Representing and Learning Regular Sets and Functions

Jeffrey Heinz

Department of Linguistics and Cognitive Science

University of Delaware

Support from NSF#1123692 and #1035577

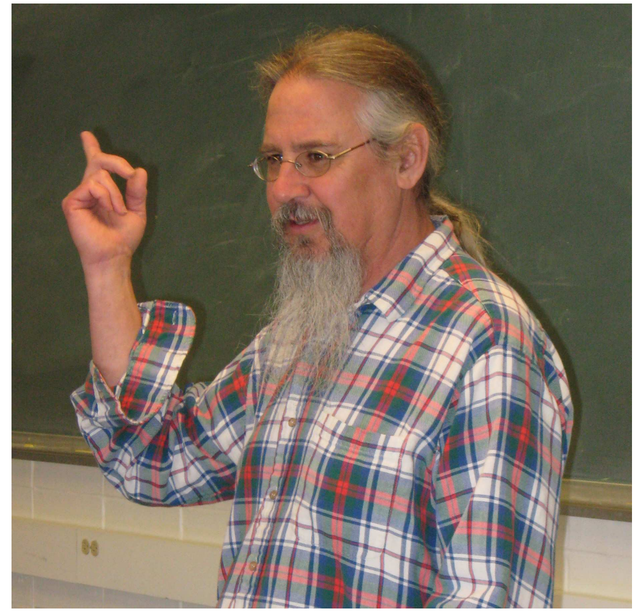
PRECISE seminar

University of Pennsylvania

November 20, 2014

Collaborators

- Jim Rogers (Earlham College)
- Jane Chandlee (Delaware/Nemours)
- Rémi Eyraud (Marseilles)
- Jie Fu (Penn)
- Adam Jardine (Delaware)
- Bill Idsardi (Maryland)
- Regine Lai (HKIEd)
- Bert Tanner (Delaware)
- Ryo Yoshinaka (Kyoto)



Jim Rogers (circa 2010)

Regular Sets, Functions, and Relations

- They can be defined over **different data structures**:
  strings, trees, and graphs.

- They have **applications in several domains**:
  natural language, planning, control, verification, ...

- They have **independently motivated characterizations**:
  MSO-definability, finite-state automata, regular expressions, finite monoid property, ...

- They have many **useful properties**:
  sets are closed under boolean operations, relations are closed under composition, ...

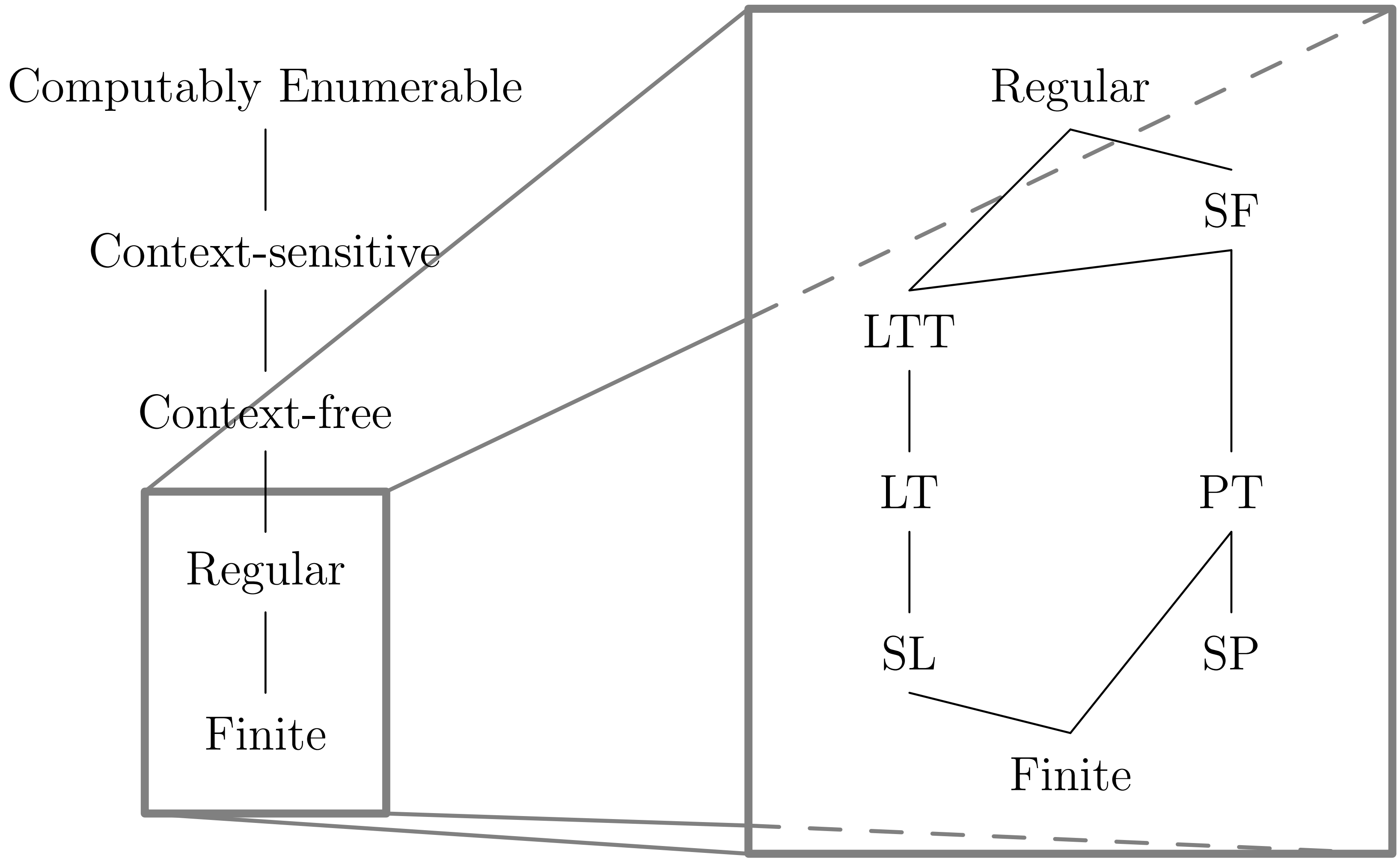
Today's talk: The specific goal

1. For strings, an alphabet Σ is fixed.
2. A string is a sequence of events. Which events are latent and which are observable?
3. Theorems by Medvedev (1964) and Elgot and Mezei (1965) tell us **the choice of alphabet matters**.
4. This choice, along with **determinism**, also matter for *learning* regular sets and functions.

Today's talk: More general goals

1. Introduce you to literature on **subregular** classes of sets and functions.
   - Like the regular class, these classes are natural and have multiple characterizations.
   - Unlike the regular class, some of them are feasibly learnable from positive evidence only.
2. Introduce you to literature on **learning** regular sets and functions (grammatical inference).
3. **Main lesson: For applications, better characterizations of the problem space lead to better solutions.**

Subregular Hierarchies (strings of finite length)



(McNaughton and Papert 1971, Thomas 1997, Rogers and Pullum 2011, Rogers et al. 2013)

Regular stringsets and functions

- Regular stringsets have multiple, equivalent representations.

$$\mathbb{L}(\text{DFA}) \equiv \mathbb{L}(\text{NFA}) \equiv \mathbb{L}(\text{MSO}_L) \equiv \mathbb{L}(\text{RE}) \equiv \mathbb{L}(\text{GRE})$$

- The expressive capacity of these representations separate when we consider probabilty distributions over strings.

$$\mathbb{L}(\text{PDFA}) \subsetneq \mathbb{L}(\text{PNFA})$$

- And they separate when we consider regular functions.

$$\mathbb{L}(\text{DFT}) \subsetneq \mathbb{L}(\text{NFT}) \subsetneq \mathbb{L}(\text{MSO}_f)$$

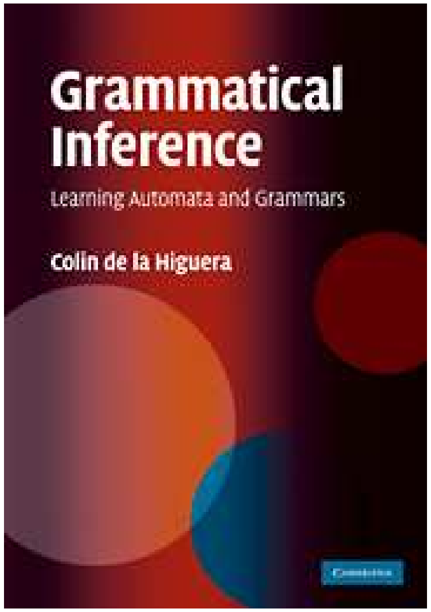
(Kleene 1956, Scott and Rabin 1959, Büchi 1960, Berstel 1979, Vidal et al. 2005, Engelfriet and Hoogeboom 2001)

How can one learn regular stringsets and functions from examples?

Answer

1. Define ‘learning.’
2. Define ‘examples.’

de la Higuera (2010) provides a comprehensive survey of research that addresses these questions and definitions.



Defining 'learning'

Let $\mathbb{T}$ be a class, and $\mathbb{R}$ a class of representations for $\mathbb{T}$.

Definition 1 (Strong characteristic sample) *For a $(\mathbb{T},\mathbb{R})$-learning algorithm $\mathfrak{A}$, a sample CS is a* strong characteristic sample *of a representation $r \in \mathbb{R}$ if for all samples S for $\mathbb{L}(r)$ such that $CS \subseteq S$, $\mathfrak{A}$ returns r.*

Definition 2 (Strong identification in polynomial time and data) *A class $\mathbb{T}$ of functions is strongly identifiable in polynomial time and data if there exists a $(\mathbb{T},\mathbb{R})$-learning algorithm $\mathfrak{A}$ and two polynomials $p()$ and $q()$ such that:*

1. *For any sample S of size m for $t \in \mathbb{R}$, $\mathfrak{A}$ returns a hypothesis $r \in \mathbb{R}$ in $\mathcal{O}(p(m))$ time.*
2. *For each representation $r \in \mathbb{R}$ of size k, there exists a strong characteristic sample of r for $\mathfrak{A}$ of size at most $\mathcal{O}(q(k))$.*

(de la Higuera 1997, 2010, Eyraud et al. to appear)

Defining ‘examples’

1. *Positive* examples are ones labeled as belonging to the target stringset or function.

2. *Negative* examples are ones labeled as **not** belonging to the target stringset or function.

Learning results

1. The class of regular stringsets is strongly identifiable in polynomial time and data with **positive and negative** examples by the algorithm RPNI, which uses DFA.

   (Oncina and García 1992)

2. Any class properly containing FIN is **not** so identifiable with **only positive** examples. This holds even if the polynomial bounds are removed, and 'strong' identification is relaxed.

   $\Rightarrow$ Regular stringsets are not learnable from positive data only.

   (Gold 1967)

3. OTOH, deterministic, but **not non**deterministic, total regular functions (and distributions) **are** strongly identifiable in polynomial time and data from **only positive** examples by the algorithm OSTIA (ALEGRIA) which uses DFT (PDFA).

   (Oncina, García, and Vidal 1993, Carrasco and Oncina 1994, 1999)

Models of strings

Suppose $\Sigma = \{a, b, c\}$. Two models:

$$\begin{array}{ll} \textbf{substring model}: & \langle \mathcal{D}, \rhd, P_a, P_b, P_c \rangle \\ \textbf{subsequence model}: & \langle \mathcal{D}, \preceq, P_a, P_b, P_c \rangle \end{array}$$

- $\mathcal{D}$ is the domain (positions in the string)
- $P_\sigma \subseteq \mathcal{D}$ are labeling predicates (positions labeled σ)
- $\rhd$ is the **successor** relation ($x \rhd y \Leftrightarrow x + 1 = y$).
- $\preceq$ is the **precedence** relation ($x \preceq y \Leftrightarrow x \leq y$).

Example Models

Suppose $\Sigma = \{a, b, c\}$. Two models:

$$\begin{array}{ll} \textbf{substring model}: & \langle \mathcal{D}, \triangleright, P_a, P_b, P_c \rangle \\ \textbf{subsequence model}: & \langle \mathcal{D}, \preceq, P_a, P_b, P_c \rangle \end{array}$$

$$a\ b\ c\ c\ a\ b$$

Under the **substring** model: **bcc** is a sub-structure of a**bcc**ab.

Under the **subsequence** model: **aab** is a sub-structure of **a**bcc**ab**.

Building the Hierarchies

$\triangleright$ $\preceq$

SL SP Conjunction of Negative Literals

Finite

Strictly Local: Conjunctions of negative literals under the substring model

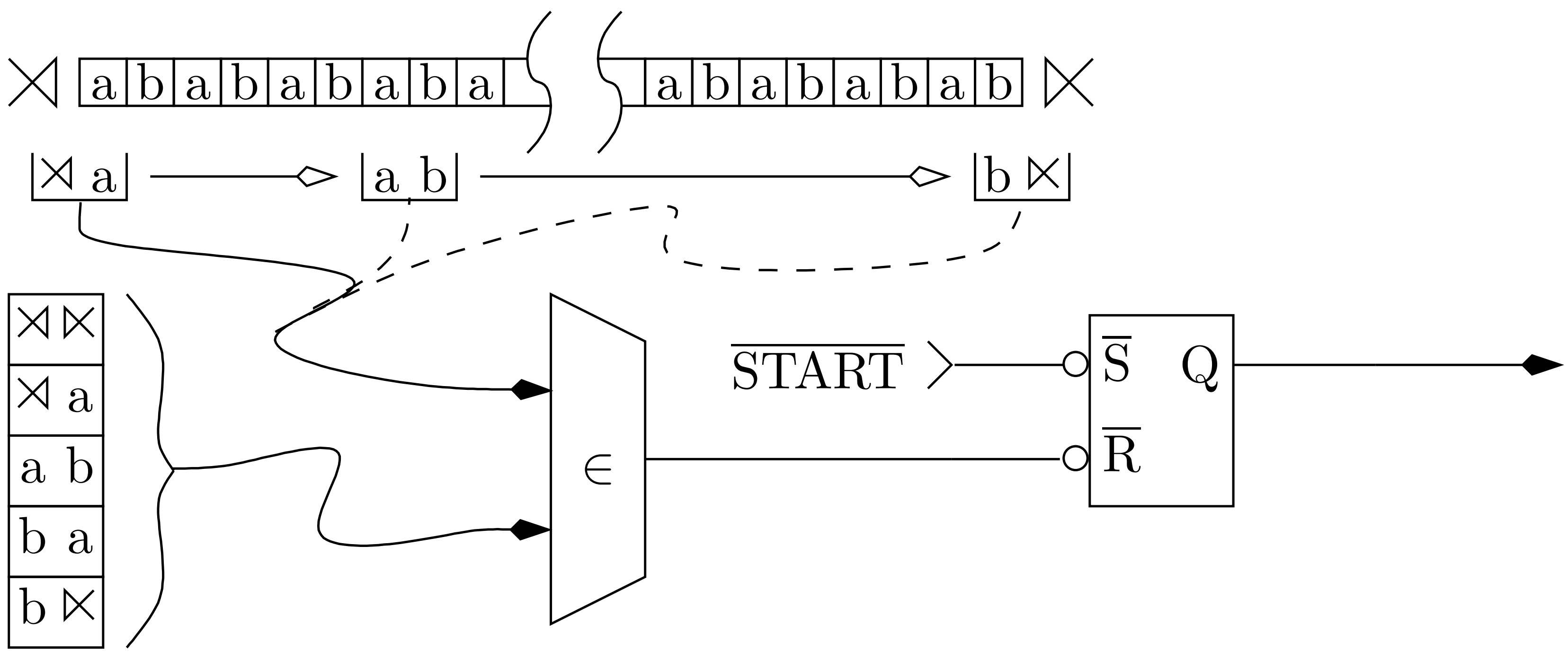
$$L \;=\; \{ab, abab, ababab, \ldots\}$$

$$\varphi \;=\; (\neg \rtimes b) \;\wedge\; (\neg aa) \;\wedge\; (\neg bb) \;\wedge\; (\neg a \ltimes)$$

Strictly Local: Conjunctions of negative literals under the substring model

$$
\begin{array}{rcl}
L & = & \{ab, abab, ababab, \ldots\} = \\[1em]
\mathbb{L}(\varphi) & = & \overline{b\Sigma^*} \quad \cap \quad \overline{\Sigma^* aa\Sigma^*} \quad \cap \quad \overline{\Sigma^* bb\Sigma^*} \quad \cap \quad \overline{\Sigma^* a} \\[1em]
\varphi & = & (\neg \rtimes b) \quad \wedge \quad (\neg aa) \quad \wedge \quad (\neg bb) \quad \wedge \quad (\neg a \ltimes)
\end{array}
$$

A Strictly Local automaton is a scanner



Strictly k-Local stringsets

1. A SL_k stringset is one whose longest forbidden substring is of length k.

2. SL stringsets are those that are SL_k for some k.

- **Theorem:** $(\forall k)[\mathrm{SL}_k \subsetneq \mathrm{SL}_{k+1}]$.
- **Theorem:** $(\forall L \in \mathrm{FIN})(\exists k)[L \in \mathrm{SL}_k]$.
- **Theorem:** $L \in \mathrm{SL} \Leftrightarrow L$ is closed under suffix substitution.
- **Theorem:** For all k, SL_k is strongly identifiable in polynomial time and data from positive examples only.

(McNaughton and Papert 1971, Garcia et al. 1990, Rogers and Pullum 2011, Heinz et al. 2012, Heinz and Rogers 2013, Rogers et al. 2013)

Building the Hierarchies

$\rhd$ $\preceq$

SL SP Conjunction of Negative Literals

Finite

Strictly Piecewise: Conjunctions of negative literals under the subsequence model

$$\varphi \;=\; (\neg aa) \;\wedge\; (\neg bc)$$

Strictly Piecewise: Conjunctions of negative literals under the subsequence model

$$\begin{array}{rcl} \mathbb{L}(\varphi) & = & \overline{\Sigma^* a \Sigma^* a \Sigma^*} \quad \cap \quad \overline{\Sigma^* b \Sigma^* c \Sigma^*} \\ \varphi & = & (\neg aa) \qquad\qquad \wedge \quad (\neg bc) \end{array}$$

Strictly k-Piecewise stringsets

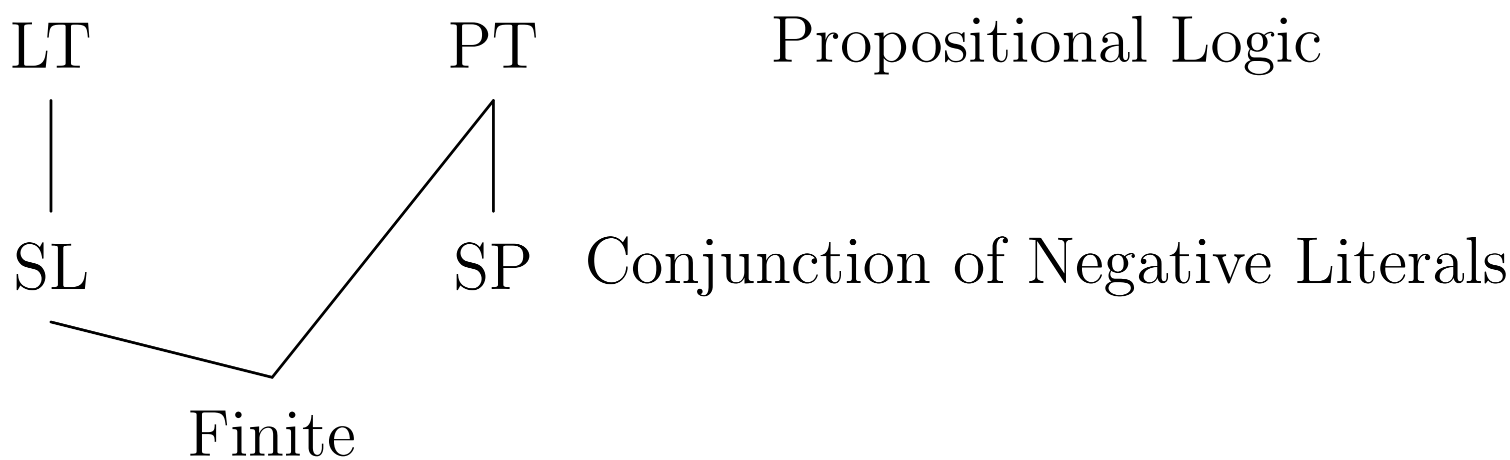
1. A SP_k stringset is one whose longest forbidden subsequence is of length k.
2. SP stringsets are those that are SP_k for some k.

- **Theorem:** $(\forall k)[\mathrm{SP}_k \subsetneq \mathrm{SP}_{k+1}]$.
- **Theorem:** $L \in SP \Leftrightarrow L$ is closed under subsequence.
- **Corollary:** There are finite languages not in SP.
- **Theorem:** For all k, SP_k is strongly identifiable in polynomial time and data from positive examples only.

(Heinz 2007, 2010, Rogers et al. 2010, 2013, Heinz et al. 2012, Heinz and Rogers 2013)

Building the Hierarchies

$\rhd$ $\preceq$



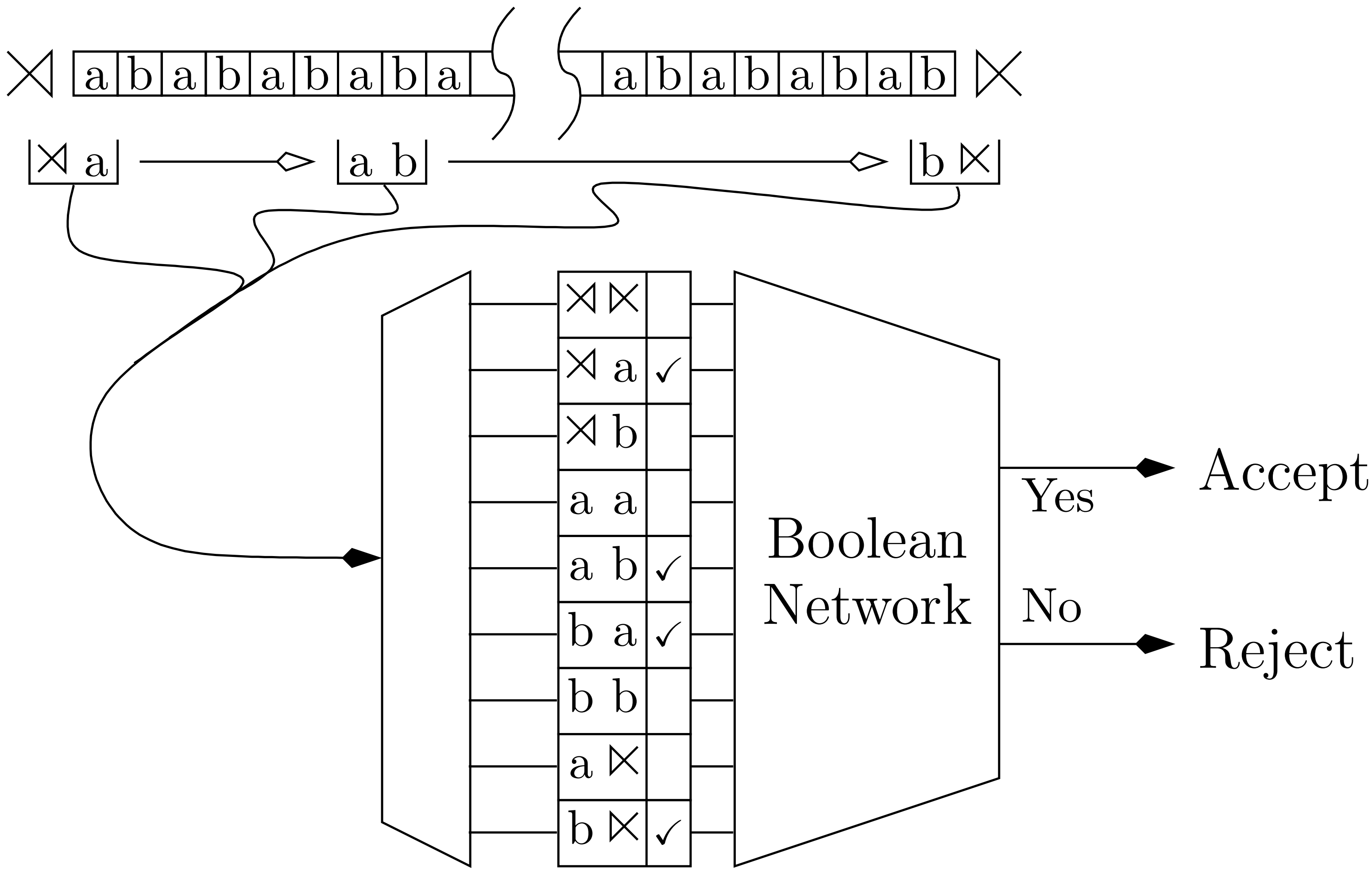
Locally Testable: Propositional logic with the substring model

$$\varphi \;=\; b \;\vee\; (ab \Rightarrow bc)$$

Locally Testable: Propositional logic with the substring model

$$\begin{array}{lcl} \mathbb{L}(\varphi) & = & \Sigma^* b \Sigma^* \;\; \cup \;\; (\Sigma^* ab \Sigma^* ac \Sigma^* \cup \Sigma^* ac \Sigma^* ab \Sigma^*) \\ \varphi & = & b \qquad\qquad \vee \;\; (ab \Rightarrow ac) \end{array}$$

A Locally Testable automaton is a boolean network



Locally k-Testable stringsets

1. A LT_k stringset is one defined with a formula whose longest string is of length k.

2. LT stringsets are those that are LT_k for some k.

- **Theorem:** $(\forall k)[\mathrm{LT}_k \subsetneq \mathrm{LT}_{k+1}]$.

- **Theorem:** $SL \subsetneq \mathrm{LT}$

- **Theorem:** LT is the smallest class which is closed under boolean operations and contains SL.

- **Theorem:** $L \in \mathrm{LT} \Leftrightarrow (\exists k)(\forall u, v)[$ if u, v have the same k-long substrings then either $u, v \in L$ or $u, v \notin L]$.

- **Theorem:** For all k, LT_k **is** strongly identifiable from positive examples only, but **not** in polynomial time and data.

(McNaughton and Papert 1971, García and Ruiz 1996, 2004, Rogers and Pullum 2011, Heinz et al. 2012, Heinz and Rogers 2013, Rogers et al. 2013)

Piecewise Testable: Propositional logic with the subsequence model

$$
\begin{array}{rcll}
\mathbb{L}(\varphi) & = & \Sigma^*b\Sigma^*c\Sigma^* & \cup \quad \overline{\Sigma^*a\Sigma^*b\Sigma^*} \\
\varphi & = & bc & \vee \quad (\neg ab)
\end{array}
$$

Piecewise k-Testable stringsets

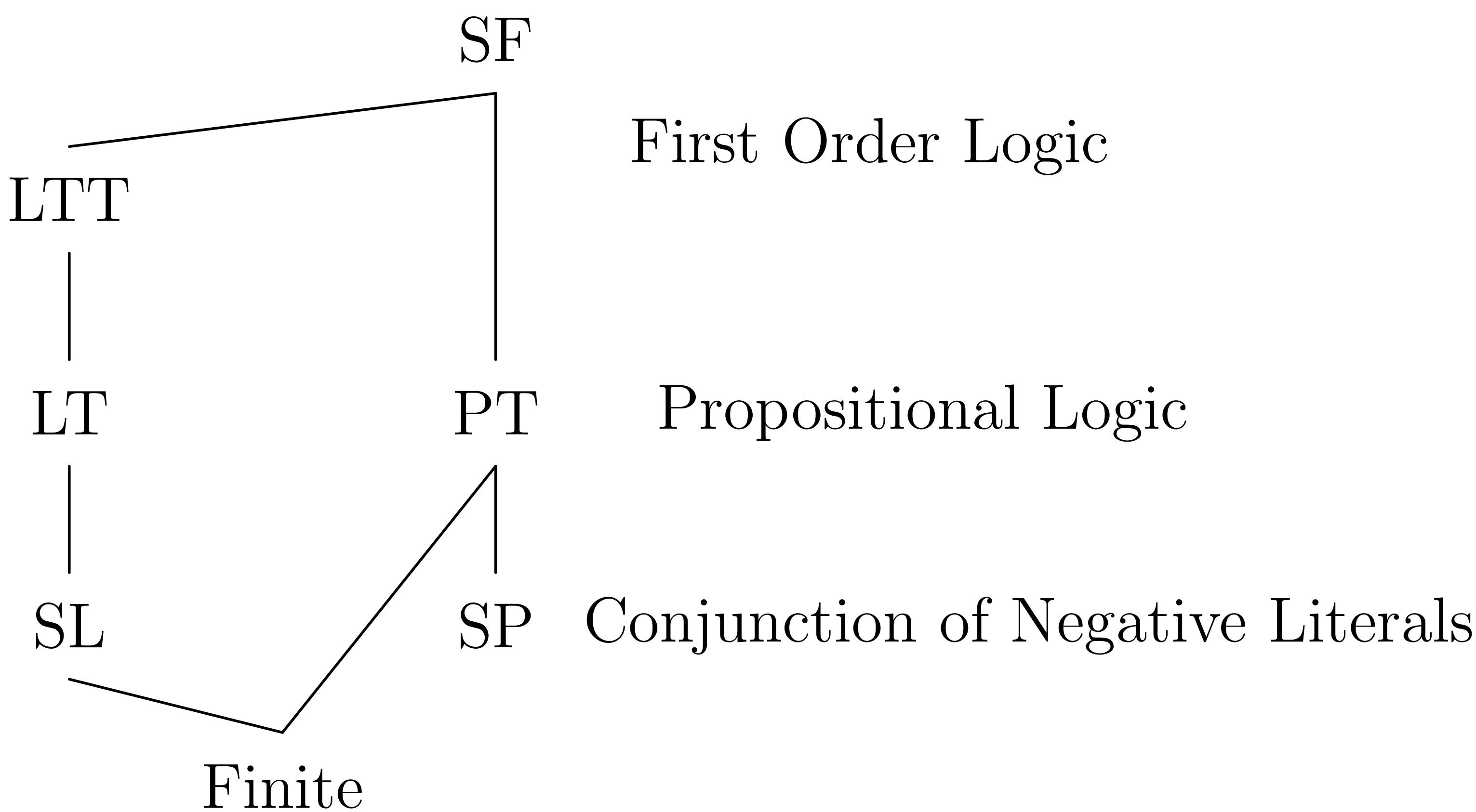
1. A PT_k stringset is one defined with a formula whose longest string is of length k.
2. PT stringsets are those that are PT_k for some k.

- **Theorem:** $(\forall k)[\mathrm{PT}_k \subsetneq \mathrm{PT}_{k+1}]$.
- **Theorem:** $SP \subsetneq \mathrm{PT}$
- **Theorem:** PT is the smallest class which is closed under boolean operations and contains SP.
- **Theorem:** $L \in \mathrm{PT} \Leftrightarrow (\exists k)(\forall u, v)[$ if u, v have the same k-long subsequences then either $u, v \in L$ or $u, v \notin L]$.
- **Theorem:** For all k, PT_k **is** strongly identifiable from positive examples only, but **not** in polynomial time and data.

(McNaughton and Papert 1971, Simon 1975, García and Ruiz 1996, 2004, Rogers and Pullum 2011, Heinz et al. 2012, Heinz and Rogers 2013, Rogers et al. 2013)

Building the Hierarchies

$\triangleright$ $\qquad\qquad\qquad$ $\preceq$



Locally Threshold Testable: First order logic with the substring model

$$\textbf{substring model}: \quad \langle \mathcal{D}, \triangleright, P_a, P_b, P_c \rangle$$

$$\begin{aligned}\varphi \;=\; (\exists w, x, y, z)[\;\; & P_a(w) \wedge P_b(x) \wedge w \triangleright x \\ & P_a(y) \wedge P_b(z) \wedge y \triangleright z \\ & \wedge w \neq x \neq y \neq z]\end{aligned}$$

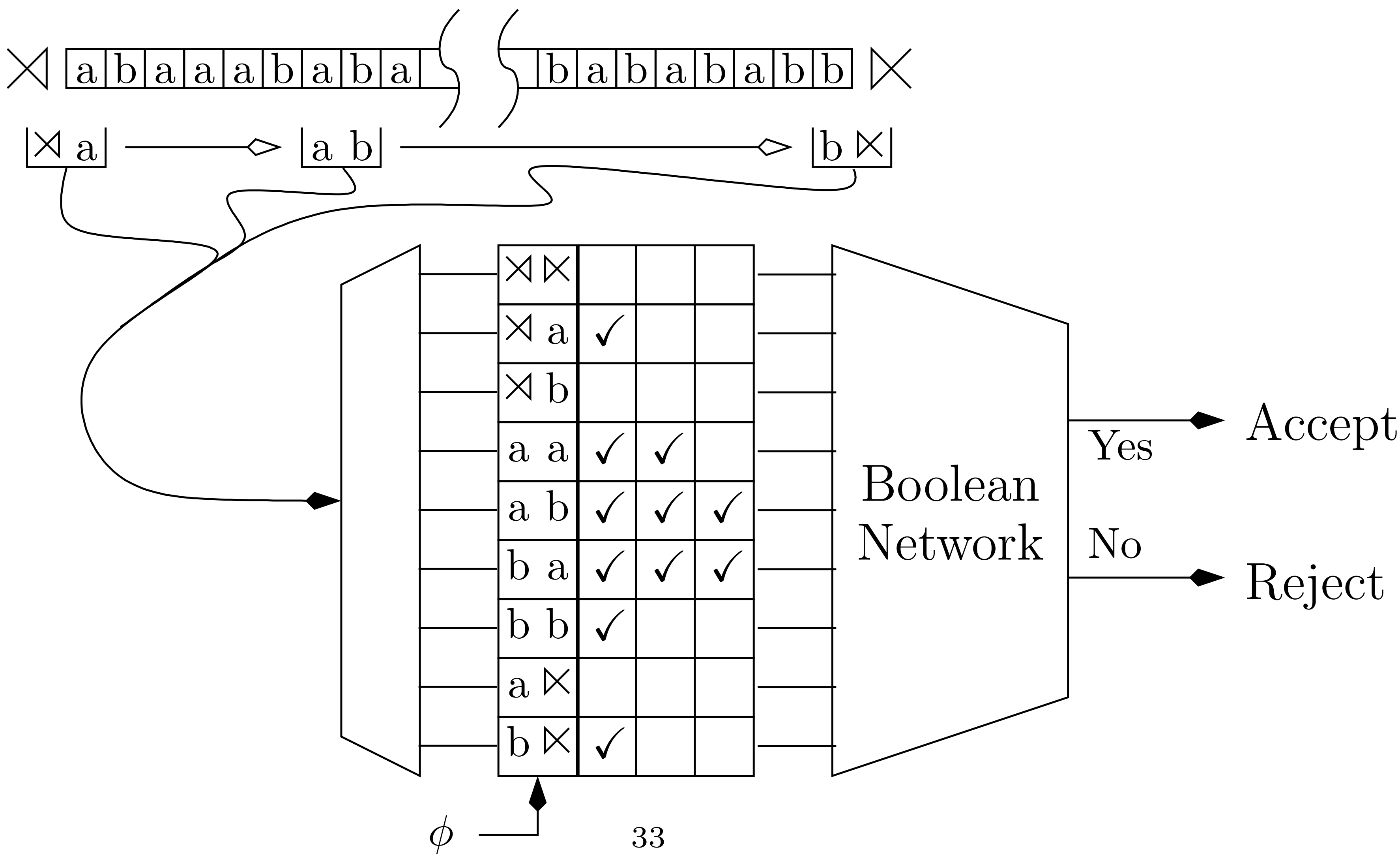
Locally Threshold Testable: First order logic with the substring model

$$\textbf{substring model}: \quad \langle \mathcal{D}, \triangleright, P_a, P_b, P_c \rangle$$

$$\mathbb{L}(\varphi) = \Sigma^* ab \Sigma^* ab \Sigma^*$$

$$\begin{aligned} \varphi \;\; = \;\; (\exists w, x, y, z)[\;\; & P_a(w) \wedge P_b(x) \wedge w \triangleright x \\ & P_a(y) \wedge P_b(z) \wedge y \triangleright z \\ & \wedge w \neq x \neq y \neq z] \end{aligned}$$

A LTT automaton is a LT automaton which counts up to some threshold t



Locally Threshold t, k-Testable stringsets

1. LTT strings are parameterized by the length of substrings k and a maximum counting capacity t.

2. LT stringsets are those that are LT_k for some k.

- **Theorem:** $(\forall k)[\mathrm{LTT}_{t,k} \subsetneq \mathrm{LT}_{t,k+1}]$.

- **Theorem:** $(\forall t)[\mathrm{LTT}_{t,k} \subsetneq \mathrm{LT}_{t+1,k}]$.

- **Theorem:** $\mathrm{LT} \subsetneq \mathrm{LTT}$

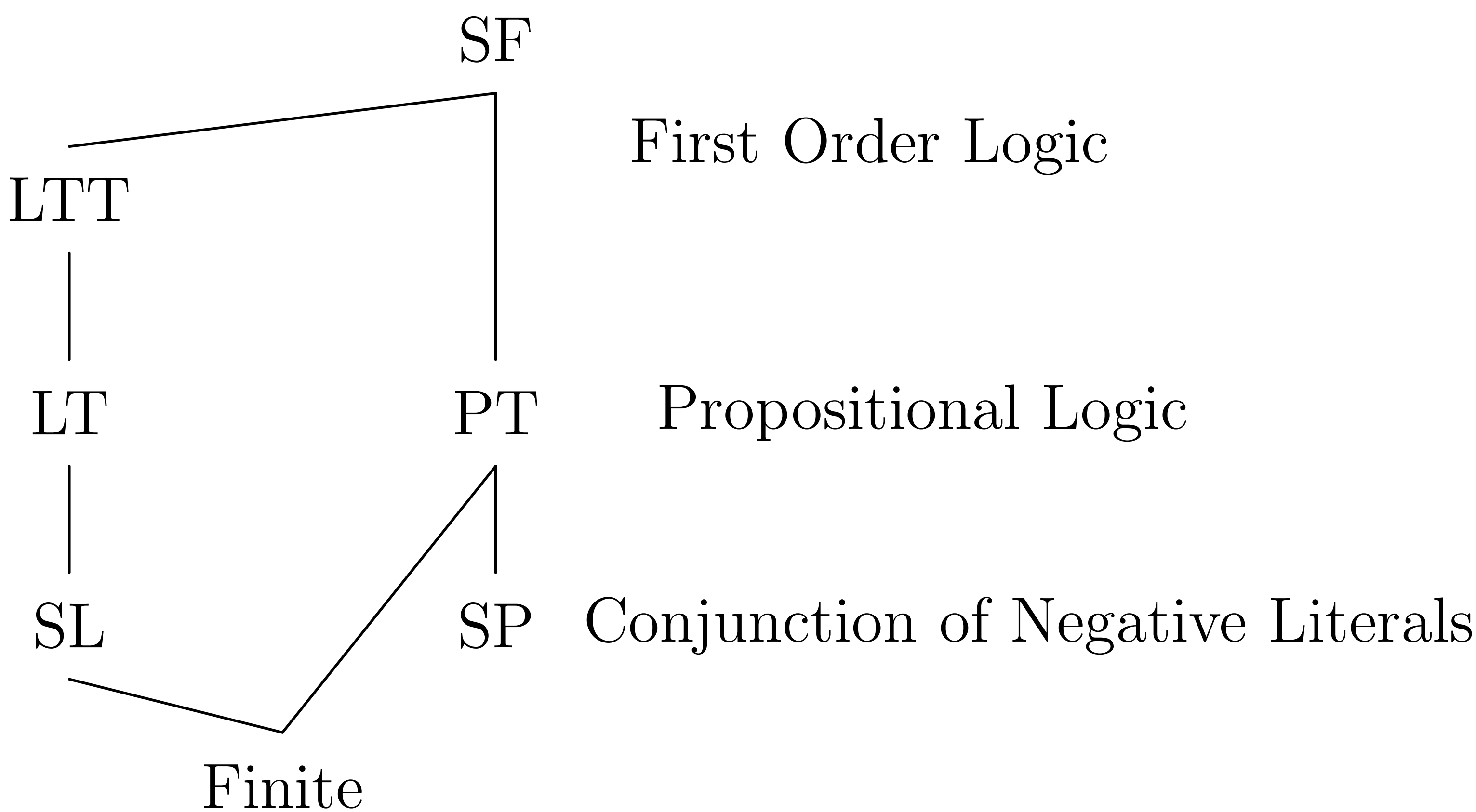
- **Theorem:** $L \in \mathrm{LTT} \Leftrightarrow (\exists k, t)(\forall u, v)[$ if u, v have the same number of k-long substrings (up to t) then either $u, v \in L$ or $u, v \notin L]$.

- **Theorem:** For all t, k, $\mathrm{LTT}_{t,k}$ **is** strongly identifiable from positive examples only, but **not** in polynomial time and data.

(McNaughton and Papert 1971, Thomas 1997, Rogers and Pullum 2011, Heinz et al. 2012, Rogers et al. 2013)

Building the Hierarchies

$\rhd$ $\preceq$



Star-Free: First order logic with the subsequence model

$$\textbf{subsequence model}: \quad \langle \mathcal{D}, \preceq, P_a, P_b, P_c \rangle$$

$$\mathbb{L}(\varphi) = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^* b \Sigma^* \cup \Sigma^* a \Sigma^* a \Sigma^* b \Sigma^* b \Sigma^*$$

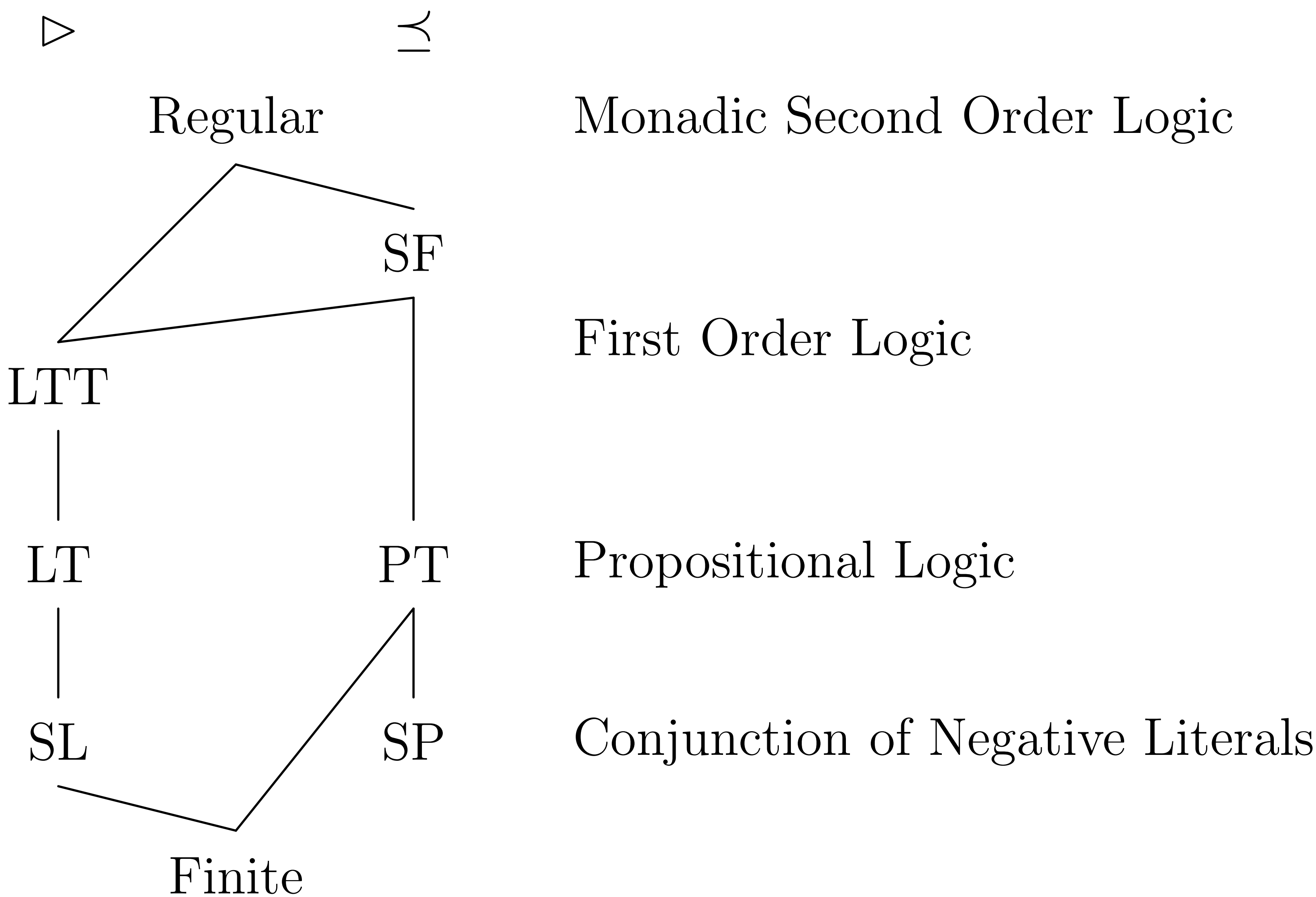
$$\begin{aligned} \varphi \;\; = \;\; (\exists w, x, y, z)[\;\; & P_a(w) \wedge P_b(x) \wedge w \preceq x \\ & P_a(y) \wedge P_b(z) \wedge y \preceq z \\ & \wedge w \neq x \neq y \neq z] \end{aligned}$$

Star-free Stringsets

- **Theorem:** PT $\subsetneq$ SF.
- **Theorem:** LTT $\subsetneq$ SF ($\triangleleft$ is first-order definable from $\preceq$ but **not** vice versa).
- **Theorem:** $L \in \mathrm{SF} \Leftrightarrow (\exists k)(\forall u, v, w)[uv^k w \in L \Rightarrow uv^{k+1} w \in L]$ (so SF is also called NonCounting).
- **Theorem:** $L \in \mathrm{SF} \Leftrightarrow (\exists r \in GRE)[\ \mathbb{L}(r) = L \wedge r$ is a star-free expression].
- **Theorem:** SF is the smallest class of languages obtained by closing LT under concatenation (so SF is also called Locally Testable with Order).
- **Theorem:** $\big(\forall \varphi \in \mathrm{FO}(\preceq)\big)[\exists \varphi' \in \mathrm{TL}(\texttt{until,since})$ such that $\mathbb{L}(\varphi) = \mathbb{L}(\varphi')]$ and vice versa (these are ω-regular languages).

(Kamp 1968, McNaughton and Papert 1971, Rogers et al. 2013)

Building the Hierarchies



Other subregular learnable classes

- **Theorem:** Every DFA defines a class of languages in terms of its sub-DFA that is strongly identifiable in the limit from positive examples.

  (For each k, t, $\mathrm{SL}_k, LT_k, LTT_{t,k}, PT_k$ are such classes).

- **Theorem:** Classes formed by the intersection of languages drawn from learnable classes are also strongly identifiable in the limit from positive examples.

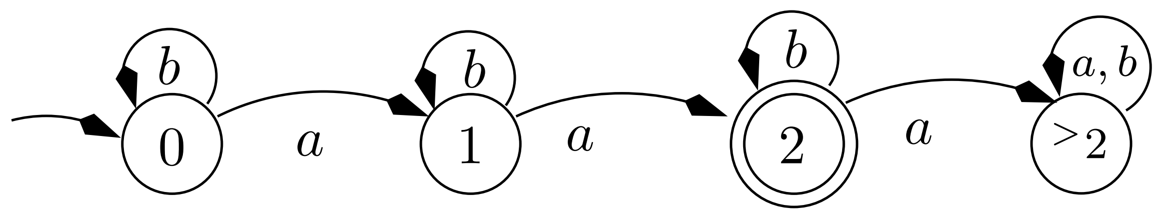
  Example. $\mathcal{L} = \{X \cap Y \mid X \in SL_k \wedge Y \in SP_\ell\}$.

- **Theorem:** Every **list** of DFA defines a class of languages in terms of their sub-DFA that is strongly identifiable in the limit from positive examples. Also, this list representation may be **exponentially** smaller than the single DFA rep. (For each k, SP_k is such a class.)

(Heinz et al. 2012, Heinz and Rogers 2013)

Medvedev's Theorem (1956/1964)

Every regular stringset is a projection (the image under an alphabetic homomorphism) of a Strictly 2-Local stringset.



$$\langle 0,1,a\rangle, \langle 0,0,b\rangle, \langle 1,2,a\rangle, \langle 1,1,b\rangle, \langle 2,^{>}2,a\rangle, \langle 2,2,b\rangle, \langle ^{>}2,^{>}2,a\rangle, \langle ^{>}2,^{>}2,b\rangle$$

- Possible runs through a DFA is a sequence of transitions.

$$abbab \approx \langle 0,1,a\rangle\langle 1,1,b\rangle\langle 1,1,b\rangle\langle 1,2,a\rangle\langle 2,2,b\rangle$$

- The transitions themselves are symbols, forming an alphabet.
- The set of runs leading to final states in a DFA with this alphabet is a SL_2 stringset.

$$\langle p,q,\sigma\rangle \mapsto_h \sigma$$

Moral (Medvedev's Theorem)

- If there is no latent information, and a finite alphabet, everything is SL_2. So if all the possible world states are known, learning becomes trivial in one sense. On the other hand, the size of the alphabet may be astronomical…

- If there is latent information, but the underlying structure is known, then learning is also straightforward. The results on the previous slide for instance can be thought of in a Medvedevian way.

What about regular functions? (Recent work)

$$f : \Sigma^* \rightarrow \Gamma^*.$$

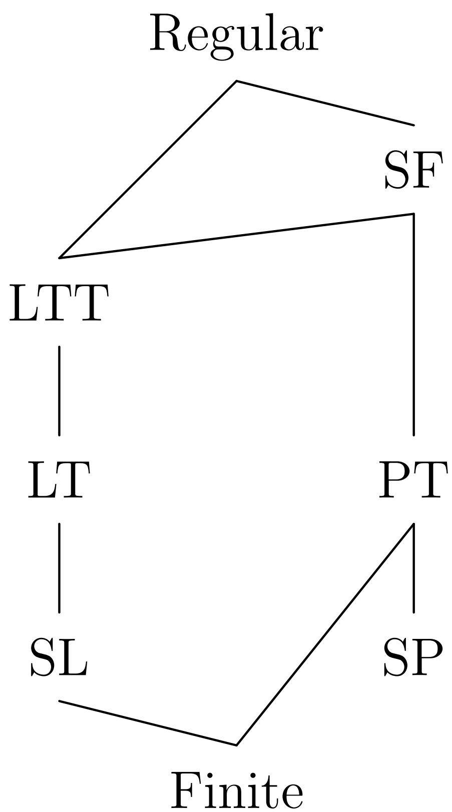
- **Theorem:** Nondeterministic regular functions $\mathbb{L}(\text{NFT})$ are not identifiable in the limit.
- **Theorem:** Total determinstic regular functions are strongly identifiable in polynomial time (and data?) from positive examples. (Two types of determinism: **Left** and **Right**)

$$\mathbb{L}(\text{LDFT}) \text{ and } \mathbb{L}(\text{RDFT})$$

- **Theorem:** There are subclasses of deterministic regular functions, which include partial functions, are strongly identifiable in polynomial time and data from positive examples.

(Oncina et al. 1993, Chandlee 2014, Jardine et. al 2014, Chandlee et al. 2014)

Subregular functions?



- No comparable body of theory for **subregular functions** exists.
- But one of the aforementioned learnable subclasses generalizes the notion of Strict Locality from stringsets to functions (Chandlee 2014).
- Other such subclasses are on their way...

Elgot and Mezei's Theorem (1965)

Let $T : A^* \to C^*$ be a function. Then $T \in \mathbb{L}(\mathrm{NFT})$ iff there exists $L : A^* \to B^* \in \mathbb{L}(\mathrm{LDFT})$, and $R : B^* \to C^* \in \mathbb{L}(\mathrm{RDFT})$ with $A \subseteq B$ such that $T = R \circ L$.

- Notice how the alphabet may grow in the intermediate step!
- **Moral:** Nondeterminism and latent information are deeply connected...

That's it!

1. Theorems by Medvedev (1964) and Elgot and Mezei (1965) tell us **the choice of alphabet matters**.

2. This choice, along with **determinism**, also matter for *learning* regular sets and functions.

3. **Main lesson: For applications, better characterizations of the problem space lead to better solutions.**

4. Many **subregular** classes of sets and functions have a variety of characterizations (=tools) and well as an array of available learning algorithms.

Thanks!

References

Berstel, Jean. 1979. *Transductions and Context-Free languages*. Teubner-Verlag.

Büchi, J. Richard. 1960. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly* 6:66–92.

Carrasco, R. C., and J. Oncina. 1999. Learning deterministic regular grammars from stochastic samples in polynomial time. *RAIRO (Theoretical Informatics and Applications)* 33:1–20.

Carrasco, Rafael C., and José Oncina. 1994. Learning stochastic regular grammars by means of a state merging method. In *Grammatical Inference and Applications, Second International Colloquium, ICGI-94, Alicante, Spain, September 21-23, 1994, Proceedings*, 139–152.

Chandlee, Jane. 2014. Strictly local phonological processes. Doctoral dissertation, The University of Delaware.

Chandlee, Jane, Rémi Eyraud, and Jeffrey Heinz. 2014. Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics* 2:491–503.

Engelfriet, Joost, and Hendrik Jan Hoogeboom. 2001. Mso definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Logic* 2:216–254.

Eyraud, Rémi, Jeffrey Heinz, and Ryo Yoshinaka. to appear. Efficiency in the identification in the limit learning paradigm. In *Advanced Topics in Grammatical Inference*, edited by Jeffrey Heinz and Jose Sempere. Springer.

García, Pedro, and José Ruiz. 1996. Learning k-piecewise testable languages from positive data. In *Grammatical Interference: Learning Syntax from Sentences*, edited by Laurent Miclet and Colin de la Higuera, vol. 1147 of *Lecture Notes in Computer Science*, 203–210. Springer.

García, Pedro, and José Ruiz. 2004. Learning k-testable and k-piecewise testable languages from positive data. *Grammars* 7:125–140.

Garcia, Pedro, Enrique Vidal, and José Oncina. 1990. Learning locally testable languages in the strict sense. In *Proceedings of the Workshop on Algorithmic Learning Theory*, 325–338.

Gold, E.M. 1967. Language identification in the limit. *Information and Control* 10:447–474.

Heinz, Jeffrey. 2007. The inductive learning of phonotactic patterns. Doctoral dissertation, University of California, Los Angeles.

Heinz, Jeffrey, Anna Kasprzik, and Timo Kötzing. 2012. Learning with lattice-structured hypothesis spaces. *Theoretical Computer Science* 457:111–127.

Heinz, Jeffrey, and James Rogers. 2013. Learning subregular classes of languages with factored deterministic automata. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, edited by Andras Kornai and Marco Kuhlmann, 64–71. Sofia, Bulgaria: Association for Computational Linguistics.

de la Higuera, Colin. 1997. Characteristic sets for polynomial grammatical inference. *Machine Learning* 27:125–138.

de la Higuera, Colin. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

Jardine, Adam, Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2014. Very efficient learning of structured classes of subsequential functions from positive data. In *Proceedings of the Twelfth International Conference on Grammatical Inference (ICGI 2014)*, edited by Alexander Clark, Makoto Kanazawa, and Ryo Yoshinaka, vol. 34, 94–108. JMLR: Workshop and Conference Proceedings.

Kamp, Hans. 1968. Tense logic and the theory of linear order. Doctoral dissertation, UCLA.

Kleene, S.C. 1956. Representation of events in nerve nets. In *Automata Studies*, edited by C.E. Shannon and J. McCarthy, 3–40. Princeton University. Press.

McNaughton, Robert, and Seymour Papert. 1971. *Counter-Free Automata*. MIT Press.

Oncina, Jose, and Pedro Garcia. 1992. Identifying regular languages in polynomial time. In *Advances In Structural And Syntactic Pattern Recognition, Volume 5 Of Series In Machine Perception And Artificial Intelligence*, 99–108. World Scientific.

Oncina, José, Pedro García, and Enrique Vidal. 1993. Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15:448–458.

Rogers, James, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. 2010. On languages piecewise testable in the strict sense. In *The Mathematics of Language*, edited by Christian Ebert, Gerhard Jäger, and Jens Michaelis, vol. 6149 of *Lecture Notes in Artifical Intelligence*, 255–265. Springer.

Rogers, James, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2013. Cognitive and sub-regular complexity. In *Formal Grammar*, edited by Glyn Morrill and Mark-Jan Nederhof, vol. 8036 of *Lecture Notes in Computer Science*, 90–108. Springer.

Rogers, James, and Geoffrey Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information* 20:329–342.

Scott, Dana, and Michael Rabin. 1959. Finite automata and their decision problems. *IBM Journal of Research and Development* 5:114–125.

Simon, Imre. 1975. Piecewise testable events. In *Automata Theory and Formal Languages*, 214–222.

Thomas, Wolfgang. 1997. Languages, automata, and logic. vol. 3, chap. 7. Springer.

Vidal, Enrique, Franck Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. 2005a. Probabilistic finite-state machines-part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27:1013–1025.

Vidal, Enrique, Frank Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. 2005b. Probabilistic finite-state machines-part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27:1026–1039.