Subregular languages for robotics

Jeffrey Heinz
heinz@udel.edu

University of Delaware

Boston University
March 18, 2011

Joint work with

Jie Fu (UD)

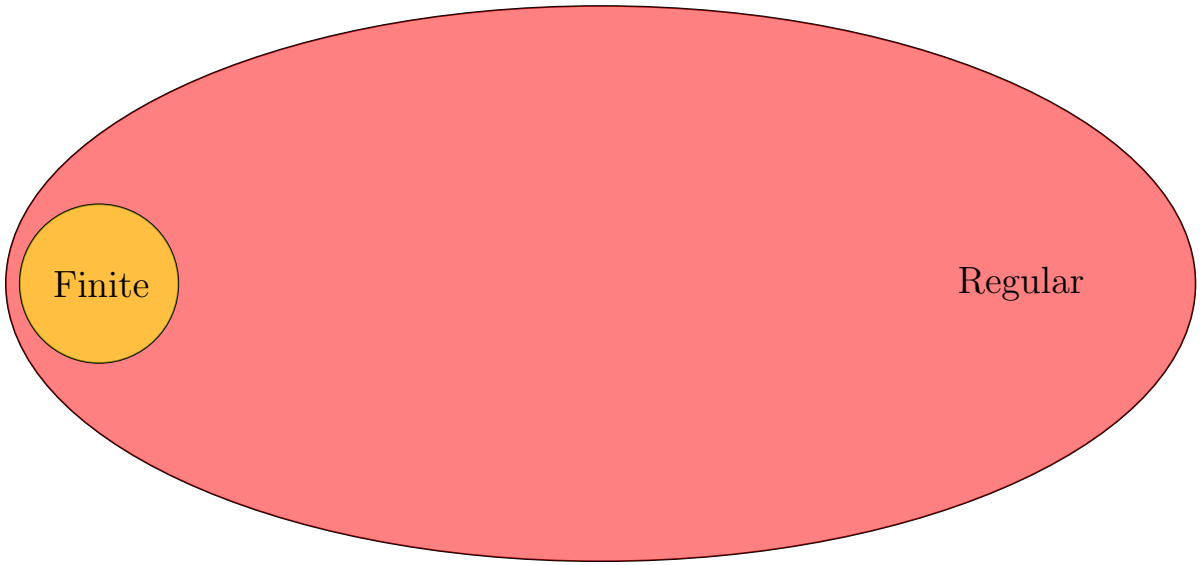
Chetan Rawal (UD)

Herbert Tanner (UD)

Why subregular?

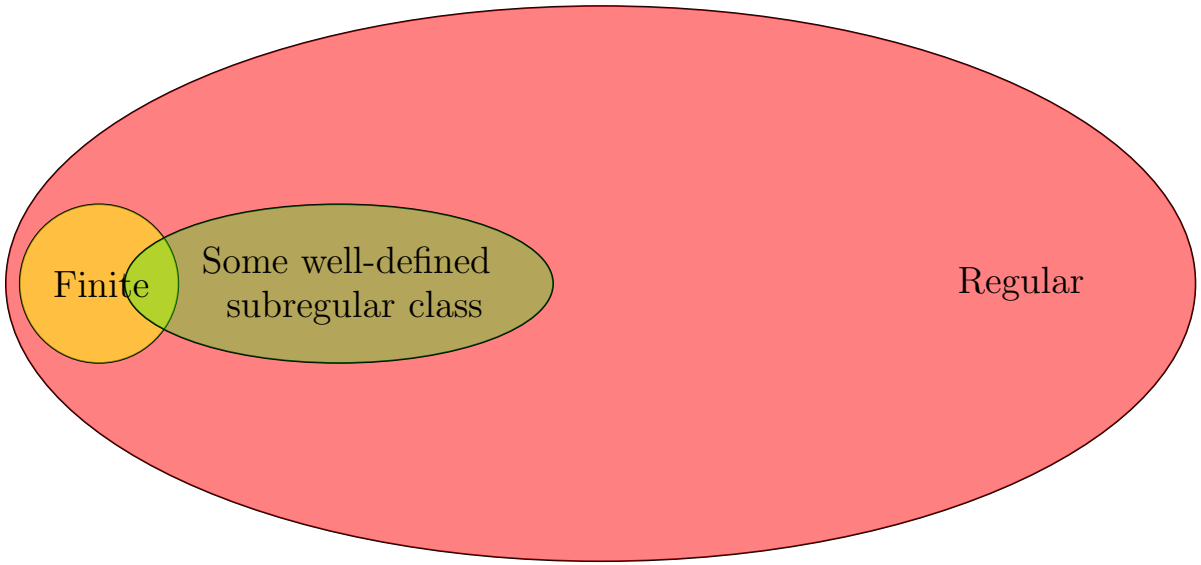
1. Subregular language classes *better characterize* the patterns we are interested in (both in phonology and in robotics).
2. Hard problems are *easier to solve* with better characterizations because the instance space of the problem is smaller.

Example #1: Product



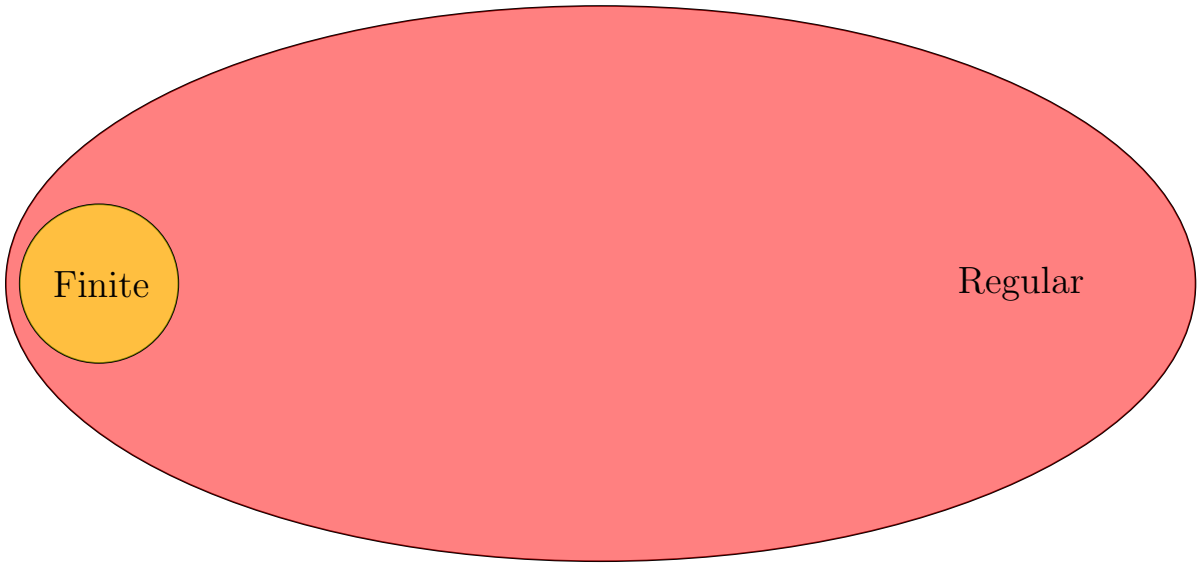
Instead of intersecting arbitrarily many arbitrary regular sets (which is NP-hard), intersect arbitrarily many sets *which belong to some subregular region.*

Example #1: Product



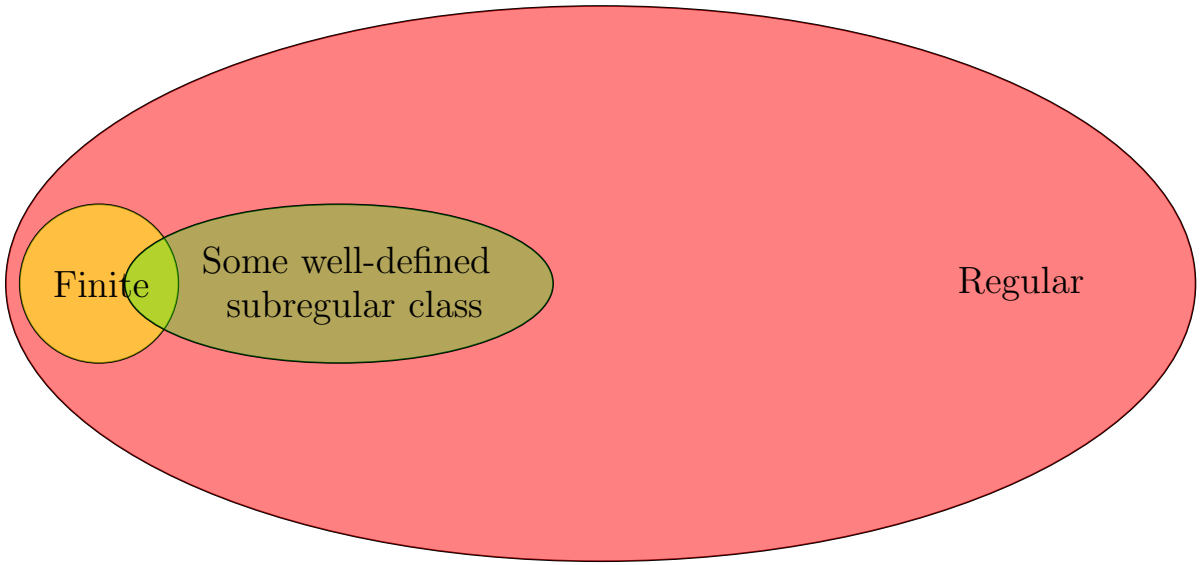
Instead of intersecting arbitrarily many arbitrary regular sets (which is NP-hard), intersect arbitrarily many sets *which belong to some subregular region.*

Example #2: Learning



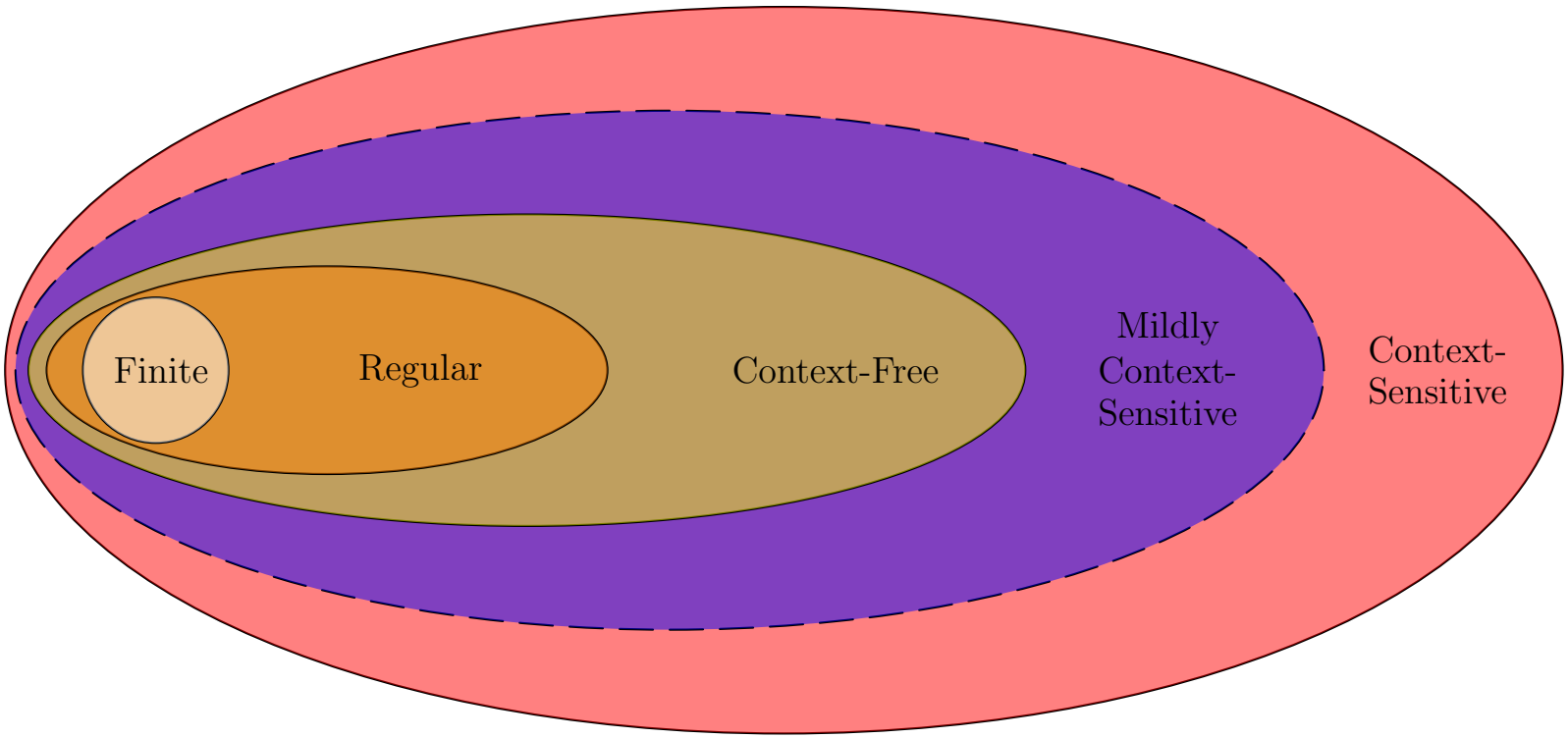
Instead of trying to learn any regular set (which is either impossible or NP-hard depending on the definition of "learn"), try to learn only those *which belong to some subregular region*.

Example #2: Learning



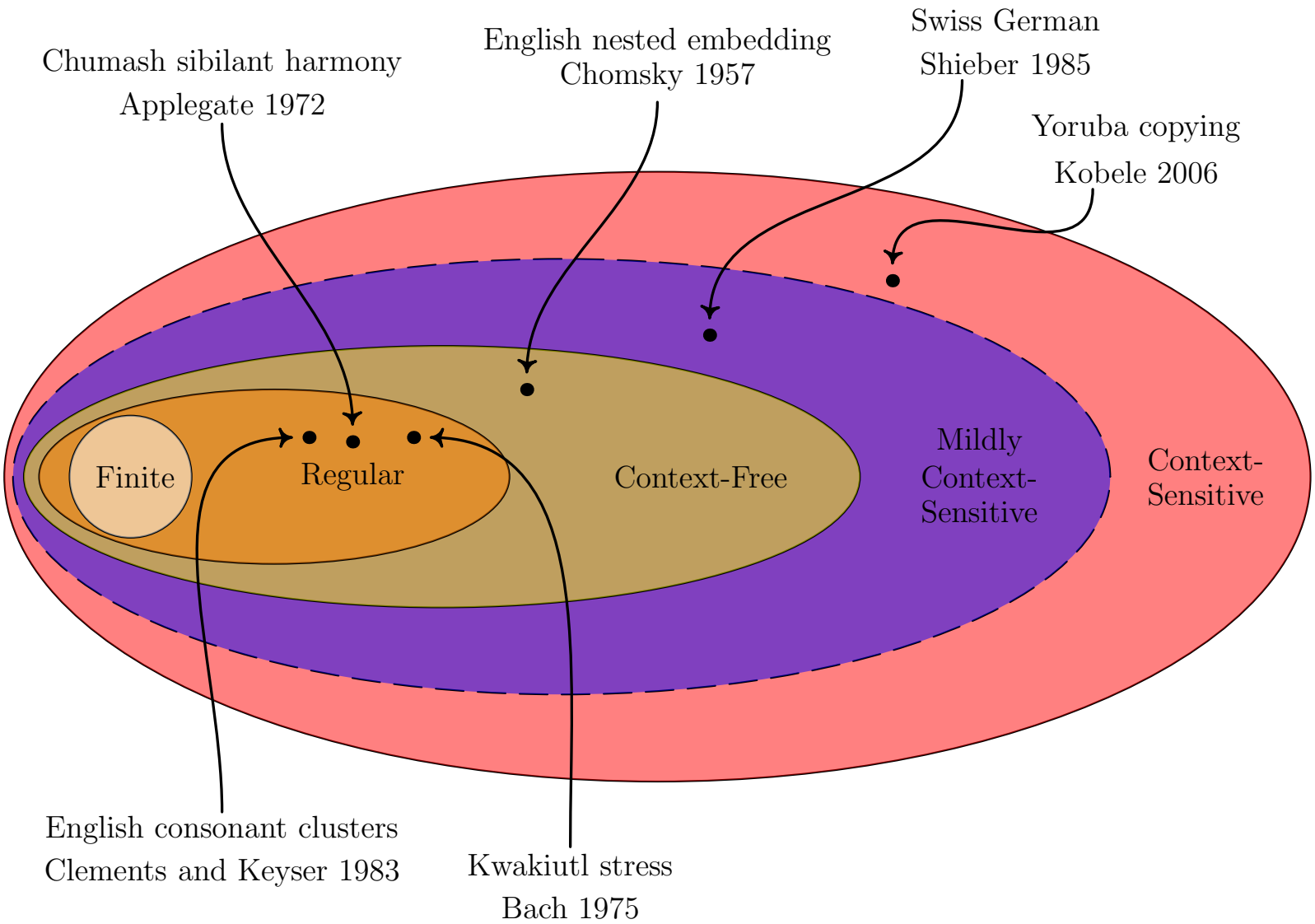
Instead of trying to learn any regular set (which is either impossible or NP-hard depending on the definition of "learn"), try to learn only those *which belong to some subregular region.*

Formal language theory and linguistics



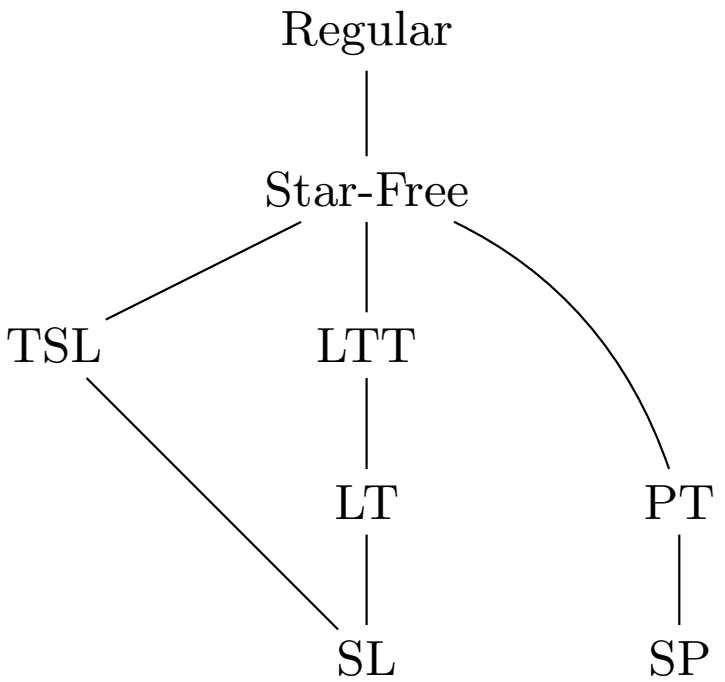
The Chomsky Hierarchy

Formal language theory and linguistics



The Chomsky Hierarchy

Subregular classes



Proper inclusion relationships among subregular language classes (indicated from top to bottom). This paper establishes the TSL class and its place in the figure.

TSL	Tier-based Strictly Local	PT	Piecewise Testable
LTT	Locally Threshold Testable	SL	Strictly Local
LT	Locally Testable	SP	Strictly Piecewise

(McNaughton and Papert 1971, Simon 1975, Rogers and Pullum 2007, in press, Rogers et al. 2010)

A Robotic Language

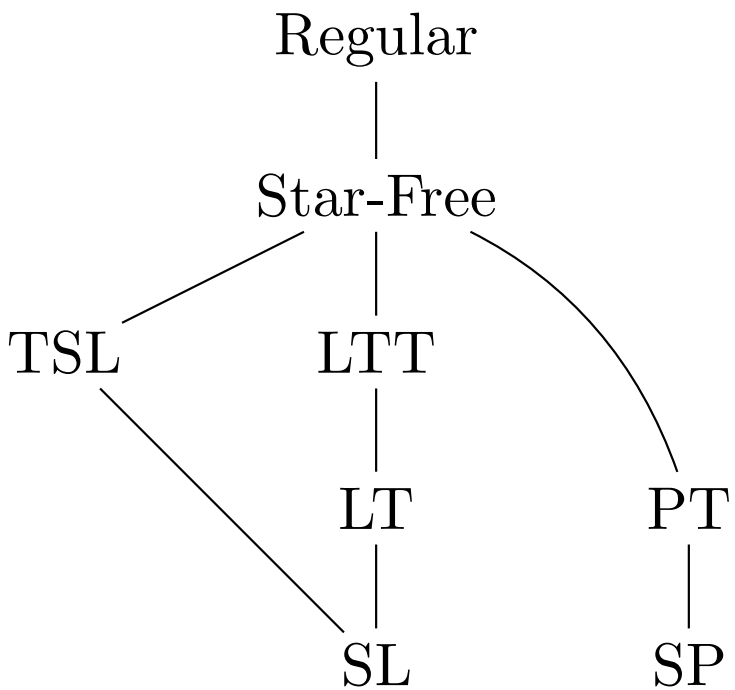
(Picture from the Hammacher Schlemmer catalog)

A	move
B	grab
C	release

- AABAAAACAAABAAA is a well-formed action sequence.
- AABAAAABAAACAAA is not.

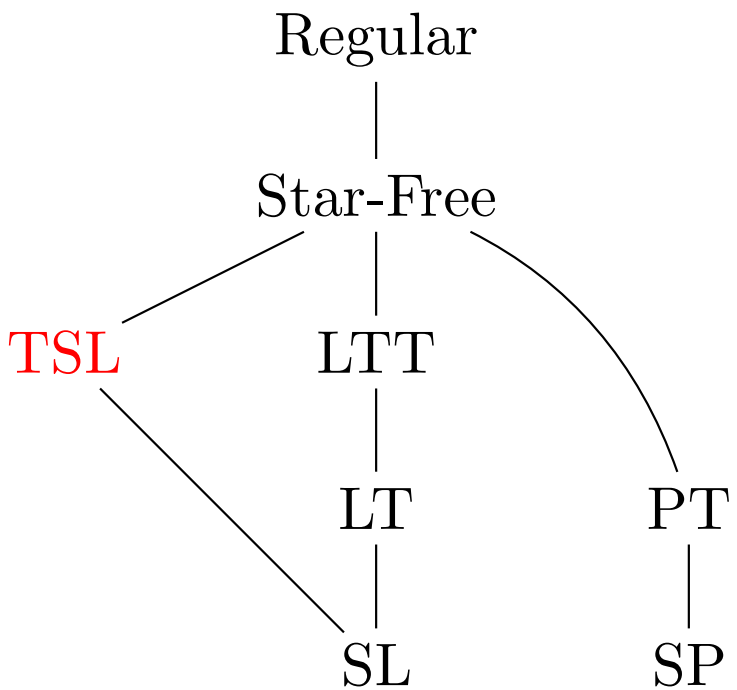
The Robotic Language

1. What language does this describe?
2. Where is it in the hierarchies?



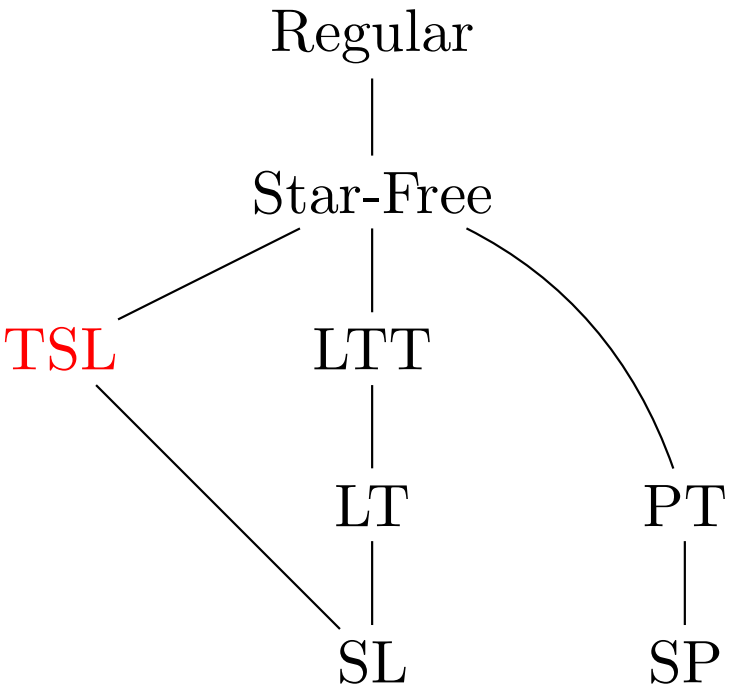
The Robotic Language

1. What language does this describe?
2. Where is it in the hierarchies?



Tier-based Strictly Local Languages

1. Intersection is easier.
2. Learning is easier.



Defining Regular

Generalized Regular Expressions (GREs)

1. λ, $\emptyset$ and each letter of Σ are GREs.
2. $R, S \in$ GREs $\Rightarrow$ RS, $R + S$, $R \times S$, $\overline{R}$, and R^* are GREs.

Language of GREs

1. $L(\emptyset) = \emptyset$.
2. For all $\sigma \in \Sigma \cup \{\lambda\}$, $L(\sigma) = \{\sigma\}$.
3. If $R, S \in$ GREs $\Rightarrow$
   - 3.1 $L(RS) = L(R)L(S)$
   - 3.2 $L(R + S) = L(R) \cup L(S)$
   - 3.3 $L(R \times S) = L(R) \cap L(S)$
   - 3.4 $L(\overline{R}) = \Sigma^* - L(R)$
   - 3.5 $L(R^*) = L(R)^*$.

A language is *regular* iff there exists a GRE describing it.

Robot Claw Language is regular

$$A^*(BA^*CA^*)^*(\lambda + B)A^*$$

1. Of “Grab” and “Release”, must “Grab” first.
2. A “Release” must intervene between two “Grab”
3. A “Grab” must intervene between two “Release”
4. Of “Grab” and “Release”, last one can be either.

Robot Claw Language is regular

$$A^*(BA^*CA^*)^*(\lambda + B)A^*$$

1. Of "Grab" and "Release", must "Grab" first.
2. A "Release" must intervene between two "Grab"
3. A "Grab" must intervene between two "Release"
4. Of "Grab" and "Release", last one can be either.

Robot Claw Language is regular

$$A^*(BA^*CA^*)^*(\lambda + B)A^*$$

1. Of "Grab" and "Release", must "Grab" first.
2. A "Release" must intervene between two "Grab"
3. A "Grab" must intervene between two "Release"
4. Of "Grab" and "Release", last one can be either.

Robot Claw Language is regular

$$A^*(BA^*CA^*)^*(\lambda + B)A^*$$

1. Of "Grab" and "Release", must "Grab" first.
2. A "Release" must intervene between two "Grab"
3. A "Grab" must intervene between two "Release"
4. Of "Grab" and "Release", last one can be either.

Defining Star-Free

Definition

A language is *star-free* iff there exists a GRE describing it which contains no Kleene star '*'.

Theorem

(McNaughton and Papert 1971) A language L is star-free iff there exists some n such that for all $u, v, w \in \Sigma^*$:

$$uv^n w \in L \Rightarrow uv^{n+1} w \in L$$

Robot Claw Language is star-free

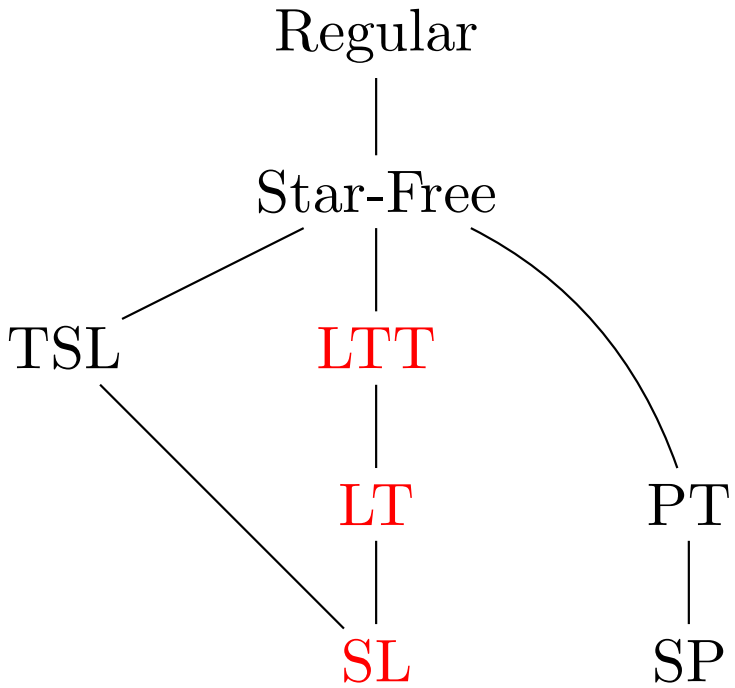
$$A^*(BA^*CA^*)^*(\lambda + B)A^*$$

Robot Claw Language is star-free

$$A^*(BA^*CA^*)^*(\lambda + B)A^*$$

Take my word for it, for now!

Defining Factors for the Local Branch



Definition

String u is a *factor* of string w iff $\exists x, y \in \Sigma^*$ such that $w = xuy$.
If also $|u| = k$ then u is a *k-factor* of w.

Example: ab is a 2-factor of $aaabbb$.

Factor Functions

The function F_k maps words to the set of k-factors within them.

$$F_k(w) = \{u : u \text{ is a } k\text{-factor of } w\} \tag{1}$$

Example: $F_2(abc) = \{ab, bc\}$.

The function $F_{k,t}$ *counts* k-factors up to some threshold t.

$$\begin{aligned} F_{k,t}(w) = \quad & \{(u, n) : u \text{ is a } k\text{-factor of } w \text{ and} \\ & n = |w|_u \text{ iff } |w|_u < t \text{ else } n = t\} \end{aligned} \tag{2}$$

Example: $F_{2,3}(aaaaab) = \{(aa, 3), (ab, 1)\}$.

Defining Strictly Local

Definition

A language L is *Strictly Local* (SL) iff there exists a $k \in \mathbb{N}$ and a finite set $S \subseteq F_k(\rtimes \Sigma^* \ltimes)$ such that

$$L = \{w \in \Sigma^* : F_k(\rtimes w \ltimes) \subseteq S\}$$

Example: a^*bb^* is SL. Let $k = 2$ and $S = \{\rtimes a, aa, ab, bb, b \ltimes\}$.

Forbidden and Permissible k-factors

Example: a^*bb^* is SL. Let $k = 2$ and $S = \{\rtimes a, aa, ab, bb, b\ltimes\}$.

- $aaabbbb \in L$ because its 2-factors are *permissible*.
- $abaab \notin L$ because it contains a *forbidden* factors (ba)

Defining Locally Testable

Definition

A language L is *Locally Testable* (LT) iff there exists some $k \in \mathbb{N}$ such that for all $u, v \in \Sigma^*$:

$$F_k(\rtimes u \ltimes) = F_k(\rtimes v \ltimes) \Rightarrow u, v \in L \text{ or } u, v \notin L$$

Defining Locally Threshold Testable

Definition

A language L is *Locally Threshold Testable* (LTT) for iff there exists some $k, t \in \mathbb{N}$ such that for all $u, v \in \Sigma^*$:

$$F_{k,t}(\rtimes u \ltimes) = F_{k,t}(\rtimes v \ltimes) \Rightarrow u, v \in L \text{ or } u, v \notin L$$

Robot Claw Language is not LTT

Theorem

The Robot Claw language is not LTT.

Proof.

Pick any k and threhold t. Let

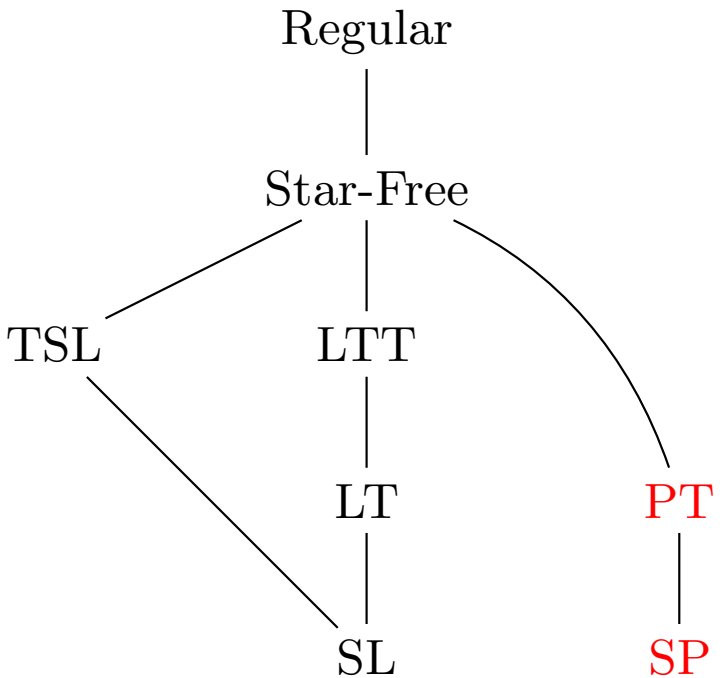
$$u = A^k B A^k C A^k B A^k \in L$$

$$v = A^k B A^k B A^k C A^k \notin L$$

But

$$F_{k,t}(\rtimes u \ltimes) = F_{k,t}(\rtimes v \ltimes)$$

Moving on to the Piecewise Branch



Proper inclusion relationships among subregular language classes (indicated from top to bottom). This paper establishes the TSL class and its place in the figure.

TSL	Tier-based Strictly Local
LTT	Locally Threshold Testable
LT	Locally Testable

PT	Piecewise Testable
SL	Strictly Local
SP	Strictly Piecewise

Defining Subsequences for the Piecewise Branch

Definition

String $u = \sigma_1 \cdots \sigma_n$ is a *subsequence* of string w iff

$$w \in \Sigma^* \sigma_1 \Sigma^* \sigma_2 \Sigma^* \cdots \Sigma^* \sigma_n \Sigma^*$$

If also $|u| = k$ then u is a *k-subsequence* of w.

Example: *ab* is a 2-subsequence of *bacccbb*.

Subsequence Functions

The function $P_{\leq k}$ maps words to the set of k-susbequences within them.

$$P_{\leq k}(w) = \{u : u \text{ is a } k\text{-factor of } w\} \tag{3}$$

Example: $P_{\leq 2}(abcd) = \{ab, ac, ad, bc, bd, cd\}$.

Defining Strictly Piecewise

Definition

A language L is *Strictly Piecewise* (SP) iff there exists a $k \in \mathbb{N}$ and a finite set $S \subseteq P_{\leq k}(\Sigma^*)$ such that

$$L = \{w \in \Sigma^* : P_{\leq k}(w) \subseteq S\}$$

Example: Let $\Sigma = \{a, b\}$. Then $\overline{\Sigma^* b \Sigma^* b \Sigma^*}$ is SP. Let $k = 2$ and $S = \{aa, ab, ba\}$. The subsequence bb is $forbidden$.

Defining Piecewise Testable

Definition

A language L is *Piecewise Testable* (PT) iff there exists some $k \in \mathbb{N}$ such that for all $u, v \in \Sigma^*$:

$$P_{\leq k}(u) = P_{\leq k}(v) \Rightarrow u, v \in L \text{ or } u, v \notin L$$

Robot Claw Language is not PT

Theorem

The Robot Claw language is not PT.

Proof.

Pick any k and threhold t. Let

$$u = A^k(BA^kCA^kBA^kCA^k)^k \in L$$

$$v = A^k(BA^kBA^kCA^kCA^k)^k \notin L$$

But

$$P_{\leq k}(u) = P_{\leq k}(v)$$

Defining Tier-based Strictly Local

1. A *tier* is level of representation where not all alphabetic symbols are present.
2. The idea is "Grab" and "Release" form their own tier and at this level of representation, the "Grab-Grab" and "Release-Release" sequences are forbidden.

Tier-based Factors

The Erasing function:

$$E_T(\sigma_1 \cdots \sigma_n) = u_1 \cdots u_n$$

$$\text{where } u_i = \sigma_i \text{ iff } \sigma_i \in T \text{ and } u_i = \lambda \text{ otherwise}$$

Example: if $\Sigma = \{a, b, c\}$ and $T = \{b, c\}$ then $E_T(aabaaacaaabaa) = bcb$.

Definition

A string $u = \sigma_1 \cdots \sigma_n \in \rtimes T^* \ltimes$ is a *factor on tier* T of a string w iff u is a factor of $E_T(w)$.

Definition of Tier-based SL

Definition

A language L is *Tier-based Strictly Local* iff there exists a tier $k \in \mathbb{N}$, $T \subseteq \Sigma$ and finite set $S \subseteq F_k(\rtimes T^* \ltimes)$ such that

$$L = \{w \in \Sigma^* : F_k(\rtimes E_T(w) \ltimes) \subseteq S\}$$

Robot Claw Language is TSL

A	move
B	grab
C	release

Let $\Sigma = \{A, B, C\}$ and $T = \{B, C\}$.

Let the *permissible* tier-based factors be $S = \{\rtimes B, BC, CB, B\ltimes, C\ltimes\}$. Thus the *forbidden* tier-based factors are $\{\rtimes C, BB, CC\}$.

TSL languages in terms of forbidden factors

Definition

Let the *tier-based container* of $w \in \rtimes T^* \ltimes$ be $C_T(w) =$

$$\{u \in \Sigma^* : w \text{ is a factor on tier } T \text{ of } \rtimes u \ltimes\}$$

For example, $C_T(\rtimes b) = (\Sigma - T)^* b \Sigma^*$.

Theorem

For any $L \in TSL$, let T, k, S be the tier, length, and permissible factors, resp., and $\bar{S}$ the forbidden factors. Then $L = \bigcap_{w \in \bar{S}} \overline{C_T(w)}$.

TSL is neither LTT nor PT

Theorem

TSL is neither LTT nor PT.

Proof.

The Robot Claw language is TSL, but is neither LTT nor PT. □

TSL is star-free

Theorem

TSL is star-free.

Proof.

(sketch). By previous Theorem there exists a finite set $\bar{S}, T, k$ such that $L = \cap_{w \in \bar{S}} \overline{C_T(w)}$. Since the star-free languages are closed under finite intersection and complement, it is sufficient to show that $C_T(w)$ is star-free for all $w \in \rtimes T^* \ltimes$.
First consider any $w = \sigma_1 \cdots \sigma_n \in T^*$. $C_T(w) =$

$$\Sigma^* \sigma_1 (\Sigma - T)^* \sigma_2 (\Sigma - T)^* \cdots (\Sigma - T)^* \sigma_n \Sigma^*$$

Since $(\Sigma - T)^* = \overline{\Sigma^* T \Sigma^*}$ and $\Sigma^* = \overline{\emptyset}$ the set $C_T(w)$ can be written as

$$\overline{\emptyset}\ \overline{\overline{\emptyset} T \overline{\emptyset}}\ \sigma_1\ \overline{\overline{\emptyset} T \overline{\emptyset}}\ \sigma_2\ \overline{\overline{\emptyset} T \overline{\emptyset}}\ \cdots\ \sigma_n\ \overline{\emptyset}$$

Properties of TSL

The choice of $T \subseteq \Sigma$ and $k \in \mathbb{N}$ define systematic classes of languages which are TSL. Let $\mathcal{L}_{T,k}$ denote such a class.

1. Automata represenations of $\mathcal{L}_{T,k}$ are straightforward.
2. Intersections within the class are (sub?)linear (Heinz 2010).
3. $\mathcal{L}_{T,k}$ is a lattice-structured class and therefore efficiently learnable under the most difficult definitions of learning (Heinz 2010, Kasprzik and Kötzing 2010).

Summary

- Better characterizing the instance space of the problem leads to better solutions!

Further Questions

1. Is TSL sufficiently expressive for all formal languages relevant to robotics?
2. How expensive is intersection across $\mathcal{L}_{T,k}$ classes?
3. For known k, but unknown T, can TSL languages be learned?

THANK YOU