Translating Rewrite Rules to Finite State Automata

Ling 667
5/7/2012
Adam Jardine
Overview

• This work is an (attempted) implementation of the ideas in Kaplan & Kay (1994): *Regular Models of Phonological Rule Systems*

• K&K define how phonological rewrite rules, like the ones in Chomsky & Halle's *The Sound Pattern of English*, can be expressed using regular relations

• Any regular relation can be implemented with an FSA
Overview

- K&K break down their translation into a series of operations on regular relations and regular languages; i.e.:
  - Concatenation \([R_1 \cdot R_2]\) or \([L_1 \cdot L_1]\)
  - Kleene Closure \(R^*\) or \(L^*\)
  - Optionality \(\text{Opt}(R)\)
  - Identity \(\text{Id}(L)\)
  - Cross Product \(R_1 \times R_2\)

- These operations can be implemented in OCaml as functions that take one or more FSA's as arguments and return a new FSA
Overview

- This is most easily implemented using a non-deterministic finite-state *transducer* with *epsilon* transitions

Non-deterministic FST, from K&K 1994 (p.334)
The Relation

- Kaplan & Kay initially define the regular relation translation of the rule
  \[ A \rightarrow B / X \_\_\_ Y \]
  as
  \[ [\text{Id}(\Sigma^*) \text{ Opt } (\text{Id}(X) A \times B \text{ Id}(Y))]^* \]

- Let's see how this works...