Learning Left-to-Right and Right-to-Left Iterative Languages

Jeffrey Heinz
heinz@udel.edu

University of Delaware

St. Malo
September 24, 2008

LRI and RLI Languages

1. previously unnoticed infinite subclasses of the regular languages
2. identifiable in the limit from positive data
3. essentially the classes obtainable by merging final and start states in prefix and suffix trees, respectively

LRI and RLI Languages

1. previously unnoticed infinite subclasses of the regular languages
2. identifiable in the limit from positive data
3. essentially the classes obtainable by merging final and start states in prefix and suffix trees, respectively

LRI and RLI Languages

1. previously unnoticed infinite subclasses of the regular languages
2. identifiable in the limit from positive data
3. essentially the classes obtainable by merging final and start states in prefix and suffix trees, respectively

Why they are interesting

1. related algorthimically to the zero-reversible languages (remove one line!) (Angluin 1982)
2. a step towards mapping out space of language classes obtainable by Muggleton's (1990) general state-merging IM1 algorithm
3. help reveal the algebraic structure underlying state-merging and the reverse operator
4. related to a linguistic hypothesis: all phonotactic patterns are neighborhood-distinct (Heinz 2007)

Why they are interesting

1. related algorthimically to the zero-reversible languages (remove one line!) (Angluin 1982)
2. a step towards mapping out space of language classes obtainable by Muggleton's (1990) general state-merging IM1 algorithm
3. help reveal the algebraic structure underlying state-merging and the reverse operator
4. related to a linguistic hypothesis: all phonotactic patterns are neighborhood-distinct (Heinz 2007)

Why they are interesting

1. related algorthimically to the zero-reversible languages (remove one line!) (Angluin 1982)
2. a step towards mapping out space of language classes obtainable by Muggleton's (1990) general state-merging IM1 algorithm
3. help reveal the algebraic structure underlying state-merging and the reverse operator
4. related to a linguistic hypothesis: all phonotactic patterns are neighborhood-distinct (Heinz 2007)

Why they are interesting

1. related algorthimically to the zero-reversible languages (remove one line!) (Angluin 1982)
2. a step towards mapping out space of language classes obtainable by Muggleton's (1990) general state-merging IM1 algorithm
3. help reveal the algebraic structure underlying state-merging and the reverse operator
4. related to a linguistic hypothesis: all phonotactic patterns are neighborhood-distinct (Heinz 2007)

What are phonotactic patterns?

- Rules or constraints governing **word well-formedness**
- Possible words of English:

{ slam, fist, blick, flump, . . . }

- This set excludes:

{ sram, fizt, bnick, flumk, . . . }

What are phonotactic patterns?

- Rules or constraints governing **word well-formedness**
- Possible words of English:

{ slam, fist, blick, flump, . . . }

- This set excludes:

{ sram, fizt, bnick, flumk, . . . }

What are phonotactic patterns?

- Rules or constraints governing **word well-formedness**
- Possible words of English:

  { slam, fist, blick, flump, . . . }
- This set excludes:

  { sram, fizt, bnick, flumk, . . . }

Specific Sound Patterns

- No Triple Consonant Clusters in Yokuts:
  - Includes { ab, abba, ababa, . . . }
  - Excludes { bbb, abbb, abbba, bbbba, . . . }
- Symmetric Sibilant Harmony (Navajo):
  - Includes {sos, sototos, . . . ʃoʃ, ʃotototoʃ . . . }
  - Excludes { soʃ, ʃotos, sototoʃ, . . . }
- Asymmetric Sibilant Harmony (Sarcee):
  - Includes {sos, sototos, . . . ʃoʃ, ʃotototoʃ . . . ʃos, ʃotos, . . . }
  - Excludes { soʃ, sotoʃ, sototoʃ, . . . }

Specific Sound Patterns

- No Triple Consonant Clusters in Yokuts:
  - Includes { ab, abba, ababa, . . . }
  - Excludes { bbb, abbb, abbba, bbbba, . . . }
- Symmetric Sibilant Harmony (Navajo):
  - Includes {sos, sototos, . . . ʃoʃ, ʃototoʃ . . . }
  - Excludes { soʃ, ʃotos, sototoʃ, . . . }
- Asymmetric Sibilant Harmony (Sarcee):
  - Includes {sos, sototos, . . . ʃoʃ, ʃototoʃ . . . ʃos, ʃotos, . . . }
  - Excludes { soʃ, sotoʃ, sototoʃ, . . . }

Specific Sound Patterns

- No Triple Consonant Clusters in Yokuts:
  - Includes { ab, abba, ababa, . . . }
  - Excludes { bbb, abbb, abbba, bbbba, . . . }
- Symmetric Sibilant Harmony (Navajo):
  - Includes {sos, sototos, . . . ʃoʃ, ʃotototoʃ . . . }
  - Excludes { soʃ, ʃotos, sototoʃ, . . . }
- Asymmetric Sibilant Harmony (Sarcee):
  - Includes {sos, sototos, . . . ʃoʃ, ʃotototoʃ . . . ʃos, ʃotos, . . . }
  - Excludes { soʃ, sotoʃ, sototoʃ, . . . }

Specific Sound Patterns

- No Triple Consonant Clusters in Yokuts:
  - Includes { ab, abba, ababa, . . . }
  - Excludes { bbb, abbb, abbba, bbbba, . . . }
- Symmetric Sibilant Harmony (Navajo):
  - Includes {sos, sototos, . . . ʃoʃ, ʃototoʃ . . . }
  - Excludes { soʃ, ʃotos, sototoʃ, . . . }
- Asymmetric Sibilant Harmony (Sarcee):
  - Includes {sos, sototos, . . . ʃoʃ, ʃototoʃ . . . ʃos, ʃotos, . . . }
  - Excludes { soʃ, sotoʃ, sototoʃ, . . . }

Specific Sound Patterns

- No Triple Consonant Clusters in Yokuts:
  - Includes { ab, abba, ababa, . . . }
  - Excludes { bbb, abbb, abbba, bbbba, . . . }
- Symmetric Sibilant Harmony (Navajo):
  - Includes {sos, sotototos, . . . ʃoʃ, ʃototototoʃ . . . }
  - Excludes { soʃ, ʃotos, sototoʃ, . . . }
- Asymmetric Sibilant Harmony (Sarcee):
  - Includes {sos, sotototos, . . . ʃoʃ, ʃototototoʃ . . . ʃos, ʃotos, . . . }
  - Excludes { soʃ, sotoʃ, sototoʃ, . . . }

Specific Sound Patterns

- No Triple Consonant Clusters in Yokuts:
  - Includes { ab, abba, ababa, . . . }
  - Excludes { bbb, abbb, abbba, bbbba, . . . }
- Symmetric Sibilant Harmony (Navajo):
  - Includes {sos, sotototos, . . . ʃoʃ, ʃotototoʃ . . . }
  - Excludes { soʃ, ʃotos, sototoʃ, . . . }
- Asymmetric Sibilant Harmony (Sarcee):
  - Includes {sos, sotototos, . . . ʃoʃ, ʃotototoʃ . . . ʃos, ʃotos, . . . }
  - Excludes { soʃ, sotoʃ, sototoʃ, . . . }

Specific Sound Patterns

- No Triple Consonant Clusters in Yokuts:
  - Includes { ab, abba, ababa, . . . }
  - Excludes { bbb, abbb, abbba, bbbba, . . . }
- Symmetric Sibilant Harmony (Navajo):
  - Includes {sos, sototos, . . . ʃoʃ, ʃotototoʃ . . . }
  - Excludes { soʃ, ʃotos, sototoʃ, . . . }
- Asymmetric Sibilant Harmony (Sarcee):
  - Includes {sos, sototos, . . . ʃoʃ, ʃotototoʃ . . . ʃos, ʃotos, . . . }
  - Excludes { soʃ, sotoʃ, sototoʃ, . . . }

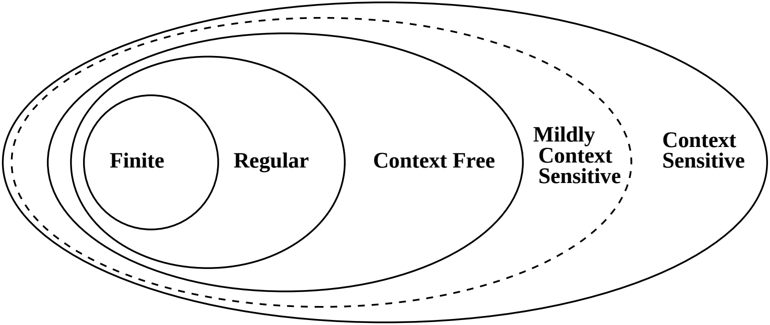
Specific Sound Patterns

- No Triple Consonant Clusters in Yokuts:
  - Includes { ab, abba, ababa, ... }
  - Excludes { bbb, abbb, abbba, bbbba, ... }
- Symmetric Sibilant Harmony (Navajo):
  - Includes {sos, sototos, ...ʃoʃ, ʃototoʃ ...}
  - Excludes { soʃ, ʃotos, sototoʃ, ... }
- Asymmetric Sibilant Harmony (Sarcee):
  - Includes {sos, sototos, ...ʃoʃ, ʃototoʃ ...ʃos, ʃotos, ... }
  - Excludes { soʃ, sotoʃ, sototoʃ, ... }

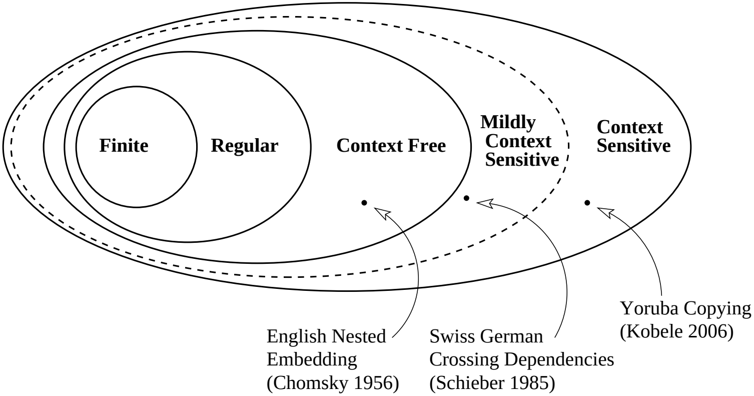
Specific Sound Patterns

- No Triple Consonant Clusters in Yokuts:
  - Includes { ab, abba, ababa, . . . }
  - Excludes { bbb, abbb, abbba, bbbba, . . . }
- Symmetric Sibilant Harmony (Navajo):
  - Includes {sos, sotototos, . . . ʃoʃ, ʃototoʃ . . . }
  - Excludes { soʃ, ʃotos, sototoʃ, . . . }
- Asymmetric Sibilant Harmony (Sarcee):
  - Includes {sos, sotototos, . . . ʃoʃ, ʃotototoʃ . . . ʃos, ʃotos, . . . }
  - Excludes { soʃ, sotoʃ, sototoʃ, . . . }

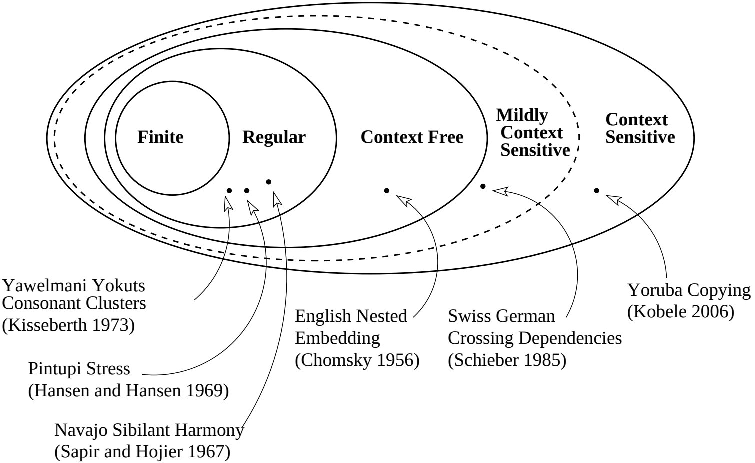
Language Patterns and the Chomsky Hierarchy



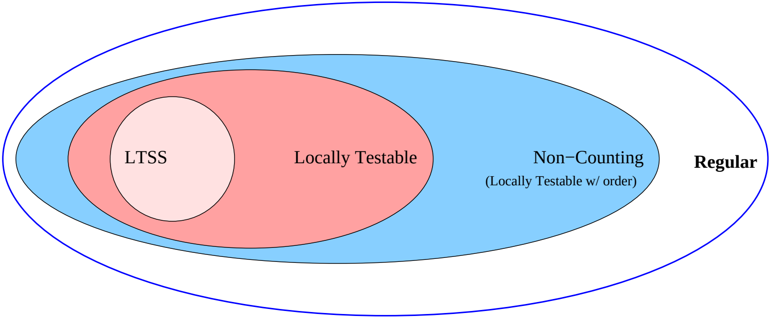
Language Patterns and the Chomsky Hierarchy



Language Patterns and the Chomsky Hierarchy

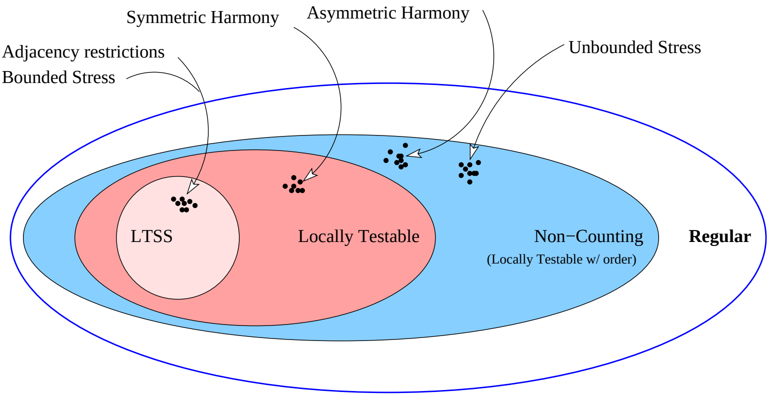


The Subregular Hierarchy



(McNaughton and Papert 1971, Pullum and Rogers 2007)

The Subregular Hierarchy



(Greenberg 1978, Hansson 2001, Hayes 1995)

Grammatical Inference of Regular Languages is Theoretical Phonology

- The properties of learnable subclasses of the regular languages are candidates as universal properties of sound patterns

  E.g. Angluin 1982, Muggleton 1990, Fernau 2003, . . .

- Which can be evaluated by comparing them to the
  1. Linguists' knowledge of the range of variation

     E.g. Greenberg 1978, Hansson 2001, Hayes 1995, . . .

  2. Psycholinguistic evidence about the state of infants' knowledge

     E.g. Juscyk et al. 1999, Mattys and Juscyk 2001, Saffran et al. 1996, Saffran and Thiessen 2003, . . .

Grammatical Inference of Regular Languages is Theoretical Phonology

- The properties of learnable subclasses of the regular languages are candidates as universal properties of sound patterns

  E.g. Angluin 1982, Muggleton 1990, Fernau 2003, …

- Which can be evaluated by comparing them to the
  1. Linguists' knowledge of the range of variation

     E.g. Greenberg 1978, Hansson 2001, Hayes 1995, …

  2. Psycholinguistic evidence about the state of infants' knowledge

     E.g. Jusczyk et al. 1999, Mattys and Jusczyk 2001, Saffran et al. 1996, Saffran and Thiessen 2003, …

Grammatical Inference of Regular Languages is Theoretical Phonology

- The properties of learnable subclasses of the regular languages are candidates as universal properties of sound patterns

  E.g. Angluin 1982, Muggleton 1990, Fernau 2003, ...
- Which can be evaluated by comparing them to the
  1. Linguists' knowledge of the range of variation

     E.g. Greenberg 1978, Hansson 2001, Hayes 1995, ...
  2. Psycholinguistic evidence about the state of infants' knowledge

     E.g. Juscyk et al. 1999, Mattys and Juscyk 2001, Saffran et al. 1996, Saffran and Thiessen 2003, ...

Grammatical Inference of Regular Languages is Theoretical Phonology

- The properties of learnable subclasses of the regular languages are candidates as universal properties of sound patterns

  E.g. Angluin 1982, Muggleton 1990, Fernau 2003, . . .

- Which can be evaluated by comparing them to the
  1. Linguists' knowledge of the range of variation

     E.g. Greenberg 1978, Hansson 2001, Hayes 1995, . . .

  2. Psycholinguistic evidence about the state of infants' knowledge

     E.g. Juscyk et al. 1999, Mattys and Juscyk 2001, Saffran et al. 1996, Saffran and Thiessen 2003, . . .

Grammatical Inference of Regular Languages is Theoretical Phonology

- The properties of learnable subclasses of the regular languages are candidates as universal properties of sound patterns

  E.g. Angluin 1982, Muggleton 1990, Fernau 2003, ...

- Which can be evaluated by comparing them to the
  1. Linguists' knowledge of the range of variation

     E.g. Greenberg 1978, Hansson 2001, Hayes 1995, ...

  2. Psycholinguistic evidence about the state of infants' knowledge

     E.g. Juscyk et al. 1999, Mattys and Juscyk 2001, Saffran et al. 1996, Saffran and Thiessen 2003, ...

Grammatical Inference of Regular Languages is Theoretical Phonology

- The properties of learnable subclasses of the regular languages are candidates as universal properties of sound patterns

  E.g. Angluin 1982, Muggleton 1990, Fernau 2003, . . .

- Which can be evaluated by comparing them to the

  1. Linguists' knowledge of the range of variation

     E.g. Greenberg 1978, Hansson 2001, Hayes 1995, . . .

  2. Psycholinguistic evidence about the state of infants' knowledge

     E.g. Juscyk et al. 1999, Mattys and Juscyk 2001, Saffran et al. 1996, Saffran and Thiessen 2003, . . .

Grammatical Inference of Regular Languages is Theoretical Phonology

- The properties of learnable subclasses of the regular languages are candidates as universal properties of sound patterns

  E.g. Angluin 1982, Muggleton 1990, Fernau 2003, …

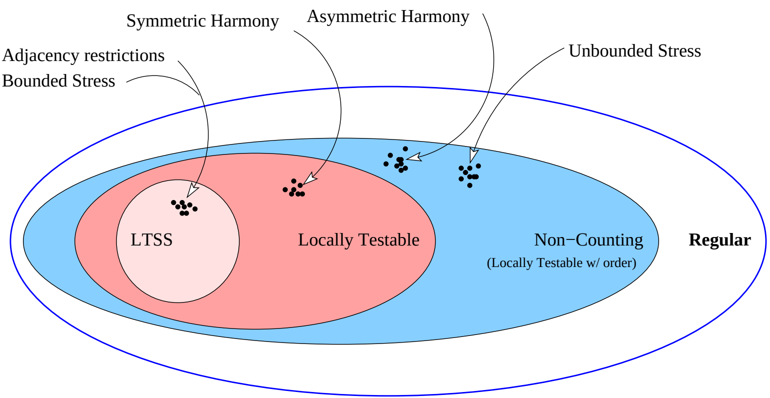
- Which can be evaluated by comparing them to the
  1. Linguists' knowledge of the range of variation

     E.g. Greenberg 1978, Hansson 2001, Hayes 1995, …

  2. Psycholinguistic evidence about the state of infants' knowledge

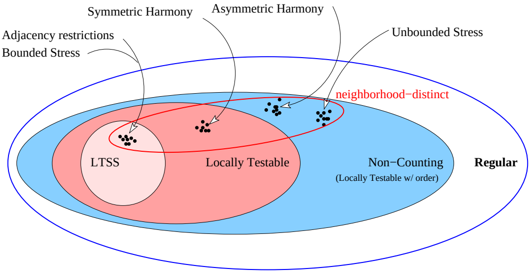
     E.g. Juscyk et al. 1999, Mattys and Juscyk 2001, Saffran et al. 1996, Saffran and Thiessen 2003, …

The Subregular Hierarchy



- For small neighborhoods, they are all neighborhood-distinct.
  - 3-LTSS ⊂ 1-1 ND
  - precedence languages ⊂ 1-1 ND

The Subregular Hierarchy



- For small neighborhoods, they are all neighborhood-distinct.
  - 3-LTSS $\subset$ 1-1 ND
  - precedence languages $\subset$ 1-1 ND
  - all but 2 attested stress patterns $\subset$ 1-1 ND

(Heinz 2007)

LRI: Language-theoretic Characterization

- LRI languages are defined as the intersection of two classes of languages.

1. $\{L : \text{whenever } u, v \in L, T_L(u) = T_L(v)\}$
2. $\{L_1 \cdot L_2^* : L_1, L_2 \in \mathcal{L}_{fin}\}$

LRI: Language-theoretic Characterization

- LRI languages are defined as the intersection of two classes of languages.

1. $\{L : \text{whenever } u, v \in L, T_L(u) = T_L(v)\}$
2. $\{L_1 \cdot L_2^* : L_1, L_2 \in \mathcal{L}_{fin}\}$

LRI: Language-theoretic Characterization

- LRI languages are defined as the intersection of two classes of languages.

1. $\{L : \text{whenever } u, v \in L, T_L(u) = T_L(v)\}$
2. $\{L_1 \cdot L_2^* : L_1, L_2 \in \mathcal{L}_{fin}\}$

LRI: Towards an Automata-theoretic Characterization

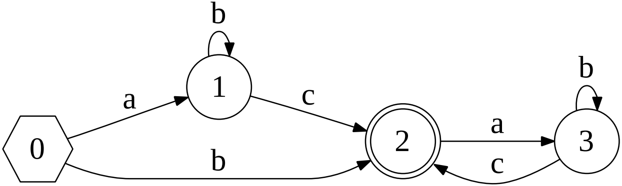
Theorem.

The class $\{L : \text{whenever } u, v \in L, T_L(u) = T_L(v)\}$ coincides with those languages recognizable by finite-state automata which are forward deterministic and have at most one final state.

LRI: Towards an Automata-theoretic Characterization

Theorem.

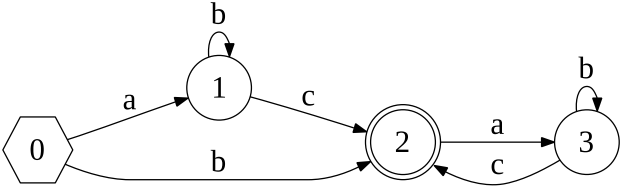
The class $\{L : \text{whenever } u, v \in L, T_L(u) = T_L(v)\}$ coincides with those languages recognizable by finite-state automata which are forward deterministic and have at most one final state.



LRI: Towards an Automata-theoretic Characterization

Theorem.

The class $\{L : \text{whenever } u, v \in L, T_L(u) = T_L(v)\}$ coincides with those languages recognizable by finite-state automata which are forward deterministic and have at most one final state.



These languages I call 1-final deterministic

LRI: Automata-theoretic Characterization

Theorem.

A language L is **left-to-right iterative** iff

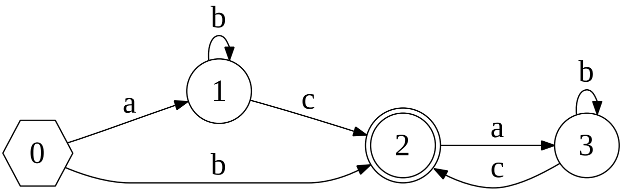
1. The tail-canonical acceptor $A_T(L)$ is 1-final-deterministic, and
2. if L is infinite, then every loop in $A_T(L)$ passes through the final state.

LRI: Automata-theoretic Characterization

Theorem.

A language L is **left-to-right iterative** iff

1. The tail-canonical acceptor $A_T(L)$ is 1-final-deterministic, and
2. if L is infinite, then every loop in $A_T(L)$ passes through the final state.

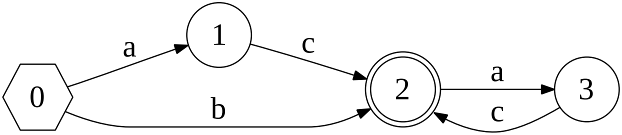


LRI: Automata-theoretic Characterization

Theorem.

A language L is **left-to-right iterative** iff

1. The tail-canonical acceptor $A_T(L)$ is 1-final-deterministic, and
2. if L is infinite, then every loop in $A_T(L)$ passes through the final state.



Consequences for Inference

1. Given $ab, abcd$, we can infer $ab(cd)^* \subseteq L$

$\Rightarrow$ Generally, given $u, uv \in L$, we infer $uv^* \subseteq L$.

Consequences for Inference

1. Given $ab, abcd$, we can infer $ab(cd)^* \subseteq L$

$\Rightarrow$ Generally, given $u, uv \in L$, we infer $uv^* \subseteq L$.

Consequences for Inference

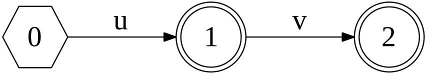
1. Given $ab, abcd$, we can infer $ab(cd)^* \subseteq L$

$\Rightarrow$ Generally, given $u, uv \in L$, we infer $uv^* \subseteq L$.

Consequences for Inference

1. Given $ab, abcd$, we can infer $ab(cd)^* \subseteq L$

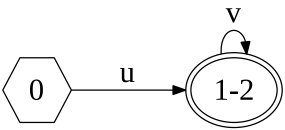
⇒ Generally, given $u, uv \in L$, we infer $uv^* \subseteq L$.



Consequences for Inference

1. Given $ab, abcd$, we can infer $ab(cd)^* \subseteq L$

$\Rightarrow$ Generally, given $u, uv \in L$, we infer $uv^* \subseteq L$.



Learning LRI

$$\boxed{\phi(S) = PT(S)/\pi_{final}}$$

1. Merge final states in the prefix tree
2. Merge states to eliminate forward non-determinism

⇒ This last step is not required — it does not change the language of the machine

Learning LRI

$$\boxed{\phi(S) = PT(S)/\pi_{final}}$$

1. Merge final states in the prefix tree
2. Merge states to eliminate forward non-determinism

⇒ This last step is not required — it does not change the language of the machine

Learning LRI

$$\boxed{\phi(S) = PT(S)/\pi_{final}}$$

1. Merge final states in the prefix tree
2. Merge states to eliminate forward non-determinism

⇒ This last step is not required — it does not change the language of the machine

Illustration of Learning LRI

Sample = { a, b, abcd, bbcb}

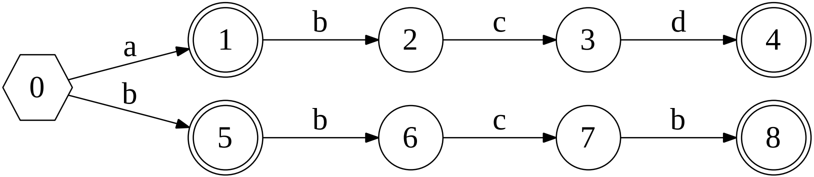


Illustration of Learning LRI

Sample = { a, b, abcd, bbcb}

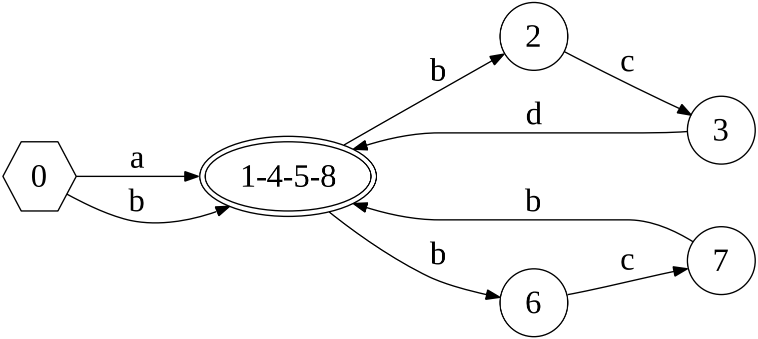


Illustration of Learning LRI

Sample = { a, b, abcd, bbcb}

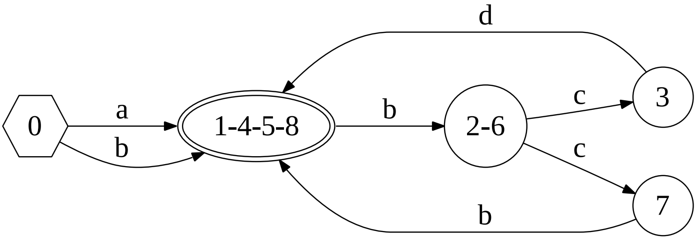


Illustration of Learning LRI

Sample = { a, b, abcd, bbcb}

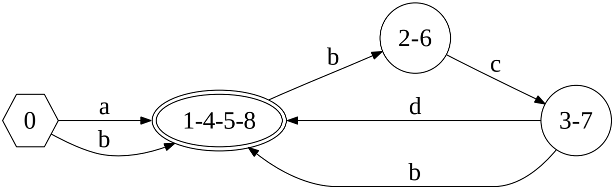
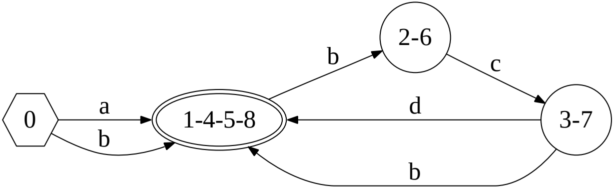


Illustration of Learning LRI

Sample = { a, b, abcd, bbcb}



- The algorithm differs only from ZR (Angluin 1982) in that states are **not** merged to remove reverse non-determinism!

Learning Results for LRI

Theorem. Every $L \in LRI$ has a characterstic sample. Since $L = L_1 \cdot L_2^*$ where $L_1, L_2 \in \mathcal{L}_{fin}$, such a sample is

$$L_1 \cup L_1 \cdot L_2$$

Theorem. $L{=}L(PT(S)/\pi_{final})$ is the smallest language in LRI which includes S.

Theorem. The learner ϕ identifies LRI in the limit from positive data.

Relation to other classes

- LRI is incomparable with ZR . . .
- and incomparable with LTSS, LT, . . .

  i.e. it crosscuts the Subregular Hierarchy
- Unknown if it is function-distinguishable (Fernau 2003)

Relation to other classes

- LRI is incomparable with ZR . . .
- and incomparable with LTSS, LT, . . .

  i.e. it crosscuts the Subregular Hierarchy
- Unknown if it is function-distinguishable (Fernau 2003)

Relation to other classes

- LRI is incomparable with ZR . . .
- and incomparable with LTSS, LT, . . .

  i.e. it crosscuts the Subregular Hierarchy
- Unknown if it is function-distinguishable (Fernau 2003)

RLI: Language-theoretic Characterization

- Languages in RLI are the **reverse** of languages in LRI.
- They are those languages recognized by FSAs whose
  - head-canonical acceptors have at most one start state
  - all loops pass through the start state
- RLI can be learned by a learner which merges start states in the suffix tree of the sample.

RLI: Language-theoretic Characterization

- Languages in RLI are the **reverse** of languages in LRI.
- They are those languages recognized by FSAs whose
  - head-canonical acceptors have at most one start state
  - all loops pass through the start state
- RLI can be learned by a learner which merges start states in the suffix tree of the sample.

RLI: Language-theoretic Characterization

- Languages in RLI are the **reverse** of languages in LRI.
- They are those languages recognized by FSAs whose
  - head-canonical acceptors have at most one start state
  - all loops pass through the start state
- RLI can be learned by a learner which merges start states in the suffix tree of the sample.

RLI: Language-theoretic Characterization

- Languages in RLI are the **reverse** of languages in LRI.
- They are those languages recognized by FSAs whose
  - head-canonical acceptors have at most one start state
  - all loops pass through the start state
- RLI can be learned by a learner which merges start states in the suffix tree of the sample.

RLI: Language-theoretic Characterization

- Languages in RLI are the **reverse** of languages in LRI.
- They are those languages recognized by FSAs whose
  - head-canonical acceptors have at most one start state
  - all loops pass through the start state
- RLI can be learned by a learner which merges start states in the suffix tree of the sample.

State-merging: Algorithm IM1

- Begin with a structured representation *PT* of the sample
- Use an equivalence relation to determine which states to merge
- The equivalence relation is determined by a function f

$$p \sim q \text{ iff } f(p) = f(q)$$

- I.e. given sample S, compute

$$PT(S)/\pi_f$$

(Muggleton 1990)

State-merging: Algorithm IM1

- Begin with a structured representation *PT* of the sample
- Use an equivalence relation to determine which states to merge
- The equivalence relation is determined by a function f

$$p \sim q \text{ iff } f(p) = f(q)$$

- I.e. given sample S, compute

$$PT(S)/\pi_f$$

(Muggleton 1990)

State-merging: Algorithm IM1

- Begin with a structured representation *PT* of the sample
- Use an equivalence relation to determine which states to merge
- The equivalence relation is determined by a function *f*

$$p \sim q \text{ iff } f(p) = f(q)$$

- I.e. given sample *S*, compute

$$PT(S)/\pi_f$$

(Muggleton 1990)

State-merging: Algorithm IM1

- Begin with a structured representation PT of the sample
- Use an equivalence relation to determine which states to merge
- The equivalence relation is determined by a function f

$$p \sim q \text{ iff } f(p) = f(q)$$

- I.e. given sample S, compute

$$PT(S)/\pi_f$$

(Muggleton 1990)

State-merging: Algorithm IM1

- Begin with a structured representation M of the sample
- Use an equivalence relation to determine which states to merge
- The equivalence relation is determined by a function f

$$p \sim q \text{ iff } f(p) = f(q)$$

- I.e. given sample S, compute

$$M(S)/\pi_f$$

(Muggleton 1990)

Choices of M and f

- Choice of M:
  - Prefix Tree
  - Suffix Tree
- Choice of f
  - same incoming k-paths $I_k(q)$
  - same outgoing k-paths $O_k(q)$
  - final states $final(q)$
  - nonfinal states $nonfinal(q)$
  - start states $start(q)$
  - nonstart states $nonstart(q)$

Choices of M and f

- Choice of M:
  1. Prefix Tree
  2. Suffix Tree
- Choice of f
  1. same incoming k-paths $I_k(q)$
  2. same outgoing k-paths $O_k(q)$
  3. final states $final(q)$
  4. nonfinal states $nonfinal(q)$
  5. start states $start(q)$
  6. nonstart states $nonstart(q)$

Choices of M and f

- Choice of M:
  1. Prefix Tree
  2. Suffix Tree
- Choice of f
  1. same incoming k-paths $I_k(q)$
  2. same outgoing k-paths $O_k(q)$
  3. final states $final(q)$
  4. nonfinal states $nonfinal(q)$
  5. start states $start(q)$
  6. nonstart states $nonstart(q)$

Choices of M and f

- Choice of M:
  1. Prefix Tree
  2. Suffix Tree
- Choice of f
  1. same incoming k-paths $I_k(q)$
  2. same outgoing k-paths $O_k(q)$
  3. final states $final(q)$
  4. nonfinal states $nonfinal(q)$
  5. start states $start(q)$
  6. nonstart states $nonstart(q)$

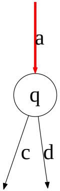
Choices of M and f

- Choice of M:
  1. Prefix Tree
  2. Suffix Tree
- Choice of f
  1. same incoming k-paths $I_k(q)$
  2. same outgoing k-paths $O_k(q)$
  3. final states $final(q)$
  4. nonfinal states $nonfinal(q)$
  5. start states $start(q)$
  6. nonstart states $nonstart(q)$

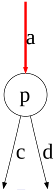
Choices of M and f

- Choice of M:
  1. Prefix Tree
  2. Suffix Tree
- Choice of f
  1. same incoming k-paths $I_k(q)$
  2. same outgoing k-paths $O_k(q)$
  3. final states $final(q)$
  4. nonfinal states $nonfinal(q)$
  5. start states $start(q)$
  6. nonstart states $nonstart(q)$

Choices of M and f

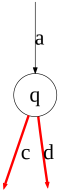


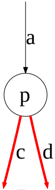
- Choice of M:
  1. Prefix Tree
  2. Suffix Tree
- Choice of f
  1. same incoming k-paths $I_k(q)$
  2. same outgoing k-paths $O_k(q)$
  3. final states $final(q)$
  4. nonfinal states $nonfinal(q)$
  5. start states $start(q)$
  6. nonstart states $nonstart(q)$



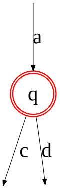
Choices of M and f

- Choice of M:
  1. Prefix Tree
  2. Suffix Tree
- Choice of f
  1. same incoming k-paths $I_k(q)$
  2. same outgoing k-paths $O_k(q)$
  3. final states $final(q)$
  4. nonfinal states $nonfinal(q)$
  5. start states $start(q)$
  6. nonstart states $nonstart(q)$

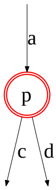




Choices of M and f

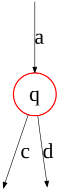


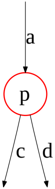
- Choice of M:
  1. Prefix Tree
  2. Suffix Tree
- Choice of f
  1. same incoming k-paths $I_k(q)$
  2. same outgoing k-paths $O_k(q)$
  3. final states $final(q)$
  4. nonfinal states $nonfinal(q)$
  5. start states $start(q)$
  6. nonstart states $nonstart(q)$



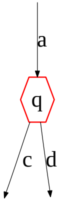
Choices of M and f

- Choice of M:
  1. Prefix Tree
  2. Suffix Tree
- Choice of f
  1. same incoming k-paths $I_k(q)$
  2. same outgoing k-paths $O_k(q)$
  3. final states $final(q)$
  4. nonfinal states $nonfinal(q)$
  5. start states $start(q)$
  6. nonstart states $nonstart(q)$

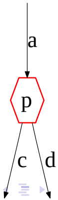




Choices of M and f

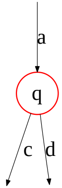


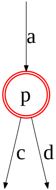
- Choice of M:
  1. Prefix Tree
  2. Suffix Tree
- Choice of f
  1. same incoming k-paths $I_k(q)$
  2. same outgoing k-paths $O_k(q)$
  3. final states $final(q)$
  4. nonfinal states $nonfinal(q)$
  5. start states $start(q)$
  6. nonstart states $nonstart(q)$



Choices of M and f

- Choice of M:
  1. Prefix Tree
  2. Suffix Tree
- Choice of f
  1. same incoming k-paths $I_k(q)$
  2. same outgoing k-paths $O_k(q)$
  3. final states $final(q)$
  4. nonfinal states $nonfinal(q)$
  5. start states $start(q)$
  6. nonstart states $nonstart(q)$





Choices of M and f

- Choice of M:
  1. Prefix Tree
  2. Suffix Tree
- Choice of f
  1. same incoming k-paths $I_k(q)$
  2. same outgoing k-paths $O_k(q)$
  3. final states $final(q)$
  4. nonfinal states $nonfinal(q)$
  5. start states $start(q)$
  6. nonstart states $nonstart(q)$

Summary of known classes obtainable in this way

(Garcia et. al 1990)

f	$PT(S)/\pi_f$	$ST(S)/\pi_f$
I_k	$(k+1)$ LTSS	?
O_k	?	$(k+1)$ LTSS
final		$\mathcal{L}_{fin}$
start	$\mathcal{L}_{fin}$	
nonfinal	?	$\{L_1^* \cdot L_2 : L_1, L_2 \subseteq \Sigma^1\}$
nonstart	$\{L_1 \cdot L_2^* : L_1, L_2 \subseteq \Sigma^1\}$	?

Summary of known classes obtainable in this way

f	$PT(S)/\pi_f$	$ST(S)/\pi_f$
I_k	$(k+1)$ LTSS	?
O_k	?	$(k+1)$ LTSS
final	?	$\mathcal{L}_{fin}$
start	$\mathcal{L}_{fin}$	?
nonfinal	?	$\{L_1^* \cdot L_2 : L_1, L_2 \subseteq \Sigma^1\}$
nonstart	$\{L_1 \cdot L_2^* : L_1, L_2 \subseteq \Sigma^1\}$	?

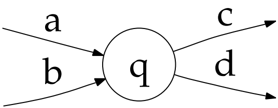
Summary of known classes obtainable in this way

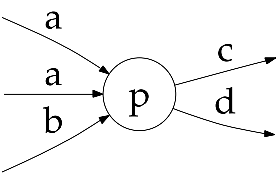
f	$PT(S)/\pi_f$	$ST(S)/\pi_f$
I_k	$(k+1)$ LTSS	?
O_k	?	$(k+1)$ LTSS
final	*LRI*	$\mathcal{L}_{fin}$
start	$\mathcal{L}_{fin}$	*RLI*
nonfinal	?	$\{L_1^* \cdot L_2 : L_1, L_2 \subseteq \Sigma^1\}$
nonstart	$\{L_1 \cdot L_2^* : L_1, L_2 \subseteq \Sigma^1\}$	?

Summary of known classes obtainable in this way

f	$PT(S)/\pi_f$	$ST(S)/\pi_f$
I_k	$(k+1)$ LTSS	?
O_k	?	$(k+1)$ LTSS
final	*LRI*	$\mathcal{L}_{fin}$
start	$\mathcal{L}_{fin}$	*RLI*
nonfinal	?	$\{L_1^* \cdot L_2 : L_1, L_2 \subseteq \Sigma^1\}$
nonstart	$\{L_1 \cdot L_2^* : L_1, L_2 \subseteq \Sigma^1\}$	?

Neighborhood-distinctness





- The neighborhood of state is determined by the function:

$$nd_j^k(q) = (I_j(q), O_k(q), [\mathbf{q} \in \mathbf{F}], [\mathbf{q} \in \mathbf{I}])$$

- Neighborhood-distinct languages are those recognized by FSAs where distinct states have distinct neighborhoods.
- But a language-theoretic characterization is missing.

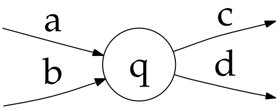
Neighborhood-distinctness

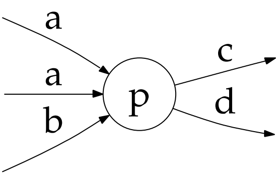
- The neighborhood of state is determined by the function:

$$nd_j^k(q) = (I_j(q), O_k(q), [\mathbf{q} \in \mathbf{F}], [\mathbf{q} \in \mathbf{I}])$$

- Neighborhood-distinct languages are those recognized by FSAs where distinct states have distinct neighborhoods.
- But a language-theoretic characterization is missing.

Neighborhood-distinctness





- The neighborhood of state is determined by the function:

$$nd_j^k(q) = (I_j(q), O_k(q), [\mathbf{q} \in \mathbf{F}], [\mathbf{q} \in \mathbf{I}])$$

- Neighborhood-distinct languages are those recognized by FSAs where distinct states have distinct neighborhoods.
- But a language-theoretic characterization is missing.

Strategy

If we understand the parts, we understand the whole.

- The neighborhood is a boolean composition of the simpler properties mentioned earlier

$$nd_j^k(q) = (I_j(q), O_k(q), [\mathbf{q} \in \mathbf{F}], [\mathbf{q} \in \mathbf{I}])$$

Strategy

If we understand the parts, we understand the whole.

- The neighborhood is a boolean composition of the simpler properties mentioned earlier

$$nd_j^k(q) = (I_j(q), O_k(q), [\mathbf{q} \in \mathbf{F}], [\mathbf{q} \in \mathbf{I}])$$

Strategy

If we understand the parts, we understand the whole.

- The neighborhood is a boolean composition of the simpler properties mentioned earlier

$$nd_j^k(q) = (I_j(q), \color{red}{O_k(q)}, [\mathbf{q} \in \mathbf{F}], [\mathbf{q} \in \mathbf{I}])$$

Strategy

If we understand the parts, we understand the whole.

- The neighborhood is a boolean composition of the simpler properties mentioned earlier

$$nd_j^k(q) = (I_j(q), O_k(q), [\mathbf{q} \in \mathbf{F}], [\mathbf{q} \in \mathbf{I}])$$

Strategy

If we understand the parts, we understand the whole.

- The neighborhood is a boolean composition of the simpler properties mentioned earlier

$$nd_j^k(q) = (I_j(q), O_k(q), [\mathbf{q} \in \mathbf{F}], [\mathbf{q} \in \mathbf{I}])$$

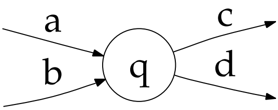
Strategy

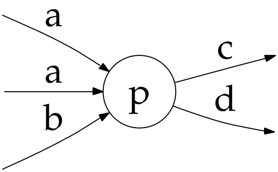
- The neighborhood is a boolean composition of the simpler properties mentioned earlier

$$nd_j^k(q) = (I_j(q), O_k(q), [\mathbf{q} \in \mathbf{F}], [\mathbf{q} \in \mathbf{I}])$$

- $final(q)$ (which helps return LRI) is part of the boolean composition of $[\mathbf{q} \in \mathbf{F}]$
- $start(q)$ (which helps return RLI) is part of the boolean composition of $[\mathbf{q} \in \mathbf{I}]$

Strategy



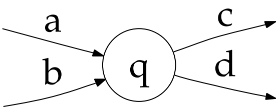


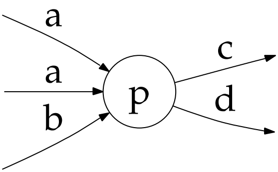
- The neighborhood is a boolean composition of the simpler properties mentioned earlier

$$nd_j^k(q) = (I_j(q), O_k(q), [\mathbf{q} \in \mathbf{F}], [\mathbf{q} \in \mathbf{I}])$$

- $final(q)$ (which helps return LRI) is part of the boolean composition of $[\mathbf{q} \in \mathbf{F}]$
- $start(q)$ (which helps return RLI) is part of the boolean composition of $[\mathbf{q} \in \mathbf{I}]$

Strategy





- The neighborhood is a boolean composition of the simpler properties mentioned earlier

$$nd_j^k(q) = (I_j(q), O_k(q), [\mathbf{q} \in \mathbf{F}], [\mathbf{q} \in \mathbf{I}])$$

- $\mathit{final}(q)$ (which helps return LRI) is part of the boolean composition of $[\mathbf{q} \in \mathbf{F}]$
- $\mathit{start}(q)$ (which helps return RLI) is part of the boolean composition of $[\mathbf{q} \in \mathbf{I}]$

Summary of known classes obtainable in this way

f	$PT(S)/\pi_f$	$ST(S)/\pi_f$
I_k	$(k+1)$ LTSS	?
O_k	?	$(k+1)$ LTSS
final	*LRI*	$\mathcal{L}_{fin}$
start	$\mathcal{L}_{fin}$	*RLI*
nonfinal	?	$\{L_1^* \cdot L_2 : L_1, L_2 \subseteq \Sigma^1\}$
nonstart	$\{L_1 \cdot L_2^* : L_1, L_2 \subseteq \Sigma^1\}$	?
...		

Conclusion

- LRI (RLI) languages are infinite subclasses of the regular languages that
  1. are obtained by merging final (start) states in prefix (suffix) trees
  2. are cousins of zero-reversible languages
  3. help reveal the algebra underlying state-merging algorithms and the reverse operator
- Phonotactic patterns are regular and it is an open question which of their properties are sufficient or necessary for learning
- The neighborhood-distinct hypothesis is one proposal
- The LRI and RLI languages are a small but necessary step towards a better understanding of this proposal

Conclusion

- LRI (RLI) languages are infinite subclasses of the regular languages that
  1. are obtained by merging final (start) states in prefix (suffix) trees
  2. are cousins of zero-reversible languages
  3. help reveal the algebra underlying state-merging algorithms and the reverse operator
- Phonotactic patterns are regular and it is an open question which of their properties are sufficient or necessary for learning
- The neighborhood-distinct hypothesis is one proposal
- The LRI and RLI languages are a small but necessary step towards a better understanding of this proposal

Conclusion

- LRI (RLI) languages are infinite subclasses of the regular languages that
  1. are obtained by merging final (start) states in prefix (suffix) trees
  2. are cousins of zero-reversible languages
  3. help reveal the algebra underlying state-merging algorithms and the reverse operator
- Phonotactic patterns are regular and it is an open question which of their properties are sufficient or necessary for learning
- The neighborhood-distinct hypothesis is one proposal
- The LRI and RLI languages are a small but necessary step towards a better understanding of this proposal

Conclusion

- LRI (RLI) languages are infinite subclasses of the regular languages that
  1. are obtained by merging final (start) states in prefix (suffix) trees
  2. are cousins of zero-reversible languages
  3. help reveal the algebra underlying state-merging algorithms and the reverse operator
- Phonotactic patterns are regular and it is an open question which of their properties are sufficient or necessary for learning
- The neighborhood-distinct hypothesis is one proposal
- The LRI and RLI languages are a small but necessary step towards a better understanding of this proposal

Conclusion

- LRI (RLI) languages are infinite subclasses of the regular languages that
  1. are obtained by merging final (start) states in prefix (suffix) trees
  2. are cousins of zero-reversible languages
  3. help reveal the algebra underlying state-merging algorithms and the reverse operator
- Phonotactic patterns are regular and it is an open question which of their properties are sufficient or necessary for learning
- The neighborhood-distinct hypothesis is one proposal
- The LRI and RLI languages are a small but necessary step towards a better understanding of this proposal

Conclusion

- LRI (RLI) languages are infinite subclasses of the regular languages that
  1. are obtained by merging final (start) states in prefix (suffix) trees
  2. are cousins of zero-reversible languages
  3. help reveal the algebra underlying state-merging algorithms and the reverse operator
- Phonotactic patterns are regular and it is an open question which of their properties are sufficient or necessary for learning
- The neighborhood-distinct hypothesis is one proposal
- The LRI and RLI languages are a small but necessary step towards a better understanding of this proposal

Conclusion

- LRI (RLI) languages are infinite subclasses of the regular languages that
  1. are obtained by merging final (start) states in prefix (suffix) trees
  2. are cousins of zero-reversible languages
  3. help reveal the algebra underlying state-merging algorithms and the reverse operator
- Phonotactic patterns are regular and it is an open question which of their properties are sufficient or necessary for learning
- The neighborhood-distinct hypothesis is one proposal
- The LRI and RLI languages are a small but necessary step towards a better understanding of this proposal

RLI: Language-theoretic Characterization

- LRI languages are defined as the intersection of two classes of languages.

1. $\{L : \text{whenever } u, v \in L, H_L(u) = H_L(v)\}$
2. $\{L_1^* \cdot L_2 : L_1, L_2 \in \mathcal{L}_{fin}\}$

RLI: Language-theoretic Characterization

- LRI languages are defined as the intersection of two classes of languages.

1. $\{L : \text{whenever } u, v \in L, H_L(u) = H_L(v)\}$
2. $\{L_1^* \cdot L_2 : L_1, L_2 \in \mathcal{L}_{fin}\}$

RLI: Language-theoretic Characterization

- LRI languages are defined as the intersection of two classes of languages.

1. $\{L : \text{whenever } u, v \in L, H_L(u) = H_L(v)\}$
2. $\{L_1^* \cdot L_2 : L_1, L_2 \in \mathcal{L}_{fin}\}$

What is $H_L(u)$?

- $H_L(u) = \{v : vu \in L\}$
- It used to define **head-canonical acceptors** which are the smallest reverse-deterministic acceptor for a regular language.

What is $H_L(u)$?

- $H_L(u) = \{v : vu \in L\}$
- It used to define **head-canonical acceptors** which are the smallest reverse-deterministic acceptor for a regular language.

Learning RLI

- Merge **start** states in the **suffix** tree

$$\phi(S_t) = ST(S_t)/\pi_{start}$$

- Merge states to remove reverse non-determinism

Learning RLI

- Merge **start** states in the **suffix** tree

$$\phi(S_t) = ST(S_t)/\pi_{start}$$

- Merge states to remove reverse non-determinism

Illustration of Learning RLI

Sample = { a, b, dcba, bcbb}

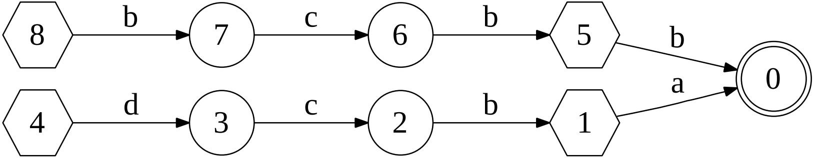


Illustration of Learning RLI

Sample = { a, b, dcba, bcbb}

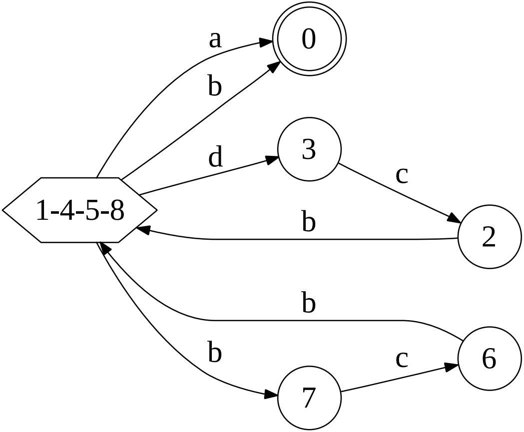


Illustration of Learning RLI

Sample = { a, b, dcba, bcbb}

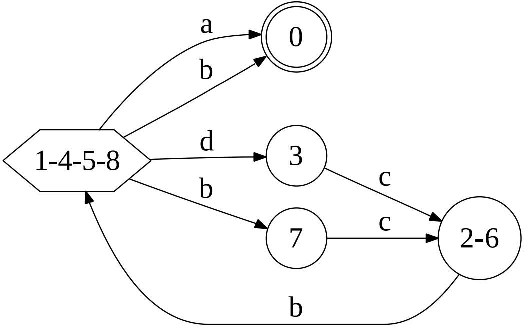
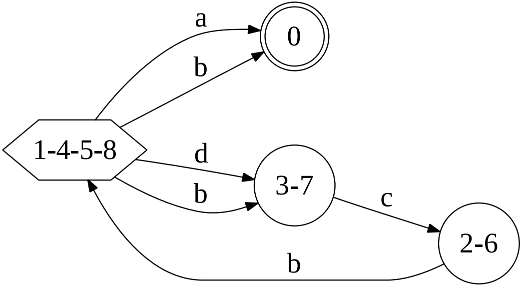


Illustration of Learning RLI

Sample = { a, b, dcba, bcbb}



Asymmetric Harmony

- Sarcee is like Navajo except the pattern is asymmetric: [ʃ] may precede [s] in a word, but [s] cannot precede [ʃ]
  - Includes { sotos, ʃotoʃ, ʃotos, . . . }
  - Excludes { sotoʃ, . . . }
- This pattern is Noncounting.

(Hansson 2001)

Asymmetric Harmony

- Sarcee is like Navajo except the pattern is asymmetric: [ʃ] may precede [s] in a word, but [s] cannot precede [ʃ]
  - Includes { sotos, ʃotoʃ, ʃotos, . . . }
  - Excludes { sotoʃ, . . . }
- This pattern is Noncounting.

(Hansson 2001)

Asymmetric Harmony

- Sarcee is like Navajo except the pattern is asymmetric: [ʃ] may precede [s] in a word, but [s] cannot precede [ʃ]
  - Includes { sotos, ʃotoʃ, ʃotos, . . . }
  - Excludes { sotoʃ, . . . }
- This pattern is Noncounting.

(Hansson 2001)

Asymmetric Harmony

- Sarcee is like Navajo except the pattern is asymmetric: [ʃ] may precede [s] in a word, but [s] cannot precede [ʃ]
  - Includes { sotos, ʃotoʃ, ʃotos, . . . }
  - Excludes { sotoʃ, . . . }
- This pattern is Noncounting.

(Hansson 2001)

Bounded Stress Patterns

- Secondary stress falls on nonfinal odd syllables (counting from left)
- Primary stress falls on the initial syllable

a.	$\acute{\sigma}\ \sigma$	páɳa	'earth'
b.	$\acute{\sigma}\ \sigma\ \sigma$	tʲúʈaya	'many'
c.	$\acute{\sigma}\ \sigma\ \grave{\sigma}\ \sigma$	máɭawàna	'through from behind'
d.	$\acute{\sigma}\ \sigma\ \grave{\sigma}\ \sigma\ \sigma$	púɭiŋkàlatʲu	'we (sat) on the hill'
e.	$\acute{\sigma}\ \sigma\ \grave{\sigma}\ \sigma\ \grave{\sigma}\ \sigma$	tʲámulìmpatʲùŋku	'our relation'
f.	$\acute{\sigma}\ \sigma\ \grave{\sigma}\ \sigma\ \grave{\sigma}\ \sigma\ \sigma$	ʈíɭirìŋulàmpatʲu	'the fire for our benefit flared up'
g.	$\acute{\sigma}\ \sigma\ \grave{\sigma}\ \sigma\ \grave{\sigma}\ \sigma\ \grave{\sigma}\ \sigma$	kúranʲùlulìmpatʲùɻa	'the first one who is our relation'
h.	$\acute{\sigma}\ \sigma\ \grave{\sigma}\ \sigma\ \grave{\sigma}\ \sigma\ \grave{\sigma}\ \sigma\ \sigma$	yúmaɻìŋkamàratʲùɻaka	'because of mother-in-law'

Hayes (1995:62) citing Hansen and Hansen (1969:163)

Bounded Stress Patterns

- Secondary stress falls on nonfinal odd syllables (counting from left)
- Primary stress falls on the initial syllable

a.	**σ́** σ	páɳa	‘earth’
b.	**σ́** σ σ	t^{j}úʈaya	‘many’
c.	**σ́** σ **σ̀** σ	máɭawàna	‘through from behind’
d.	**σ́** σ **σ̀** σ σ	púɭiŋkàlat^{j}u	‘we (sat) on the hill’
e.	**σ́** σ **σ̀** σ **σ̀** σ	t^{j}ámulìmpat^{j}ùŋku	‘our relation’
f.	**σ́** σ **σ̀** σ **σ̀** σ σ	ʈíɭirìŋulàmpat^{j}u	‘the fire for our benefit flared up’
g.	**σ́** σ **σ̀** σ **σ̀** σ **σ̀** σ	kúran^{j}ùlulìmpat^{j}ùɻa	‘the first one who is our relation’
h.	**σ́** σ **σ̀** σ **σ̀** σ **σ̀** σ σ	yúmaɻìŋkamàrat^{j}ùɻaka	‘because of mother-in-law’

Hayes (1995:62) citing Hansen and Hansen (1969:163)

Unbounded Stress Patterns

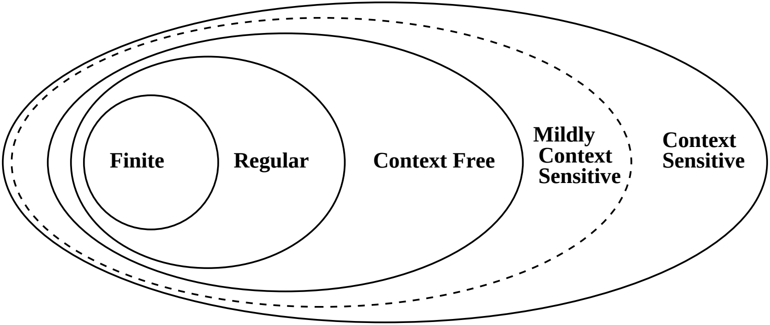
KwaKwala: Leftmost Heavy Otherwise Rightmost

- Stress the heavy syllable closest to the left edge. If there is no heavy syllable, stress the rightmost syllable.

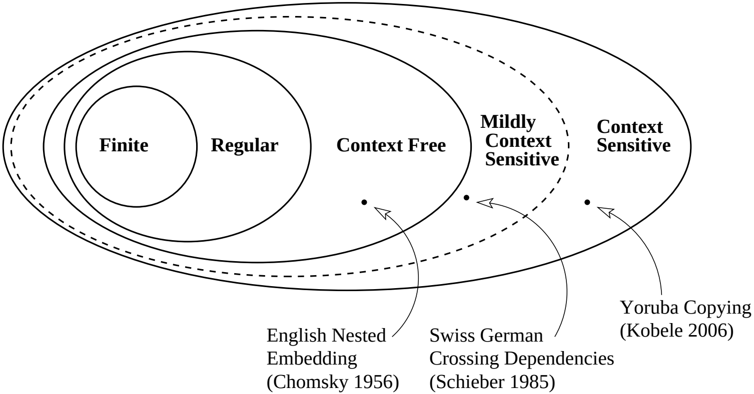
a.	**Ḣ́** H H	d.	L **Ĺ**
b.	L L **H́** L L	e.	L L **Ĺ**
c.	L L L **H́**	f.	L L L **Ĺ**

Walker (2000:21) citing Zec (1994)

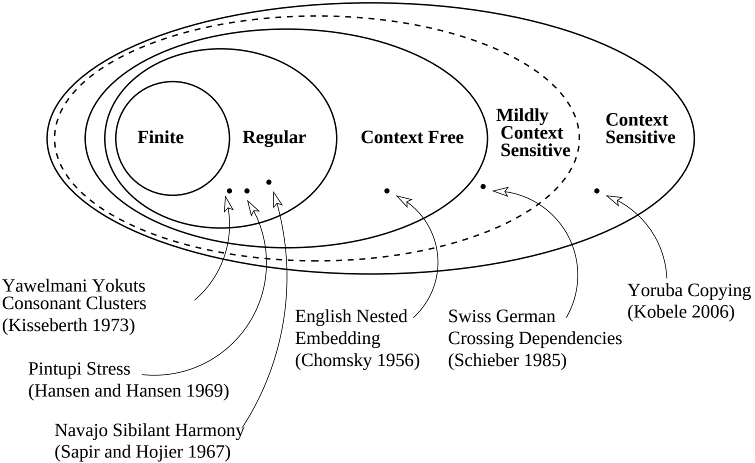
Language Patterns and the Chomsky Hierarchy



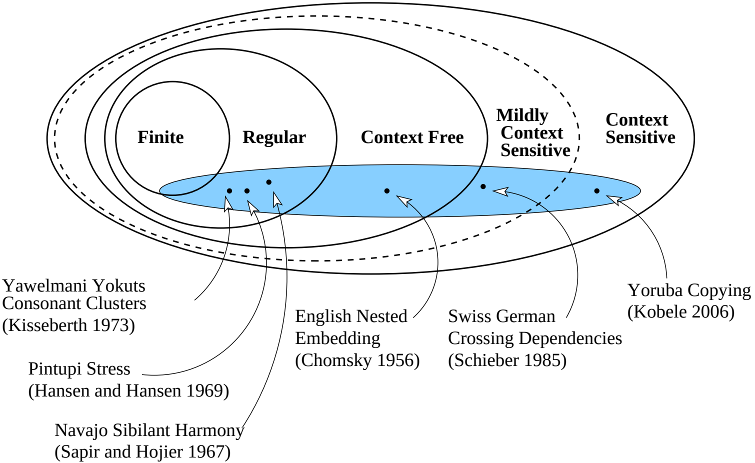
Language Patterns and the Chomsky Hierarchy



Language Patterns and the Chomsky Hierarchy



Language Patterns and the Chomsky Hierarchy



Language Patterns and the Chomsky Hierarchy