LING 306: Introduction to Computational Linguistics

Richard Sproat
rws@uiuc.edu
http://www.staff.uiuc.edu/ rws/
URL for this course: http://www.staff.uiuc.edu/ rws/Courses/L306

Lecture 7: Rule Compilation

September 8, 2003

Homework

- Problem 3 (= J&M, 2.8)
- Problem 5 (= J&M, 2.10–2.11)
- Problem 6

Context-dependent rewrite rules

Given a rule of the form

$$\phi \rightarrow \psi / \lambda \underline{\quad\quad} \rho$$

where ϕ, ψ, λ, ρ are regular expressions, how do you compile this rule into a transducer?

An Illustrative Example

$u \rightarrow i$ / i C^* __

($u \rightarrow i$ / Σ^* i C^* __ Σ^*)

Input: kikukuku

An Illustrative Example

u → i / i C* __

kikukuku

An Illustrative Example

u $\longrightarrow$ i / i C* __

kikukuku

An Illustrative Example

$u \longrightarrow i \;/\; i\; C^* \;\_\_$

kikikuku

An Illustrative Example

$u \longrightarrow i \;/\; i\ C^*\ \_\_$

kikikuku

An Illustrative Example

$u \longrightarrow i \; / \; i \; C^* \; \_\_$

kikikiku

An Illustrative Example

$u \longrightarrow i \ / \ i \ C^* \ \_\_$

kikikiku

An Illustrative Example

$u \longrightarrow i \;/\; i\ C^*$ ___

kikikiki

A Note on Directionality

- The output of one application *feeds* the next application across the string.
- But I took it for granted that we were applying left to right

Right-to-Left Application

$u \longrightarrow i \ / \ i \ C^* \ \_\_$

kikukuku

Right-to-Left Application

u $\longrightarrow$ i / i C* __

kikukuku

Right-to-Left Application

u $\longrightarrow$ i / i C* __

kikukuku

Right-to-Left Application

$u \longrightarrow i \;/\; i\ C^* \_\_$

kikukuku

Right-to-Left Application

u $\longrightarrow$ i / i C* __

kikikuku

Simultaneous Application

$u \longrightarrow i$ / i C^* __

kikukuku

Simultaneous Application

$u \longrightarrow i \;/\; i\ C^* \_\_$

kikukuku

Simultaneous Application

$u \longrightarrow i$ / i C* __

kikikuku

Bookkeeping

- Let's say we are being extra careful and we want to make sure that we're applying the rule correctly.
- To help us, we might use some auxiliary annotations, such as angle brackets, to mark contexts, as well as the actual string to be changed.

Bookkeeping: First Attempt

$u \rightarrow i \; / \; \Sigma^* \; i \; C^* \; \_\_ \; \Sigma^*$

1. Mark all possible right contexts:

kikukupapu

Bookkeeping: First Attempt

$u \rightarrow i \; / \; \Sigma^* \; i \; C^* \;\_\_\; \Sigma^*$

1. Mark all possible right contexts:

$$_{>}k_{>}i_{>}k_{>}u_{>}k_{>}u_{>}p_{>}a_{>}p_{>}u_{>}$$

Bookkeeping: First Attempt

u $\rightarrow$ i / Σ* i C* __ Σ*

2. Mark all possible left contexts:

$$_{>}\mathrm{k}_{>}\mathrm{i}_{>}\mathrm{k}_{>}\mathrm{u}_{>}\mathrm{k}_{>}\mathrm{u}_{>}\mathrm{p}_{>}\mathrm{a}_{>}\mathrm{p}_{>}\mathrm{u}_{>}$$

Bookkeeping: First Attempt

u $\rightarrow$ i / Σ* i C* __ Σ*

2. Mark all possible left contexts:

$$_{>}\mathrm{k}_{>}\mathrm{i}_{<>}\mathrm{k}_{<>}\mathrm{u}_{>}\mathrm{k}_{>}\mathrm{u}_{>}\mathrm{p}_{>}\mathrm{a}_{>}\mathrm{p}_{>}\mathrm{u}_{>}$$

Bookkeeping: First Attempt

u $\rightarrow$ i / Σ* i C* __ Σ*

3. Change any *u* into a *i* when it's delimited by $_<$ and $_>$.

$$_>\mathsf{k}_>\mathsf{i}_{<>}\mathsf{k}_{<>}\mathsf{u}_>\mathsf{k}_>\mathsf{u}_>\mathsf{p}_>\mathsf{a}_>\mathsf{p}_>\mathsf{u}_>$$

Bookkeeping: First Attempt

u $\rightarrow$ i / Σ* i C* __ Σ*

3. Change any *u* into a *i* when it's delimited by $_<$ and $_>$.

$$_>k_>i_<{}_>k_<{}_>i_>k_>u_>p_>a_>p_>u_>$$

Oops

This is wrong: we are not allowing for the left-to-right feeding. So how about the following:

- Mark the beginnings of right contexts—ρ. That much was correct.
- Mark the left edge of all ϕ that have $>$ on the righthand side, with *two* different markers, $<_1$, $<_2$:
  - $\star$ $<_1$ is to be used to trigger application of the change of ϕ to ψ
  - $\star$ $<_2$ is to be used for *non*application of the change
- Change ϕ into ψ wherever ϕ is delimited by $<_1$ and $>$.

  We can now delete $>$: we don't need it any more.
- This will give you a set of possible strings, some with the change, some without.

  We are now going to filter those outcomes.

- Now we check λ. In particular:
  - $\star$ Allow only strings where $<_1$ is preceded by λ; $<_1$ can then be deleted.
  - $\star$ Allow only strings where $<_2$ is *not* preceded by λ; $<_2$ can then be deleted.

Bookkeeping: Second Attempt

u $\rightarrow$ i / Σ* i C* __ Σ*

1. Mark all possible right contexts:

k i k u k u p a p u

Bookkeeping: Second Attempt

$\mathsf{u} \rightarrow \mathsf{i} \;/\; \Sigma^* \; \mathsf{i} \; \mathsf{C}^* \;\_\_\; \Sigma^*$

1. Mark all possible right contexts:

$> \mathsf{k} > \mathsf{i} > \mathsf{k} > \mathsf{u} > \mathsf{k} > \mathsf{u} > \mathsf{p} > \mathsf{a} > \mathsf{p} > \mathsf{u} >$

Bookkeeping: Second Attempt

u $\rightarrow$ i / Σ* i C* __ Σ*

2. Mark all *u* followed by $>$ with $<_1$ and $<_2$

$>$ k $>$ i $>$ k $>$ u $>$ k $>$ u $>$ p $>$ a $>$ p $>$ u $>$

Bookkeeping: Second Attempt

u $\rightarrow$ i / Σ* i C* __ Σ*

2. Mark all *u* followed by $>$ with $<_1$ and $<_2$

$>$	k	$>$	i	$>$	k	$<1, >$	u	$>$	k	$<1, >$	u	$>$	p	$>$	a	$>$	p	$<1, >$
						<2				<2								<2

Bookkeeping: Second Attempt

u $\rightarrow$ i / Σ* i C* __ Σ*

3. Change all *u* between $<_1$ and $>$, and delete $>$

$>$	k	$>$	i	$>$	k	$<1, >$ <2	u	$>$	k	$<1, >$ <2	u	$>$	p	$>$	a	$>$	p	$<1, >$ <2

Bookkeeping: Second Attempt

u $\rightarrow$ i / Σ* i C* __ Σ*

3. Change all *u* between $<_1$ and $>$, and delete $>$

k	i	k	<1	i	k	<1	i	p	a	p	<1	i
			<2	u		<2	u				<2	u

Bookkeeping: Second Attempt

u $\rightarrow$ i / Σ* i C* __ Σ*

4. Allow only strings where $<_1$ is preceded by *i C**, and delete $<_1$:

k	i	k	$<_1$	i	k	$<_1$	i	p	a	p	$<_1$	i
			$<_2$	u		$<_2$	u				$<_2$	u

Bookkeeping: Second Attempt

u $\rightarrow$ i / Σ* i C* __ Σ*

4. Allow only strings where $<_1$ is preceded by *i C**, and delete $<_1$:

k	i	k		i	k		i	p	a	p		
			<2	u		<2	u				<2	u

Bookkeeping: Second Attempt

u $\rightarrow$ i / Σ* i C* __ Σ*

5. Allow only strings where $<_2$ is *not* preceded by *i C**, and delete $<_2$:

k	i	k		i	k		i	p	a	p		
			<2	u		<2	u				<2	u

Bookkeeping: Second Attempt

u $\rightarrow$ i / Σ* i C* __ Σ*

5. Allow only strings where $<_2$ is *not* preceded by *i C**, and delete $<_2$:

k	i	k	i	k	i	p	a	p	
									u

Now in fact . . .

Each of these five steps can be represented as a transducer:

- A transducer r that inserts $>$ before every ρ.
- A transducer f that inserts $<_1$ and $<_2$ before every ϕ followed by $>$.
- A transducer $replace$ that replaces ϕ with ψ between $<_1$ and $>$, and also deletes $>$.
- A transducer l_1 that filters out all $<_1$ not preceded by λ, and deletes any $<_1$
- A transducer l_2 that filters out all $<_2$ preceded by λ, and deletes any $<_2$

So the Rule Transducer Is

$$r \circ f \circ replace \circ l_1 \circ l_2$$

Literature

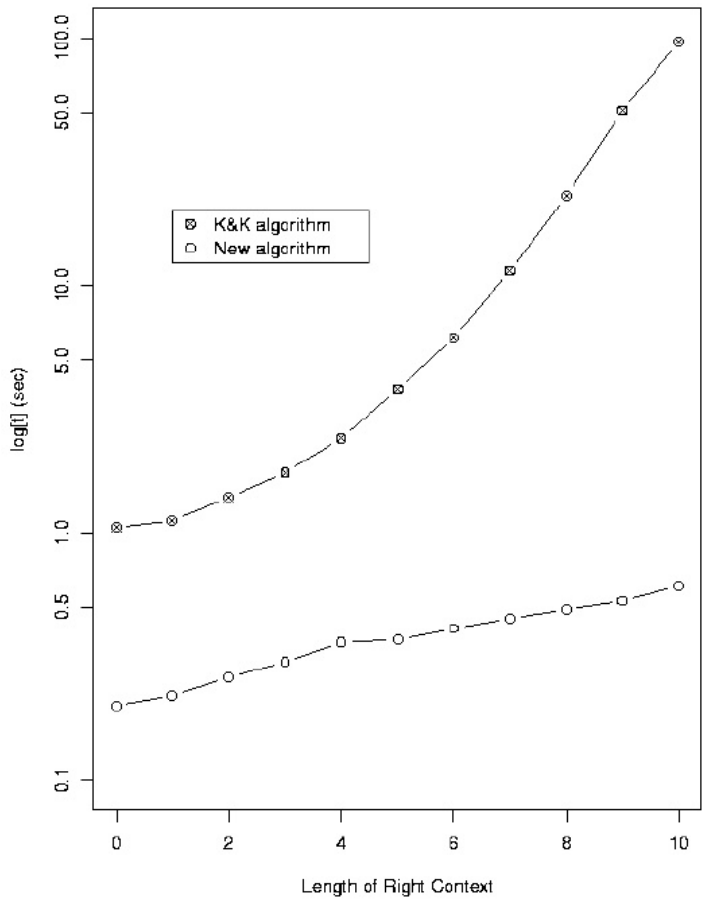
- The formulation I just sketched for left-to-right rules is from Mohri and Sproat (1996) "An Efficient Compiler for Weighted Rewrite Rules", *34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, CA, available at `http://acl.ldc.upenn.edu/P/P96/P96-1031.pdf`

- It differs from the earlier Kaplan and Kay formulation in part because K&K introduce brackets everywhere, and subsequently filter, whereas M&S try to only introduce brackets where they are really needed.

- They are algebraically equivalent, but the M&S formulation is algorithmically better.

Compilation Times

Rules of the form $a \longrightarrow b/\_\_\_c^k$, $(k \in [0, 10])$



Comparison in a Real Example

Rules	KK			New		
	time (s)	space		time (s)	space	
		states	arcs		states	arcs
<ö>	62	412	50,475	47	394	47,491
<a>	284	1,939	215,721	240	1,927	213,408

- German text-to-speech system (B. Möbius)
  - ⋆ Pronunciation rules for <ö> and <a>
  - ⋆ ≈ 20 sets of rules (≈ 200 rules)

- Time for overall construction of each set of rules (SGI Indy 4000, 100 MHZ IP20, 128 Mbytes RAM)

Reading

Nada: Have a Nice Weekend.