirteen"
- T(1 4) => "fourteen"
- T(1 5) => "fifteen"
- T(1 6) => "sixteen"
- T(1 7) => "seventeen"
- T(1 8) => "eighteen"
- T(1 9) => "nineteen"
- T(2 0) => "twenty"
- T(2 n) => "twenty" T(n)
- T(3 0) => "thirty"
- T(3 n) => "thirty" T(n)
- T(4 0) => "forty"
- T(4 n) => "forty" T(n)
- T(5 0) => "fifty"
- T(5 n) => "fifty" T(n)
- T(6 0) => "sixty"
- T(6 n) => "sixty" T(n)
- T(7 0) => "seventy"
- T(7 n) => "seventy" T(n)
- T(8 0) => "eighty"
- T(8 n) => "eighty" T(n)
- T(9 0) => "ninety"
- T(9 n) => "ninety" T(n)
- T(0) => "[zero oh]"
- T(1) => "one"
- T(2) => "two"
- T(3) => "three"
- T(4) => "four"
- T(5) => "five"
- T(6) => "six"
- T(7) => "seven"
- T(8) => "eight"
- T(9) => "nine"

NGTRS used for room numbers in the Vocera System

NUMBERS IN JAPANESE

〇	0 zero	十一	11 juu ichi	百	100 hyaku
一	1 ichi	十二	12 juu ni	千	1.000 sen
二	2 ni	十三	13 juu san	万	10.000 man
三	3 san	十四	14 juu shi	百万	1.000.000 hyaku man
四	4 shi	二十	20 ni-juu	億	100.000.000 oku
五	5 go	二十一	21 ni-juu ichi	兆	1.000.000.000.000 choo
六	6 rok	三十	30 san-juu		
七	7 nana	四十	40 shi-juu		
八	8 achi				
九	9 kyuu				
十	10 juu				

www.alfabetos.net

NG TRS for Japanese

S(0w)	→	S(w)
S(nw)	→	T(nw)
T(10)	→	十
T(1n)	→	十T(n)
$T(x^1d)$	→	$T(x^1)$ T(1d)
T(100)	→	百
T(10n)	→	百 T(n)
T(1n0)	→	百 T(n0)
$T(1n_1n_2)$	→	百 $T(n_1n_2)$
$T(x^1d_1d_2)$	→	$T(x^1)$ $T(1d_1d_2)$
T(1000)	→	千
T(100n)	→	千T(n)
T(10n0)	→	千T(n0)
$T(10n_1n_2)$	→	千$T(n_1n_2)$
T(1n00)	→	千 T(n00)
$T(1n_10n_2)$	→	千 $T(n_10n_2)$
$T(1n_1n_20)$	→	千 $T(1n_1n_20)$
$T(1n_1n_2n_3)$	→	千 $T(n_1n_2n_3)$
$T(x^1d_1d_2d_3)$	→	$T(x^1)$ $T(1d_1d_2d_3)$

T(0) → 零

T(1) → 一

T(2) → 二

T(3) → 三

…

T(9) → 九

S(2007)

→ T(2007)

→ T(2) T(1007)

→ 二千T(7)

→ 二千七

NG TRS for Mandarin

$S(0w)$	→	$S(w)$
$S(nw)$	→	$T(nw)$
$T(10)$	→	十
$T(x^1 0)$	→	$T(x^1)$ 十
$T(n_1 n_2)$	→	$T(n_1 0)T(n_2)$
$T(n00)$	→	$T(n)$ 百
$T(n_1 0 n_2)$	→	$T(n_1 00)$ 零$T(n_2)$
$T(n_1 n_2 0)$	→	$T(n_1 00)$ $T(n_2)$ 十
$T(n_1 n_2 n_3)$	→	$T(n_1 n_2 0)T(n_3)$
$T(n000)$	→	$T(n)$ 千
$T(n_1 00 n_2)$	→	$T(n_1 000)$ 零 $T(n_2)$
$T(n_1 0 n_2 0)$	→	$T(n_1 00 n_2)$ 十
$T(n_1 0 n_2 n_3)$	→	$T(n_1 0 n_2 0)$ $T(n_3)$
$T(n_1 n_2 00)$	→	$T(n_1 000)$ $T(n_2)$ 百
$T(n_1 n_2 0 n_3)$	→	$T(n_1 n_2 00)$ 零 $T(n_3)$
$T(n_1 n_2 n_3 0)$	→	$T(n_1 n_2 00)$ $T(n_3)$ 十
$T(n_1 n_2 n_3 n_4)$	→	$T(n_1 n_2 n_3 0)$ $T(n_4)$

$T(0)$	→	零
$T(1)$	→	一
$T(2)$	→	二
$T(3)$	→	三
…		
$T(9)$	→	九

$S(2007)$

→ $T(2000)$ 零 $T(7)$

→ $T(2)$ 千零 $T(7)$

→ 二千零 $T(7)$

→ 二千零七

Constructing a Grammar from an NGTRS

- Speech apps often require grammars for recognizing all phone numbers, tracking numbers, etc. of a certain length
- Such grammars are most often constructed by hand, and are often buggy or incomplete
- Given an NGTRS that generates pronunciations for such numbers, one can mechanically generate an equivalent CF grammar

Grammar Construction Algorithm

Construct a grammar G_N from NGTRS N:

- Terminal symbols are the pronunciation symbols of N
- Nonterminal symbols are certain terms of N
- The start symbol is $S(d_1..d_k)$ where
  - k is the # of digits in the numbers whose pronunciations are to be generated
- Additional nonterminal symbols and rules are added using the following *closure* operation:

Adding Rules to G_N

- For each nonterminal symbol T of G_N and each rule L → R of N such that T and L are *unifiable* with m.g.u. σ :
  - Add the functional terms in $R\sigma$ as new nonterminal symbols
  - Add a new rule T → $R\sigma$

Example with Two-Digit Numbers

N

S(dw) → T(d) S(w)
S(d) → T(d)
T(0) → zero
T(1) → one
T(2) → two
…
T(9) → nine

G_n

$S(d_1d_2)$ → $T(d_1)\ S(d_2)$
$S(d_2)$ → $T(d_2)$
$T(d_1)$ → zero
$T(d_1)$ → one
…
$T(d_1)$ → nine
$T(d_2)$ → zero
…
$T(d_2)$ → nine

Claim

- The language of G_N is the set of pronunications of k-digit numbers generated by N

Proof Outline

- Need to show that:
  1) Each terminal string in the language of G_N is a pronunciation of some k-digit number q in N
  2) Each pronunciation of a k-digit number in N is a terminal string in the language of G_N
  3) (Lifting Lemma) Each term of N derivable from an instance of $S(d_1 d_2 \ldots d_k)$ is an instance of a derivable string in G_N

The Language of G_N Consists of Pronunications of N

To Show: Each terminal string s of G_N is a pronunciation of a k-digit number in N

Lemma 1: Each derivable string s of G_N, considered as a term of N, is derivable in N from $S(d_1 d_2 \ldots d_k)$

Lemma 2: If t $\rightarrow^*$ u in N, then for each instance $u\alpha$ of u, there is an instance t' of t such that $t'\alpha \rightarrow^* u\alpha$

(Proofs by induction on the length of the derivations).

From Lemma 1, $S(d_1 \ldots d_k) \rightarrow^*$ s in N. Applying Lemma 2, there exists a ground instance $S(q_1 \ldots q_k)$ of $S(d_1 \ldots d_k)$ with $S(q_1 \ldots q_k) \rightarrow^*$ s. Hence s is a pronunciation of the k-digit number $q_1 \ldots q_k$.

The Language of G_N Contains All Pronuniciations of N

To Show: Each pronunication of N is a terminal string s of G_N

Lemma (Lifting): Each term of N derivable from an instance of $S(d_1 \ldots d_k)$ is an instance of a derivable string in G_N.

(Proof by induction on the length of derivations)

In particular, each pronunciation string s for a k-digit number $q_1 \ldots q_k$ in N must be an instance of a derivable string in G_N. But since s consists only of pronunciation symbols, s must be a terminal string of G_N.

- サンキュー
  谢谢你
  Hey! Thanks, man!