Finite-State Methods in Natural-Language Processing: Rewriting Rules

Ronald M. Kaplan

and

Martin Kay
Reference

Spelling Conventions

N → m/ __ +labial
N → n

iN+tractable → intractable
iN+practical → impractical

iN is the common negative prefix
— im before labial
— in otherwise

c.f. input → input
Rewriting Rules

\[ \alpha \rightarrow \beta / \lambda \_ \rho \]

No rerewriting!

\[ \varepsilon \rightarrow ab / a\_b \]

would map \( ab \) into \( a^n b^n \)

\[ \alpha \rightarrow \beta / \lambda \_ \rho \] is not the same as \( \lambda \alpha \rho \rightarrow \lambda \beta \rho \]
An in/im Transducer
Ordered rules

<table>
<thead>
<tr>
<th></th>
<th>try}#</th>
<th>try}s#</th>
<th>try}ed#</th>
<th>try}ing#</th>
</tr>
</thead>
<tbody>
<tr>
<td>y-&gt;i/C_}</td>
<td>tri}#</td>
<td>tri}s#</td>
<td>tri}ed#</td>
<td>tri}ing#</td>
</tr>
<tr>
<td>0-&gt;e/[ilsib]}_s#</td>
<td>tri}#</td>
<td>tri}es#</td>
<td>tri}ed#</td>
<td>tri}ing#</td>
</tr>
<tr>
<td>e-&gt;0/_}e</td>
<td>tri}#</td>
<td>tri}es#</td>
<td>tri}ed#</td>
<td>tri}ing#</td>
</tr>
<tr>
<td>i-&gt;y/_}[#, i]</td>
<td>try}#</td>
<td>tri}es#</td>
<td>tri}ed#</td>
<td>try}ing#</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>tie}#</th>
<th>tie}s#</th>
<th>tie}ed#</th>
<th>tie}ing#</th>
</tr>
</thead>
<tbody>
<tr>
<td>y-&gt;i/C_}</td>
<td>tie}#</td>
<td>tie}s#</td>
<td>tie}ed#</td>
<td>tie}ing#</td>
</tr>
<tr>
<td>0-&gt;e/[ilsib]}_s#</td>
<td>tie}#</td>
<td>tie}s#</td>
<td>tie}ed#</td>
<td>tie}ing#</td>
</tr>
<tr>
<td>e-&gt;0/_}e</td>
<td>tie}#</td>
<td>tie}es#</td>
<td>ti}ed#</td>
<td>ti}ing#</td>
</tr>
<tr>
<td>i-&gt;y/_}[#, i]</td>
<td>tie}#</td>
<td>tie}es#</td>
<td>ti}ed#</td>
<td>ty}ing#</td>
</tr>
</tbody>
</table>
## Feeding

<table>
<thead>
<tr>
<th>Rule</th>
<th>Target</th>
<th>Source 1</th>
<th>Source 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>y→i/C₁</td>
<td>try#</td>
<td>tie#</td>
<td></td>
</tr>
<tr>
<td>0→e/[ilsib]₁s</td>
<td>tri#</td>
<td>tie#</td>
<td></td>
</tr>
<tr>
<td>e→0/_₁e</td>
<td>tri#</td>
<td>ti#</td>
<td></td>
</tr>
<tr>
<td>i→y/_₁[#₁]</td>
<td>try#</td>
<td>ty#</td>
<td></td>
</tr>
</tbody>
</table>
Application Direction—Examples

\[ V: \rightarrow V / V: C^* _{\_} \]

**Gidabal**

- left to right
  
  g u n u: m + b a: + d a: ng + b e: +
  
  g u n u: m + b a + d a: ng + b e +

  *is certainly right on the stump*

**Slovak**

- right to left or simultaneous
  
  v o l + a: v + a: m e:
  
  v o l + a: v + a m e

  *we call often*

Kaplan & Kay 5 - 8
Direction of Application

\[ a \rightarrow b / ab \_ ba \]

- Input
  - \( abababababa \)
- Output
  - \( abbbbabbbaba \) (Left to right)
  - \( ababbbbabbbba \) (Right to left)
  - \( abbbbbbbbbba \) (Simultaneous)

Left to right

Right to Left

Kaplan & Kay 5 - 9
Optional Rules

\[ a \rightarrow x / b \]

Diagram:

```
abaca
   /\  \
abaca  abxca
```
Special Operators

- $\text{Intro}(S) =_{df} [\text{Id}(\Sigma) \cup [\{\varepsilon\} \times S]]^*$

  Strings of $\Sigma^*$ with members of the set $S$ of symbols freely interspersed. $\text{Intro}^{-1}(S)$ removes members of $S$ if $S$ and $\Sigma$ are disjoint.

- $L_S =_{df} \text{Range}((\text{Id}(L) \circ \text{Intro}(S))$  

  Ignore

  Strings that would be in $L$ if some symbols in $S$ were removed.

- $\text{If-}\text{-P-then-}\text{-S}(L_1, L_2) =_{df} \{x_1x_2 \mid \text{if } x_1 \in L_1, \text{ then } x_2 \in L_2\} = L_1\overline{L_2}$

- $\text{If-S-then-P}(L_1, L_2) = L_1L_2 = \text{If-}\text{-P-then-}\text{-S}(L_1, L_2)$

- $\text{P-iff-S}(L_1, L_2) = \text{If-}\text{-P-then-}\text{-S}(L_1, L_2) \cap \text{If-S-then-P}(L_1, L_2)$
Optional Replacement

\[ \alpha \rightarrow \beta \text{ (optional)} \]

\[ \text{Replace} = [\text{Id}(\Sigma)^* \text{ Opt}(\alpha \times \beta)]^* \]

where \( \text{Opt}(R) = R \cup \{<\varepsilon, \varepsilon>\} \)
Context Restrictions—1

Replace = [Id(Σ)* Opt(Id(λ) αβ Id(ρ))]*

B → b / V _ V

V B V B V
V b V B V

Accepts this

V B V B V
V b V b V

but not this

... but both should be accepted
Context Restrictions—2

Put markers in the right places

Replace = \[ Id(\Sigma)^* \text{Opt}(Id(<) \alpha_m \times \beta_m Id(>))]^*\]

Remove markers

\[ X_m \text{ is } X \text{ with } m\text{'s freely interspersed} \]
Left Context — 1

Purpose: Eliminate brackets that are not preceded by an instance of the left context.

\[ \text{LC}(\lambda) = (\Sigma^* \lambda \ <)^* \Sigma^* \]

Every \(<\) is preceded by \(\lambda\).

But there can be \(\lambda\) without \(<\)

By default, \(\langle\rangle \notin \Sigma\)

Nondeterministic

Kaplan & Kay 5 - 15
But there can be $\lambda$ without $<$

$$LC(\lambda) = \text{P-iff-S}(\Sigma^* \lambda, < \Sigma^*)$$

$$\lambda = a \ b$$

$$LC(\lambda) = ((\Sigma^* \lambda <)^* \Sigma^*)$$
Left Context — 3

\[ \text{LC}(\lambda) = \text{P-iff-S}(\Sigma^* \lambda, < \Sigma^*) \]

What about \( \lambda \)-prefixes in \( \lambda \)?

\[ \text{LC}(a(b)) \text{ contains } a < \]
\[ \quad \text{a < b} \]
\[ \quad \text{a b <} \]
\[ \quad \text{but not } a < b < \]

\[ \text{LC}(\lambda) = \text{P-iff-S}(\ldots) \]
What if $\varepsilon \in \lambda$?

Consider $\{\varepsilon\} = \lambda$. Every substring must be followed by $<$, including substrings that end in $<$. The language therefore contains no strings of finite length.

$$LC(\lambda) = P \text{-iff} S((\Sigma^* \lambda)_< < \Sigma^*_<)$$
Left Context—3

This must always be followed by this

\[ LC(\lambda) = P\text{-iff-}S((\Sigma^* \lambda)_<( - (\Sigma^* _<)_>, < \Sigma^*_<)) \]

And this must always be preceded by this
Left Context—3

The left context, ignoring < But not ending in <

\[ LC(\lambda) = P\text{-iff-}S((\Sigma^* \lambda)_< - (\Sigma^*_< <), < \Sigma^*_< <) \]
Left Context—3

The left context, ignoring $<$  

But not ending in $<$

\[ LC (\lambda) = P\text{-iff-S}((\Sigma^* \lambda)_< - (\Sigma^* _<_), \ < \Sigma^* _<_>) \]

Always ignore the "other" marker

Kaplan & Kay 5 - 21
Left Context Examples

\[ \lambda = abab \]

\[ xabxabab < xabab < ab < x \]

\[ \lambda = \varepsilon \]

\[ < a < b < c < d < e < f < g < \]
Left and Right Context

\[ LC(\lambda, \ell, r) = P-\text{iff-}S((\Sigma^* \lambda)_{\ell} - (\Sigma^* \ell)_{\ell}, < \Sigma^* \ell)_r \]

\[ \text{LeftContext}(\lambda) = LC(\lambda, <, >) \]

\[ \text{RightContext}(\rho) = \text{Reverse}(LC(\text{Reverse}(\rho), >, <)) \]
The Empty Left Context

$LeftContext(\varepsilon)$
The Empty Right Context

$RightContext(\varepsilon)$
One-character Left Context

\textit{LeftContext}(a)

\begin{center}
\includegraphics[width=0.8\textwidth]{diagram}
\end{center}
One Character Right Context

RightContext(a)
\[ \alpha \rightarrow \beta / \lambda \_ \rho \quad \text{Optional} \]

<table>
<thead>
<tr>
<th>Left to Right</th>
<th>Right to Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{Intro}({&lt;, &gt;}) \circ \textit{Id}(\text{RightContext}(\rho)) \circ \textit{Replace}(\alpha, \beta) \circ \textit{Id}(\text{LeftContext}(\lambda)) \circ \textit{Intro}^{-1}({&lt;, &gt;})</td>
<td>\textit{Intro}({&lt;, &gt;}) \circ \textit{Id}(\text{LeftContext}(\lambda)) \circ \textit{Replace}(\alpha, \beta) \circ \textit{Id}(\text{RightContext}(\rho)) \circ \textit{Intro}^{-1}({&lt;, &gt;})</td>
</tr>
</tbody>
</table>

Kaplan & Kay 5 - 28
Optional Simultaneous Application

\[ \text{Intro}(\{<, >\}) \circ \text{Id}(\text{RightContext}(\rho) \cap \text{LeftContext}(\lambda)) \circ \text{Replace}(\alpha, \beta) \circ \text{Intro}^{-1}(\{<, >\}) \]
Obligatory rules—Brackets

\(<_{a}>_{a}\) application
\(<_{i}>_{i}\) ignored
\(<_{c}>_{c}\) center

Whenever a rule is properly applied, the input side of the replacement relation will contain a substring of the form

\(<_{a}\alpha_{<_{c}>_{c}}>_{a}\)

Notation:

\(<=\{<_{a},<_{i},<_{c}\}\)
\(>=\{>_{a},>_{i},>_{c}\}\)
**Obligatory Rules**

\[ \text{Obligatory}(\phi, l, r) = \Sigma^*_{m,0} l \phi^0_m r \Sigma^*_{m,0} \]

**Left to Right**

- **Prolog**
- **Id**(Obligatory(φ, <i, >))
- **Id**(RightContext(ρ,<,>))
- **Replace** (φ , ψ)
- **Id**(LeftContext(λ,<,>))
- **Prolog**⁻¹

**Right to Left**

- **Prolog**
- **Id**(Obligatory(φ, <, >i))
- **Id**(LeftContext(λ,<,>))
- **Replace**(φ , ψ)
- **Id**(RightContext(ρ,<,>))
- **Prolog**⁻¹
Other Topics

• Boundary Contexts
• Features
• 2-level Rules
Composition

Intractable impractical
0 0 2 0 0 0 0 0 0 0 0 0 0

Intractable impractical
0 0 0 0 0 0 0 0 0 0 0 0 0

Intractable impractical
0 0 0 0 0 0 0 0 0 0 0 0 0

Kaplan & Kay 5 - 33
Lexical and Surface Forms

intractable → intractable
input → input
iNpractical → impractical

1. N → m / __ [+labial]
2. N → n

Archiphonemes
An Optional Rule

N → n (Optional)

Start state: 0
Usually leftmost in the diagram

Final states have double circles
An Obligatory Rule

\[ N \rightarrow n \text{ (Obligatory)} \]

other = all matched pairs of symbols not otherwise mentioned on any transition.
Context

\[ N \rightarrow m / \_ [+\text{labial}] \] (Obligatory)

No exit from this state except over a labial.
Composition

<table>
<thead>
<tr>
<th>input</th>
<th>m-machine</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>labial follows</td>
<td>m</td>
</tr>
<tr>
<td>N</td>
<td>nonlabial follows</td>
<td>N</td>
</tr>
</tbody>
</table>

Kaplan & Kay 5 - 39
Generation — “intractable”
Generation — “impractical”
Recognition — “intractable”
Generation — "input"
A Word Transducer

Base Forms

Morphology

Finite-State Transducers

Spelling Rules

Text Forms

Finite-state machine

Finite-state Machine
We need a separate left-context marker for each application of the rule. But they all have to be in the same place. Solution: Temporarily replace a with 0, and delete this later.
Epenthesis and Deletion

\[ LC(\lambda, l, r, a) = P-iff-S((\Sigma^* \lambda)_{l,a} - (\Sigma^* l), 1 \Sigma^*)_r \]

LeftContext(\lambda) = LC(\lambda, <, {>}, >) \\
RightContext(\rho) = Reverse(LC(Reverse(\rho), >, {<}, <))
Other Topics

• Boundary Contexts
• Batch Rules
• Features
Theorem

• Every rewriting grammar denotes a regular relation.
• Every regular relation is denoted by some rewriting grammar.
Two-level Rule Systems
Parallel Combination

Lexical String

T1  T2  Tn

Surface String

Kaplan & Kay 5 - 50
Parallel Combination

Lexical String

Intro(0)

T1

Intro⁻¹(0)

T2

Tn

Surface String
Mathematically ...

\[ Z = \text{Intro}(0) \]

\[ Z \circ [\bigcap_i T_i] \circ Z^{-1} \quad \text{(Regular)} \]

\[ \neq \bigcap_i [Z \circ T_i \circ Z^{-1}] \quad \text{(May not be regular)} \]
Spies

\[
\begin{align*}
spy + s &\# \\
spy \ 0\ s &\ 0 \\
spy + s &\# \\
spi \ 0\ s &\ 0 \\
spy + 0\ s &\# \\
spy \ 0\ e &\ s \ 0 \\
spy + 0\ s &\# \\
spi + e &\ s \ 0
\end{align*}
\]

\[
\Pi = a: a \ldots z: z, 0: e, y: i, +: 0, #: 0
\]
Sib = \{s, x, z\}

\[
\begin{align*}
0: e &\iff (\text{Sib: or } y:) +: _{s: s} #: 0 \\
y: y &\iff _{\#: \ or} +: i: \ or \ \Pi-\{+: #:\}
\end{align*}
\]
Two-level Rules

\[ \tau \Rightarrow \lambda \_ \rho \]  
**Context restriction**

If a \( \tau \) pair-string appears, \( \lambda \) is before and \( \rho \) is after.

\[ \tau \Leftarrow \lambda \_ \rho \]  
**Surface Coercion**

If the first component of a pair-string appearing between \( \lambda \) and \( \rho \) is in \( \text{Domain}(\tau) \), that pair-string must be in \( \tau \).

\[ \tau \iff \lambda \_ \rho \]  
**Bicondition**

\( \tau \Rightarrow \lambda \_ \rho \) and \( \tau \Leftarrow \lambda \_ \rho \)

\[ \tau /\Leftarrow \lambda \_ \rho \]  
**Surface Prohibition**

If the first component of a pair-string appearing between \( \lambda \) and \( \rho \) is in \( \text{Domain}(\tau) \), that pair-string must NOT be in \( \tau \).
Observations

- \( \tau, \lambda \) and \( \rho \) in \( \Pi^* \) are same-length relations
- Therefore, closed under intersection and complementation
- Therefore, we can define

\[
\text{If-P-then-S}(R_1, R_2) = \Pi^*- R_1 (\Pi^*- R_2)
\]

\[
\text{If-S-then-P}(R_1, R_2) = \Pi^*- (\Pi^*- R_1) R_2
\]