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

Jeffrey Heinz heinz@udel.edu

University of Delaware

St. Malo September 24, 2008

J. Heinz (1)

University of Delaware

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
- essentially the classes obtainable by merging final and start states in prefix and suffix trees, respectively

J. Heinz (3) Learning Left-to-Right and Right-to-Left Iterative Languages University of Delaware

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

- related algorithmically 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)

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

J. Heinz (5)

- related algorithmically to the zero-reversible languages (remove one line!) (Angluin 1982)
- 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)

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

J. Heinz (6)

- related algorithmically to the zero-reversible languages (remove one line!) (Angluin 1982)
- 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)

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

J. Heinz (7)

- related algorithmically to the zero-reversible languages (remove one line!) (Angluin 1982)
- 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
- related to a linguistic hypothesis: all phonotactic patterns are neighborhood-distinct (Heinz 2007)

J. Heinz (8)

J. Heinz (9)

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, ... }

University of Delaware

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, ... }

J. Heinz (10) Learning Left-to-Right and Right-to-Left Iterative Languages University of Delaware

J. Heinz (11)

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, ... }

J. Heinz (12)

Specific Sound Patterns

Phonotactics

■ No Triple Consonant Clusters in Yokuts:

- Includes { ab, abba, ababa, ... }
- Excludes { bbb, abbb, abbba, bbbba, ... }
- Symmetric Sibilant Harmony (Navajo):
 - Includes { sos, sotototos, ...]o],]otototo] ... }
 - Excludes { so∫, ∫otos, sototo∫, ...
- Asymmetric Sibilant Harmony (Sarcee):
 - Includes {sos, sotototos, ...]o],]otototo] ...]os,]otos, ... }
 - Excludes { so∫, soto∫, sototo∫, ...

J. Heinz (13)

Specific Sound Patterns

Phonotactics

No Triple Consonant Clusters in Yokuts:

- Includes { ab, abba, ababa, ... }
- Excludes { bbb, abbb, abbba, bbbba, ... }

Symmetric Sibilant Harmony (Navajo):

- Includes {sos, sotototos, ...]o],]otototof ... }
- Excludes { so∫, ∫otos, sototo∫, ...
- Asymmetric Sibilant Harmony (Sarcee):
 - Includes {sos, sotototos, ... ∫o∫, ∫otototo∫ ... ∫os, ∫otos, ... }
 - Excludes { so, soto, soto, ...

No Triple Consonant Clusters in Yokuts:

- Includes { ab, abba, ababa, ... }
- Excludes { bbb, abbb, abbba, bbbba, ... }
- Symmetric Sibilant Harmony (Navajo):
 - Includes {sos, sotototos, ...]o], [otototo] ... }
 - Excludes { so∫, ∫otos, sototo∫, ...
- Asymmetric Sibilant Harmony (Sarcee):
 - Includes {sos, sotototos, ...]o],]otototo] ...]os,]otos, ... }
 - Excludes { so∫, soto∫, sototo∫, ...

J. Heinz (14)

■ No Triple Consonant Clusters in Yokuts:

- Includes { ab, abba, ababa, ... }
- Excludes { bbb, abbb, abbba, bbbba, ... }

Symmetric Sibilant Harmony (Navajo):

- Includes {sos, sotototos, ... ∫o∫, ∫otototo∫ ... }
- **Excludes** { so_{j} , jotos, $sototo_{j}$, ... }
- Asymmetric Sibilant Harmony (Sarcee):
 - Includes {sos, sotototos, ...]o], [otototo] ... [os, [otos, ... }
 - Excludes { so∫, soto∫, sototo∫, ...

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

J. Heinz (15)

■ No Triple Consonant Clusters in Yokuts:

- Includes { ab, abba, ababa, ... }
- Excludes { bbb, abbb, abbba, bbbba, ... }

Symmetric Sibilant Harmony (Navajo):

- Includes {sos, sotototos, ... ∫o∫, ∫otototo∫ ... }
- Excludes { so∫, ∫otos, sototo∫, ... }
- Asymmetric Sibilant Harmony (Sarcee):
 - Includes {sos, sotototos, ...]o],]otototo] ...]os,]otos, ... }
 - Excludes { so∫, soto∫, sototo∫, ...

J. Heinz (16)

■ No Triple Consonant Clusters in Yokuts:

- Includes { ab, abba, ababa, ... }
- Excludes { bbb, abbb, abbba, bbbba, ... }
- Symmetric Sibilant Harmony (Navajo):
 - Includes {sos, sotototos, ... ∫o∫, ∫otototo∫ ... }
 - **Excludes** { sof, fotos, sototof, ... }
- Asymmetric Sibilant Harmony (Sarcee):
 - Includes {sos, sotototos, ...]o],]otototo[...]os,]otos, ... }
 - Excludes { soʃ, sotoʃ, sototoʃ, ... }

J. Heinz (17)

No Triple Consonant Clusters in Yokuts:

- Includes { ab, abba, ababa, ... }
- Excludes { bbb, abbb, abbba, ... }
- Symmetric Sibilant Harmony (Navajo):
 - Includes {sos, sotototos, ... [o], [otototo] ... }
 - Excludes { so $[, fotos, sototo], ... \}$
- Asymmetric Sibilant Harmony (Sarcee):
 - Includes {sos, sotototos, ... [o], [otototo] ... [os, [otos, ...]
 - Excludes { sof, sotof, sotof, ... }

J. Heinz (18)

No Triple Consonant Clusters in Yokuts:

- Includes { ab, abba, ababa, ... }
- Excludes { bbb, abbb, abbba, ... }
- Symmetric Sibilant Harmony (Navajo):
 - Includes {sos, sotototos, ... [o], [otototo] ... }
 - Excludes { so $[, fotos, sototo], ... \}$
- Asymmetric Sibilant Harmony (Sarcee):

J. Heinz (19)

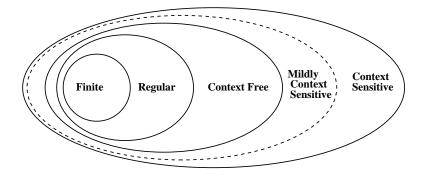
No Triple Consonant Clusters in Yokuts:

- Includes { ab, abba, ababa, ... }
- Excludes { bbb, abbb, abbba, ... }
- Symmetric Sibilant Harmony (Navajo):
 - Includes {sos, sotototos, ... [o], [otototo] ... }
 - Excludes { so $[, fotos, sototo], ... \}$
- Asymmetric Sibilant Harmony (Sarcee):
 - Includes {sos, sotototos, $\ldots \int o \int$, $\int otototo \int \ldots \int os$, $\int otos$, \ldots }
 - Excludes { so[, soto[, sototo[, ... }

J. Heinz (20)

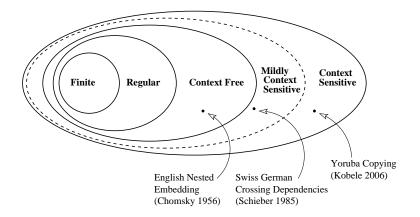
J. Heinz (21)

Language Patterns and the Chomsky Hierarchy

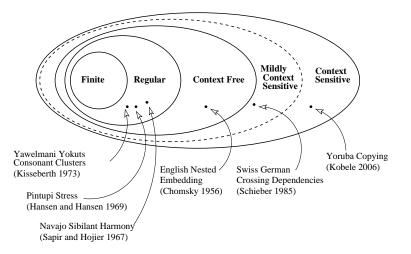


J. Heinz (22)

Language Patterns and the Chomsky Hierarchy



Language Patterns and the Chomsky Hierarchy

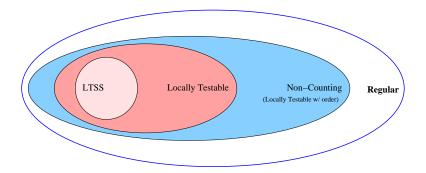


J. Heinz (23)

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

University of Delaware

The Subregular Hierarchy



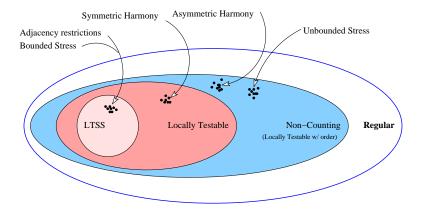
(McNaughton and Papert 1971, Pullum and Rogers 2007)

J. Heinz (24)

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

University of Delaware

The Subregular Hierarchy



(Greenberg 1978, Hansson 2001, Hayes 1995)

・ロト ・ 日 ・ ・ 回 ・

J. Heinz (25)

University of Delaware

J. Heinz (26)

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
 - Linguists' knowledge of the range of variation
 - E.g. Greenberg 1978, Hansson 2001, Hayes 1995, ...
 - Psycholinguistic evidence about the state of infants' knowledge E.g. Amerika et al. 1999, Mattyr and Amerika 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 tangenetic knowledge of the range of variation
 E.g. Psycholinguistic evidence about the state of infants' knowledge

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

Linguists' knowledge of the range of variation

E.g. 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, ...

J. Heinz (30)

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, ...

J. Heinz (31)

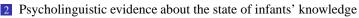
Grammatical Inference of Regular Languages is Theoretical Phonology

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, ...



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

J. Heinz (32)

Grammatical Inference of Regular Languages is Theoretical Phonology

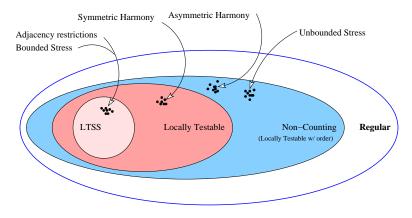
The properties of learnable subclasses of the regular languages are candidates as universal properties of sound patterns

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, ...

The Subregular Hierarchy



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

precedence languages $\subset 1-1$ ND

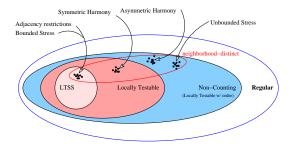
J. Heinz (33)

University of Delaware

(Heinz 2007)

University of Delaware

The Subregular Hierarchy



For small neighborhoods, they are all neighborhood-distinct.

- 3-LTSS ⊂ 1-1 ND
- precedence languages \subset 1-1 ND
- all but 2 attested stress patterns \subset 1-1 ND

J. Heinz (34)

LRI: Language-theoretic Characterization

- LRI languages are defined as the <u>intersection</u> of two classes of languages.
- 1 {L: whenever $u, v \in L, T_L(u) = T_L(v)$ } 2 { $L_1 \cdot L_2^* : L_1, L_2 \in \mathcal{L}_{fin}$ }

J. Heinz (35)

University of Delaware

J. Heinz (36)

LRI: Language-theoretic Characterization

- LRI languages are defined as the <u>intersection</u> of two classes of languages.
- $\{L: \text{ whenever } u, v \in L, T_L(u) = T_L(v) \}$

2 $\{L_1 \cdot L_2^* : L_1, L_2 \in \mathcal{L}_{fin}\}$

University of Delaware

J. Heinz (37)

LRI: Language-theoretic Characterization

- LRI languages are defined as the <u>intersection</u> of two classes of languages.
- $\{L: \text{ whenever } u, v \in L, T_L(u) = T_L(v) \}$
- 2 { $L_1 \cdot L_2^* : L_1, L_2 \in \mathcal{L}_{fin}$ }

University of Delaware

LRI: Towards an Automata-theoretic Characterization

Theorem.

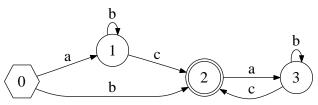
The class {*L* : whenever $u, v \in L, T_L(u) = T_L(v)$ } coincides with those languages recognizable by finite-state automata which are forward deterministic and have at most one final state.

J. Heinz (38) Learning Left-to-Right and Right-to-Left Iterative Languages

LRI: Towards an Automata-theoretic Characterization

Theorem.

The class {*L* : 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.



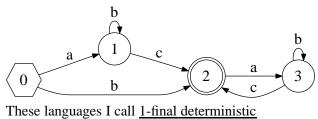
Learning Left-to-Right and Right-to-Left Iterative Languages

J. Heinz (39)

LRI: Towards an Automata-theoretic Characterization

Theorem.

The class {*L* : 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.



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

J. Heinz (40)

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.

J. Heinz (41) Learning Left-to-Right and Right-to-Left Iterative Languages

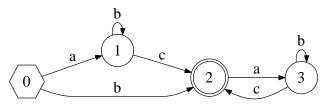
J. Heinz (42)

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.



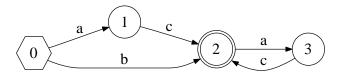
J. Heinz (43)

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 $ab(cd)^* \subseteq L$

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

J. Heinz (44) Learning Left-to-Right and Right-to-Left Iterative Languages

Consequences for Inference

- **1** Given ab, abcd, we can infer $ab(cd)^* \subseteq L$
- \Rightarrow Generally, given $u, uv \in L$, we infer $uv^* \subseteq L$.

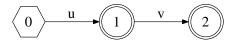
Consequences for Inference

- **1** Given ab, abcd, we can infer $ab(cd)^* \subseteq L$
- \Rightarrow Generally, given $u, uv \in L$, we infer $uv^* \subseteq L$.

J. Heinz (47)

Consequences for Inference

1 Given ab, abcd, we can infer $ab(cd)^* \subseteq L$ ⇒ Generally, given $u, uv \in L$, we infer $uv^* \subseteq L$.

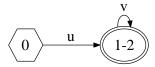




J. Heinz (48)

Consequences for Inference

I Given ab, abcd, we can infer $ab(cd)^* \subseteq L$ ⇒ Generally, given $u, uv \in L$, we infer $uv^* \subseteq L$.



University of Delaware

Learning LRI

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

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

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

- **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

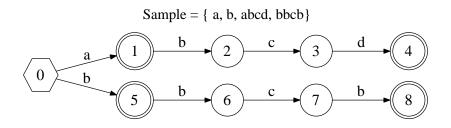
Learning LRI

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

- **1** Merge final states in the prefix tree
- 2 Merge states to eliminate forward non-determinism
- \Rightarrow This last step is not required it does not change the language of the machine

J. Heinz (52)

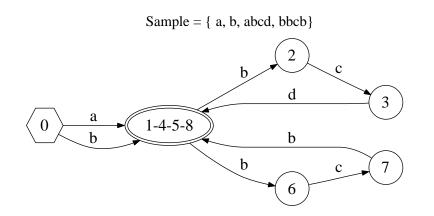
Illustration of Learning LRI



University of Delaware

J. Heinz (53)

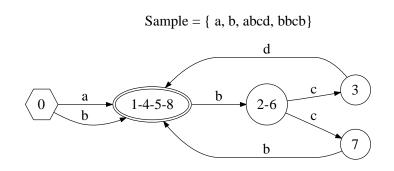
Illustration of Learning LRI



University of Delaware

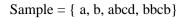
J. Heinz (54)

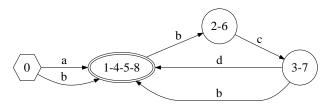
Illustration of Learning LRI



J. Heinz (55)

Illustration of Learning LRI

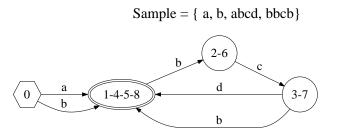




University of Delaware

University of Delaware

Illustration of Learning LRI



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

University of Delaware

Learning Results for LRI

Theorem. Every $L \in LRI$ has a characteristic 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)

J. Heinz (58) Learning Left-to-Right and Right-to-Left Iterative Languages

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)

J. Heinz (59) Learning Left-to-Right and Right-to-Left Iterative Languages

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)

J. Heinz (60) Learning Left-to-Right and Right-to-Left Iterative Languages

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 stat
 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.

J. Heinz (63)

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.

J. Heinz (64)

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.

J. Heinz (65)

Phonotactics

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*

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

■ I.e. given sample *S*, compute

 $PT(S)/\pi_f$

(Muggleton 1990)

University of Delaware

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*

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

■ I.e. given sample *S*, compute

 $PT(S)/\pi_f$

(Muggleton 1990)

University of Delaware

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

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

■ I.e. given sample *S*, compute

 $PT(S)/\pi_f$

(Muggleton 1990)

University of Delaware

J. Heinz (69)

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

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

■ I.e. given sample *S*, compute

 $PT(S)/\pi_f$

(Muggleton 1990)

J. Heinz (70)

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

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

■ I.e. given sample *S*, compute

 $M(S)/\pi_f$

(Muggleton 1990)

J. Heinz (71)

Appendices

Choices of M and f

▲口 ▶ ▲国 ▶ ▲国 ▶ ▲国 ▶ ▲日 ▶

University of Delaware

J. Heinz (72)

Choices of M and f

Choice of *M*:

- Prefix Tree
- 2 Suffix Tree

Choice of *f*

- **Solution** same incoming k-paths $I_k(q)$
- Same outgoing k-paths $O_k(q)$
- final states *final(q)*
- In nonfinal states nonfinal(q)
- 5 start states start(q)
- **nonstart states** nonstart(q)

University of Delaware

J. Heinz (73)

Choices of M and f

Choice of *M*:

- 1 Prefix Tree
- 2 Suffix Tree

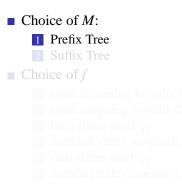
• Choice of f

- same incoming k-paths $I_k(q)$
- In same outgoing k-paths $O_k(q)$
- final states final(q)
- Inonfinal states nonfinal(q)
- 5 start states start(q)
- **nonstart states** nonstart(q)

University of Delaware

J. Heinz (74)

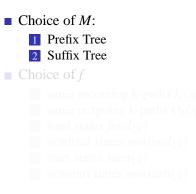
Choices of M and f



University of Delaware

J. Heinz (75)

Choices of M and f



University of Delaware

Choices of M and f

Choice of *M*:

Prefix Tree
Suffix Tree

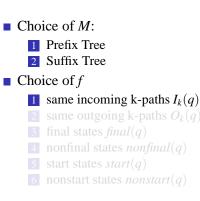
Choice of *f*same incoming k-pa
final states *final(q)*nonfinal states *nonf*start states *start(q)*nonstart states *nons*

University of Delaware

J. Heinz (76)

J. Heinz (77)

Choices of M and f





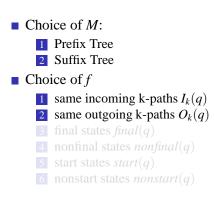
р

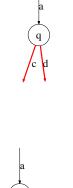
′c∖d

University of Delaware

J. Heinz (78)

Choices of M and f



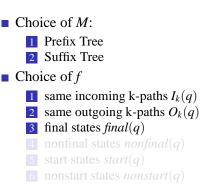


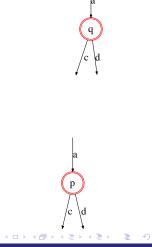
р

University of Delaware

J. Heinz (79)

Choices of M and f

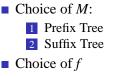




University of Delaware

J. Heinz (80)

Choices of M and 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)
- 1 nonstart states *nonstart*(q)



р

d



J. Heinz (81)

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)



a

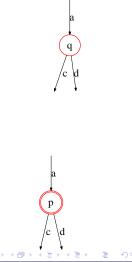
d



Choices of M and f



- 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)



University of Delaware

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

J. Heinz (82)

J. Heinz (83)

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)

J. Heinz (84)

Summary of known classes obtainable in this way

(Garcia et. al 1990)

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

University of Delaware

J. Heinz (85)

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	?	$\mathcal{L}_{\mathit{fin}}$
start	$\mathcal{L}_{\mathit{fin}}$?
nonfinal	?	$\{L_1^* \cdot L_2 : L_1, L_2 \subseteq \Sigma^1\}$
nonstart	$\{L_1 \cdot L_2^* : L_1, L_2 \subseteq \Sigma^1\}$?

J. Heinz (86)

Appendices

Summary of known classes obtainable in this way

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

University of Delaware

J. Heinz (87)

Summary of known classes obtainable in this way

I_k	(k+1) LTSS	?
O_k	?	(k+1) LTSS
final	LRI	$\mathcal{L}_{\mathit{fin}}$
start	$\mathcal{L}_{\mathit{fin}}$	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: $nd_i^k(q) = (I_j(q), O_k(q), [\mathbf{q} \in \mathbf{F}], [\mathbf{q} \in \mathbf{I}])$
- Neighborhood-distinct languages are those recognized by FSAs where distinct states have distinct neighborhoods.
- But a language-theoretic characterization is missing.

J. Heinz (88)

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

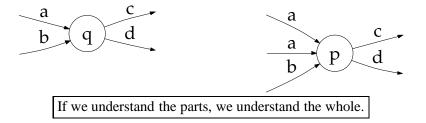


• The neighborhood of state is determined by the function:

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

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

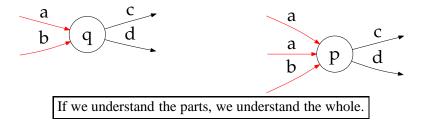
J. Heinz (91)



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

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

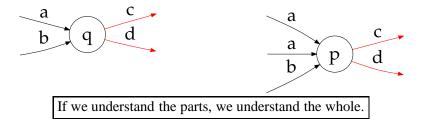
J. Heinz (92)



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

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

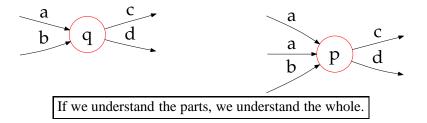
J. Heinz (93)



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

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

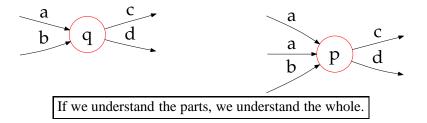
J. Heinz (94)



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

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

J. Heinz (95)



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

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



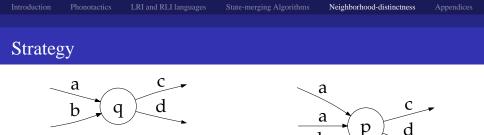
The neighborhood is a boolean composition of the simpler properties mentioned earlier

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

- *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 $[\mathbf{q} \in \mathbf{I}]$

р

h



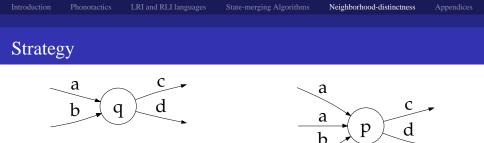
h

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 [**q** ∈ **F**]
- start(q) (which helps return RLI) is part of the boolean composition of [q ∈ I]

J. Heinz (97)



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

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

University of Delaware

- *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 [q ∈ I]

J. Heinz (98)

J. Heinz (99)

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	$\mathcal{L}_{\mathit{fin}}$
start	$\mathcal{L}_{\mathit{fin}}$	RLI
nonfinal	?	$\{L_1^* \cdot L_2 : L_1, L_2 \subseteq \Sigma^1\}$
nonstart	$\{L_1 \cdot L_2^* : L_1, L_2 \subseteq \Sigma^1\}$?

University of Delaware

 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

< 47 ▶

- 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

- 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

- 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

- 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

- 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

- 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 <u>intersection</u> of two classes of languages.
- 1 {L: whenever $u, v \in L, H_L(u) = H_L(v)$ } 2 { $L_1^* \cdot L_2 : L_1, L_2 \in \mathcal{L}_{fin}$ }

J. Heinz (107)

University of Delaware

J. Heinz (108)

RLI: Language-theoretic Characterization

- LRI languages are defined as the <u>intersection</u> of two classes of languages.
- $\blacksquare \{L: \text{ whenever } u, v \in L, H_L(u) = H_L(v)\}$
- 2 $\{L_1^* \cdot L_2 : L_1, L_2 \in \mathcal{L}_{fin}\}$

University of Delaware

J. Heinz (109)

RLI: Language-theoretic Characterization

- LRI languages are defined as the <u>intersection</u> of two classes of languages.
- $L: \text{ whenever } u, v \in L, H_L(u) = H_L(v) \}$
- 2 $\{L_1^* \cdot L_2 : L_1, L_2 \in \mathcal{L}_{fin}\}$

University of Delaware

What is $H_L(u)$?

$\blacksquare H_L(u) = \{v : vu \in L\}$

It used to define head-canonical acceptors which are the smallest reverse-deterministic acceptor for a regular language.

J. Heinz (110) Learning Left-to-Right and Right-to-Left Iterative Languages University of Delaware

What is $H_L(u)$?

- $\blacksquare H_L(u) = \{v : vu \in L\}$
- 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

J. Heinz (112) Learning Left-to-Right and Right-to-Left Iterative Languages University of Delaware

Learning RLI

Merge start states in the suffix tree

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

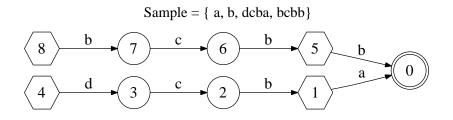
Merge states to remove reverse non-determinism

J. Heinz (113) Learning Left-to-Right and Right-to-Left Iterative Languages University of Delaware

J. Heinz (114)

Appendices

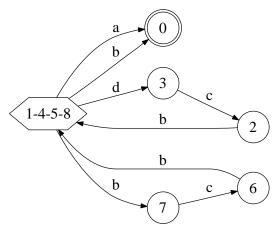
Illustration of Learning RLI



University of Delaware

Illustration of Learning RLI

Sample = $\{a, b, dcba, bcbb\}$

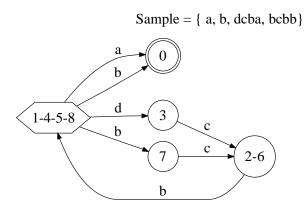


J. Heinz (115)

University of Delaware

J. Heinz (116)

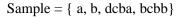
Illustration of Learning RLI

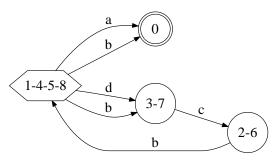


University of Delaware

J. Heinz (117)

Illustration of Learning RLI





(ロト・国ト・ヨト・ヨト・日) めんの

University of Delaware

■ Sarcee is like Navajo except the pattern is asymmetric: [ʃ] may precede [s] in a word, but [s] cannot precede [ʃ]

- Includes { sotos, ∫oto∫, ∫otos, ...]
- **Excludes** { soto \int, \dots }
- This pattern is Noncounting.

(Hansson 2001)

- Sarcee is like Navajo except the pattern is asymmetric: [ʃ] may precede [s] in a word, but [s] cannot precede [ʃ]
 - Includes { sotos, $\int oto \int$, $\int otos$, ... }
 - Excludes { soto∫, ... }
- This pattern is Noncounting.

(Hansson 2001)

- Sarcee is like Navajo except the pattern is asymmetric: [ʃ] may precede [s] in a word, but [s] cannot precede [ʃ]
 - Includes { sotos, $\int oto \int$, $\int otos$, ... }
 - Excludes { soto∫, ... }
- This pattern is Noncounting.

(Hansson 2001)

J. Heinz (120)

- Sarcee is like Navajo except the pattern is asymmetric: [ʃ] may precede [s] in a word, but [s] cannot precede [ʃ]
 - Includes { sotos, $\int oto \int$, $\int otos$, ... }
 - Excludes { soto∫, ... }
- This pattern is Noncounting.

(Hansson 2001)

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

J. Heinz (121)

Bounded Stress Patterns

Secondary stress falls on nonfinal odd syllables (counting from left)

Primary stress falls on the initial syllable

a.	$\boldsymbol{\sigma}$ σ	páŋa	'earth'
b.	ό σ σ	t ^j úťaya	'many'
c.	ό σ ờ σ	málawàna	'through from behind'
d.	ό σ ờ σ σ	pú liŋk à lat ^j u	'we (sat) on the hill'
e.	ό σ ờ σ ờ σ	t ^j ámulìmpat ^j ùŋku	'our relation'
f.	ό σ ὸ σ ὸ σ σ	tíliriŋulàmpat ^j u	'the fire for our benefit flared up'
g.	ό σ ờ σ ờ σ ờ σ	kúran ^j ùlulìmpat ^j ùĮa	'the first one who is our relation'
h.	ό σ ὸ σ ὸ σ ὸ σσ	yúmajìŋkamàrat ^j ùjaka	'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.	$\boldsymbol{\sigma}$ σ	páŋa	'earth'
b.	ό σ σ	t ^j úťaya	'many'
c.	ό σ ờ σ	málawàna	'through from behind'
d.	ό σ ờ σ σ	pú liŋk à lat ^j u	'we (sat) on the hill'
e.	ό σ ờ σ ờ σ	t ^j ámulìmpat ^j ùŋku	'our relation'
f.	ό σ δ σ δ σ σ	tíliriŋulàmpat ^j u	'the fire for our benefit flared up'
g.	ό σ ὸ σ ὸ σ ὸ σ	kúran ^j ùlulìmpat ^j ùĮa	'the first one who is our relation'
h.	ό σ ὸ σ ὸ σ ὸ σσ	yúmajìŋkamàrat ^j ùjaka	'because of mother-in-law'

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

J. Heinz (123)

University of Delaware

J. Heinz (124)

Unbounded Stress Patterns

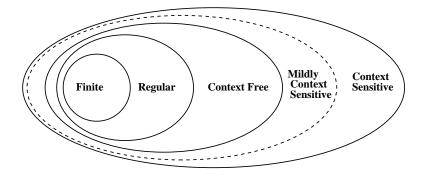
KwaKwala: Leftmost Heavy Otherwise Rightmost

- Stress the heavy syllable closest to the left edge. If there is no heavy syllable, stress the rightmost syllable.
 - a. **Ú**HH d. L**Ĺ** b. LL**Ú**LL e. LL**Ĺ** c. LLL**Ú** f. LLL**Ĺ**

Walker (2000:21) citing Zec (1994)

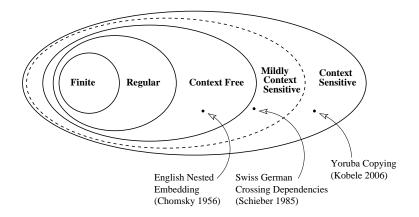
J. Heinz (125)

Language Patterns and the Chomsky Hierarchy

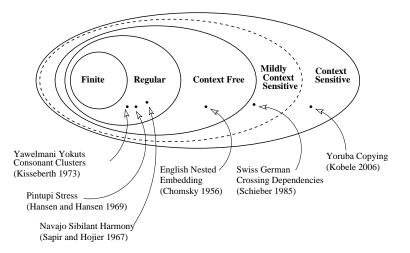


J. Heinz (126)

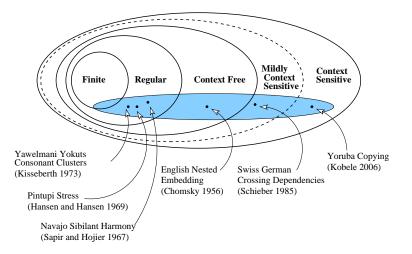
Language Patterns and the Chomsky Hierarchy



Language Patterns and the Chomsky Hierarchy



Language Patterns and the Chomsky Hierarchy

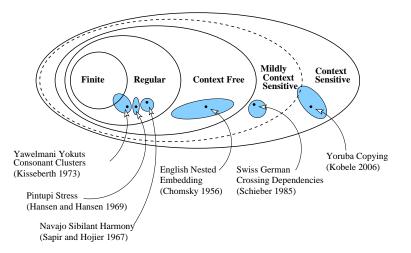


J. Heinz (128)

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

University of Delaware

Language Patterns and the Chomsky Hierarchy



J. Heinz (129)

University of Delaware