

Formal Language Aspects of Identifiable Language Classes

Henning Fernau

School of Computer Science and Software Engineering
The University of Newcastle
fernau@cs.newcastle.edu.au

Wilhelm-Schickard-Institut für Informatik
Universität Tübingen
fernau@informatik.uni-tuebingen.de

October 15, 2003

Overview

- Learning approaches and models
 - Gold-style learning
 - Subclasses of regular languages
- Identifiability results
 - Characterizations
 - Efficient algorithms
 - Closure properties; Descriptive complexity

Machine Learning Approaches

- **deterministic**
- heuristic
- stochastic
- neural networks
- ...

Models...

- Identification in the limit (Gold I&C 1967)
- Probably approximately correct (PAC) learning (Valiant CACM 1984)
- Minimally adequate teacher (MAT) learning (Angluin I&C 1987)
- Robot environment detection (Rivest, Schapire I&C 1993)
- ...

... and Material

- **Formal languages**, actually: automata & grammars, more specifically:
 - **string languages**
 - tree languages
 - ω -languages
 - picture languages
- sets/sequences of numbers
- programs

Identification in the limit

(from positive samples)

Fix language class \mathcal{L} and description formalism \mathcal{D} .

To the **inference machine** IM, a language $L \in \mathcal{L}$ is enumerated, i.e.,

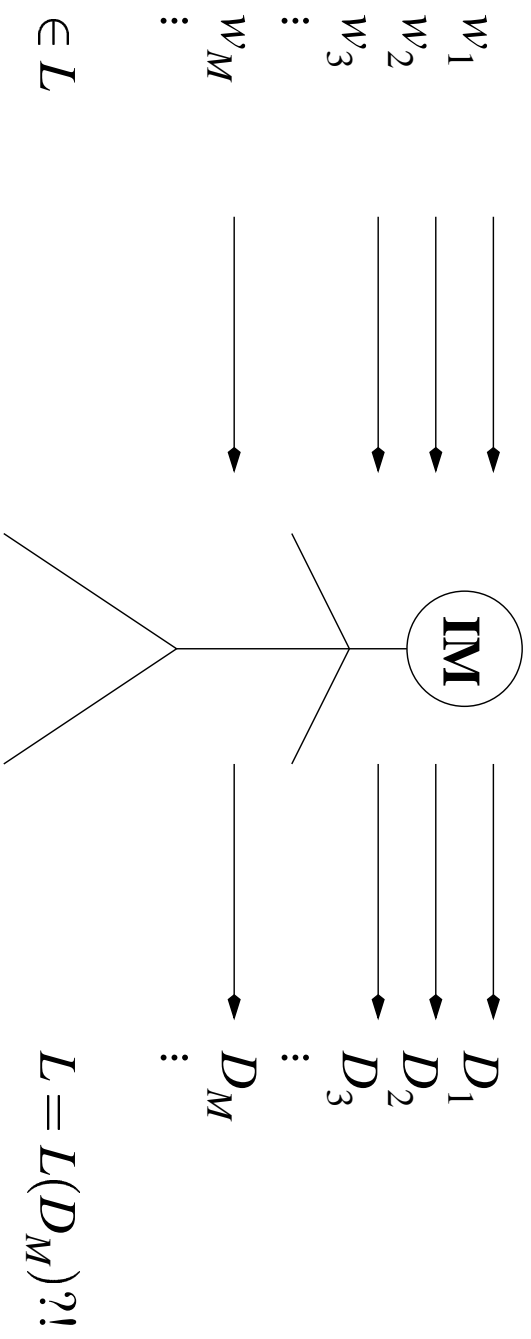
$$L = \{w_n \mid n \geq 1\},$$

and w_n is given to IM at time step n .

On receiving w_n , IM responds with a **hypothesis description** $D_n \in \mathcal{D}$.

IM is a **learner for \mathcal{L}** if the described process always converges, i.e., for all languages $L \in \mathcal{L}$ and all enumerations $(w_n \mid n \geq 1)$ of L , there is an n_0 such that, for all $n \geq n_0$, we find $D_{n_0} = D_n$; moreover, $L(D_{n_0}) = L$.

Language Identification (Positive Samples)



Discussion of the Model

- very natural IDEA: arrive at a concept by looking at examples
- not using negative examples
- not depending on “helpful environment”
- quite “weak”
- very practical: XML DTDs, Spoken Dialog (Telstra)

Known Facts on Language Identification

Theorem 1 (Gold 1967) If \mathcal{L} is **superfinitely**, it is not identifiable.

Cor. 2 REG cannot be identified in the limit from positive samples only.

Identifiable Language Classes

- Pattern languages (Angluin 1980)
- **Reversible languages** (Angluin 1982)
- **Terminal distinguishable languages** (Radhakrishnan & Nagaraja 1987)
- **Function distinguishable languages** (F)
- ...

A simple example: k -gram approach

A quite “practical” example!

```
FOR each word  $w \in I_+$  DO
  IF  $|w| < k$  THEN output  $w$ 
  ELSE FOR each subword  $v$  of  $w$  of length  $k$ 
    DO output  $(a|b)^*v(a|b)^*$ 
```

Is this a “Gold-style learner”?

Problems with “heuristics”

1. not order independent
2. no “normal form type” hypothesis space

A “brute-force” solution in the simple example

REs over Σ : lexicographically ordered

k-gram learner:

On input

$$I_+ = \{w_1, \dots, w_m\} \subset \Sigma^*$$

1. extract $S(k)$, “the” *k*-letter subwords from I_+ ,
2. output “smallest” RE describing $S(k)$.

In general: more “elegant” solutions sought for!

A plethora of “formal language tasks”

Try to characterize suggested / used “heuristics” for language learning (especially of regular and of context-free languages).

Warning: some of these tasks come “in disguise,” suggesting

- generalizers for “patterns” (mostly RE-based) or
- “XML DTDs” (RE-based or CF-based or tree language based).

Function Distinguishability

Distinguishable Functions & Languages

Let F be some finite set. A mapping $f : T^* \rightarrow F$ is called **distinguishing function** if

$f(w) = f(z)$ implies $f(wu) = f(zu)$ for all $u, w, z \in T^*$.

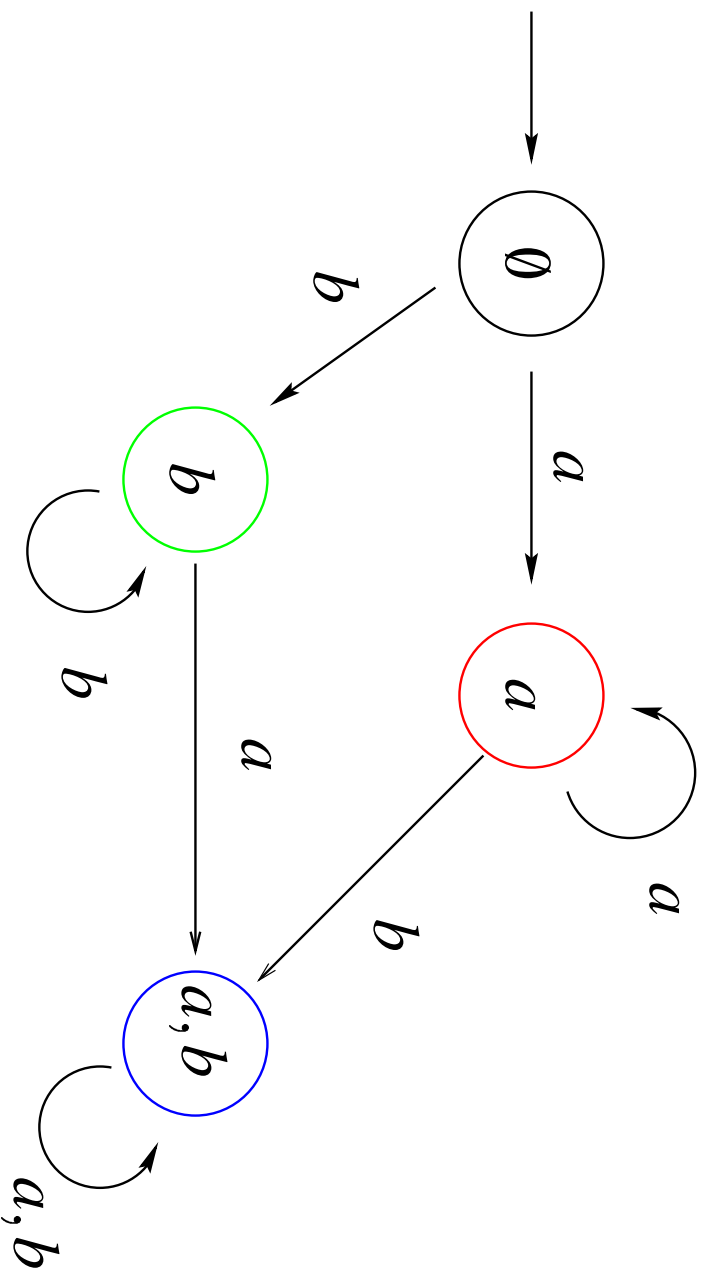
$L \subseteq T^*$ is called f -distinguishable iff, for all $u, v, w, z \in T^*$ with $f(w) = f(z)$, we have $zu \in L \iff zv \in L$ whenever $\{wu, wv\} \subseteq L$.

Language class: f -DL.

Examples for f -Distinguishing Functions & Remarks:

- $f(x) = \text{Ter}(x) = \{a \in T \mid \exists u, v \in T^* : uav = x\}$
 \rightsquigarrow (reversals of) **terminal-distinguishable languages**
- $f(x) = \text{suffix of length } k \text{ of } x \rightsquigarrow$ **k -reversible languages**
- f -distinguishing function yields finite automaton A_f
- finite automaton yields distinguishing function

The automaton belonging to Ter



Distinguishable Automata

Let $A = (Q, T, \delta, q_0, Q_F)$ be a finite automaton. Let $f : T^* \rightarrow F$ be a distinguishing function. A is called **f -distinguishable** if:

1. A is deterministic.

2. For all states $q \in Q$ and all $x, y \in T^*$ with $\delta^*(q_0, x) = \delta^*(q_0, y) = q$, we have $f(x) = f(y)$.

(For $q \in Q$, $f(q) := f(x)$ for some x with $\delta^*(q_0, x) = q$ is well-defined.)

3. For all $q_1, q_2 \in Q$, $q_1 \neq q_2$, with either (a) $q_1, q_2 \in Q_F$ or (b) there exist $q_3 \in Q$ and $a \in T$ with $\delta(q_1, a) = \delta(q_2, a) = q_3$, we have $f(q_1) \neq f(q_2)$.

Canonical Objects

Recall: the minimal DFA $A(L) = (Q, T, \delta, q_0, Q_F)$ can be described as follows:

$Q = \{u^{-1}L \mid u \in \text{Pref}(L)\}$ with $u^{-1}L = \{v \in T^* \mid uv \in L\}$;

$q_0 = \lambda^{-1}L = L$; $Q_F = \{u^{-1}L \mid u \in L\}$; and

$\delta(u^{-1}L, a) = (ua)^{-1}L$ with $u, ua \in \text{Pref}(L)$, $a \in T$.

Let $f : T^* \rightarrow F$ be a distinguishing function and let $L \subseteq T^*$ regular.

$A(L, f)$: the stripped version of the product automaton $A(L) \times A_f$.

$A(L, f)$ is called *f-canonical automaton* of L .

Characterization Theorem

The following conditions are equivalent for a regular language $L \subseteq T^*$ and a distinguishing function $f : T^* \rightarrow F$:

1. L is f -distinguishable.
2. The f -canonical automaton of L is f -distinguishable.
3. L is accepted by an f -distinguishable automaton.
4. For all $u_1, u_2, v \in T^*$ with $f(u_1) = f(u_2)$, $u_1^{-1}L = u_2^{-1}L$ when $\{u_1v, u_2v\} \subseteq L$.

Inferability

For f -DL and some $L \in f$ -DL, consider $A(L, f) = (\mathcal{Q}, T, \delta, q_0, \mathcal{Q}_F)$ and define

$$\begin{aligned} \chi(L, f) = & \{u(q)v(q) \mid q \in \mathcal{Q}\} \\ & \cup \{u(q)av(\delta(q, a)) \mid q \in \mathcal{Q}, a \in T\}, \end{aligned}$$

$u(q), v(q)$ being words of minimal length with $\delta^*(q_0, u(q)) = q$ and $\delta^*(q, v(q)) \in \mathcal{Q}_F$.

Theorem 3 $\chi(L, f)$ is a **characteristic sample** of $L \in f$ -DL.

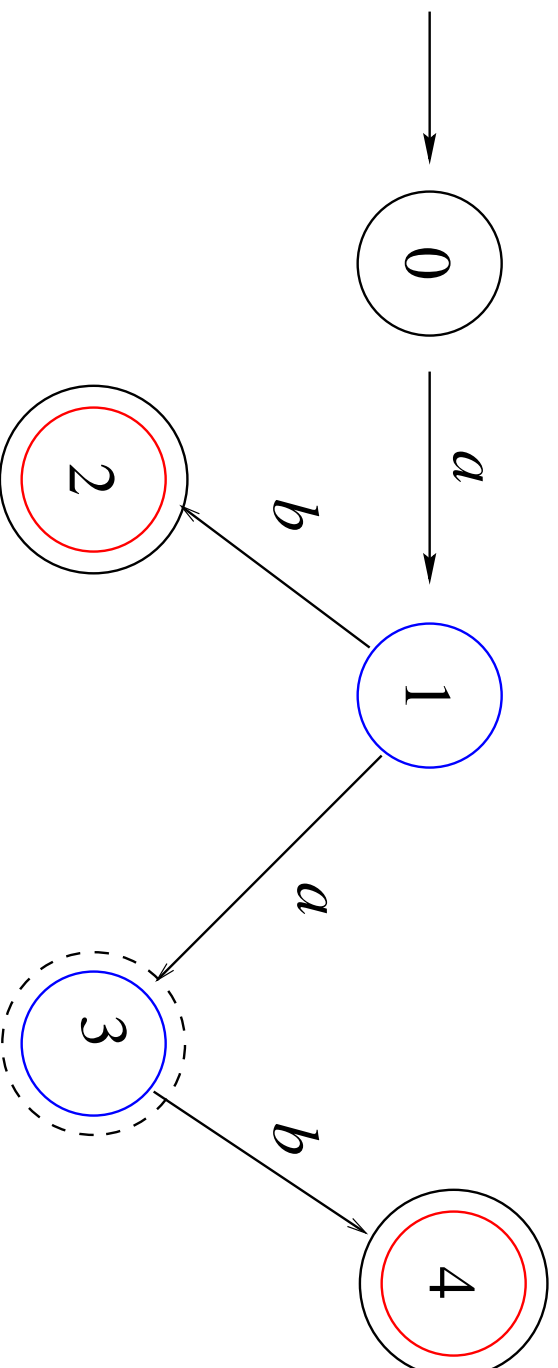
The Merging State Algorithm: Where we start

The **prefix tree acceptor** $PTA(I_+)$ = (Q, T, δ, q_0, Q_F) of a finite sample set $I_+ = \{w_1, \dots, w_M\} \subset T^*$ is a DFA defined as:

$Q = \text{Pref}(I_+)$, $q_0 = \lambda$, $Q_F = I_+$ and $\delta(v, a) = va$ for $va \in \text{Pref}(I_+)$.

Start with A_0 , the stripped version of $PTA(I_+) \times A_f$!

Sample inputs: ab , aab (and aa) (Building PTA)



$\text{Ter}(0) = \emptyset$, $\text{Ter}(1) = \text{Ter}(3) = \{a\}$, $\text{Ter}(2) = \text{Ter}(4) = \{a, b\}$

Merging state algorithm:

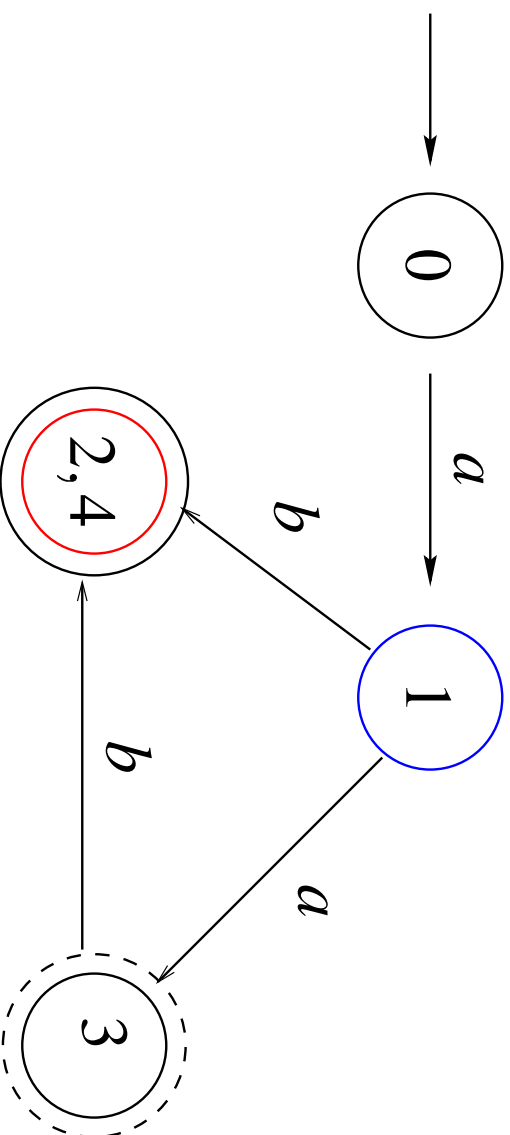
Rule of thumb: Merge “conflicting states”

Here f -Ident: Merge q_1, q_2 if they cause

- nondeterminism or
- “backward f -nondeterminism”.

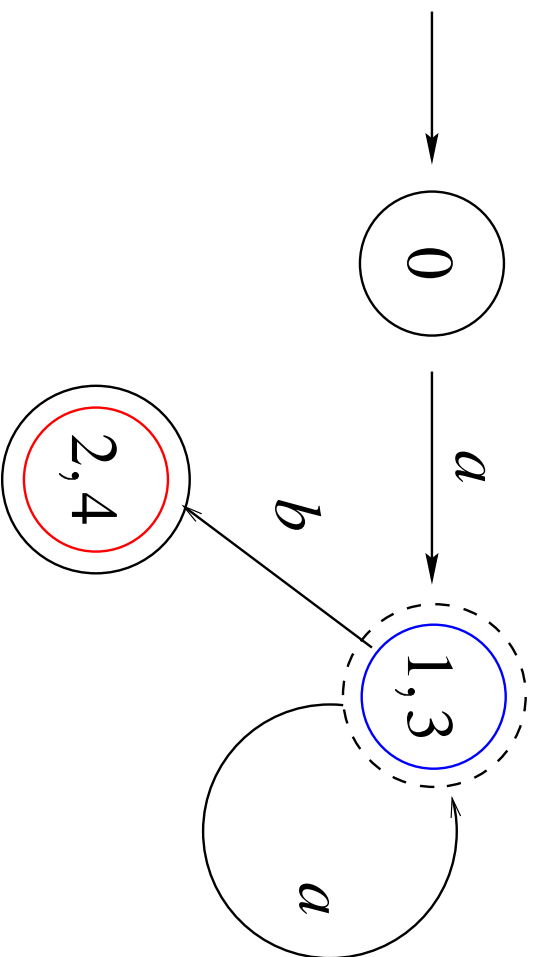
In this way, we get a chain of automata A_0, A_1, \dots, A_f .

Merging final states



$$\text{Ter}(2,4) = \{a, b\}$$

Merging internal states



$$\text{Ter}(1,3) = \{a\}$$

Formal language peculiarities: closure properties

1. non-closure is usually hard to show for a language class
2. due to its **algorithmic nature**, this is easy for inferable language classes.

Example

see Proceedings

omission in literature

Regular Patterns

Fix some alphabet Σ with $*$ $\notin \Sigma$.

Interpret any word over $(\Sigma \cup \{*\})$ as **regular pattern**, i.e., alternatively read it

- as special pattern (with different variables per $*$ occurrence)
- as special regular expression (without union and “proper” star)

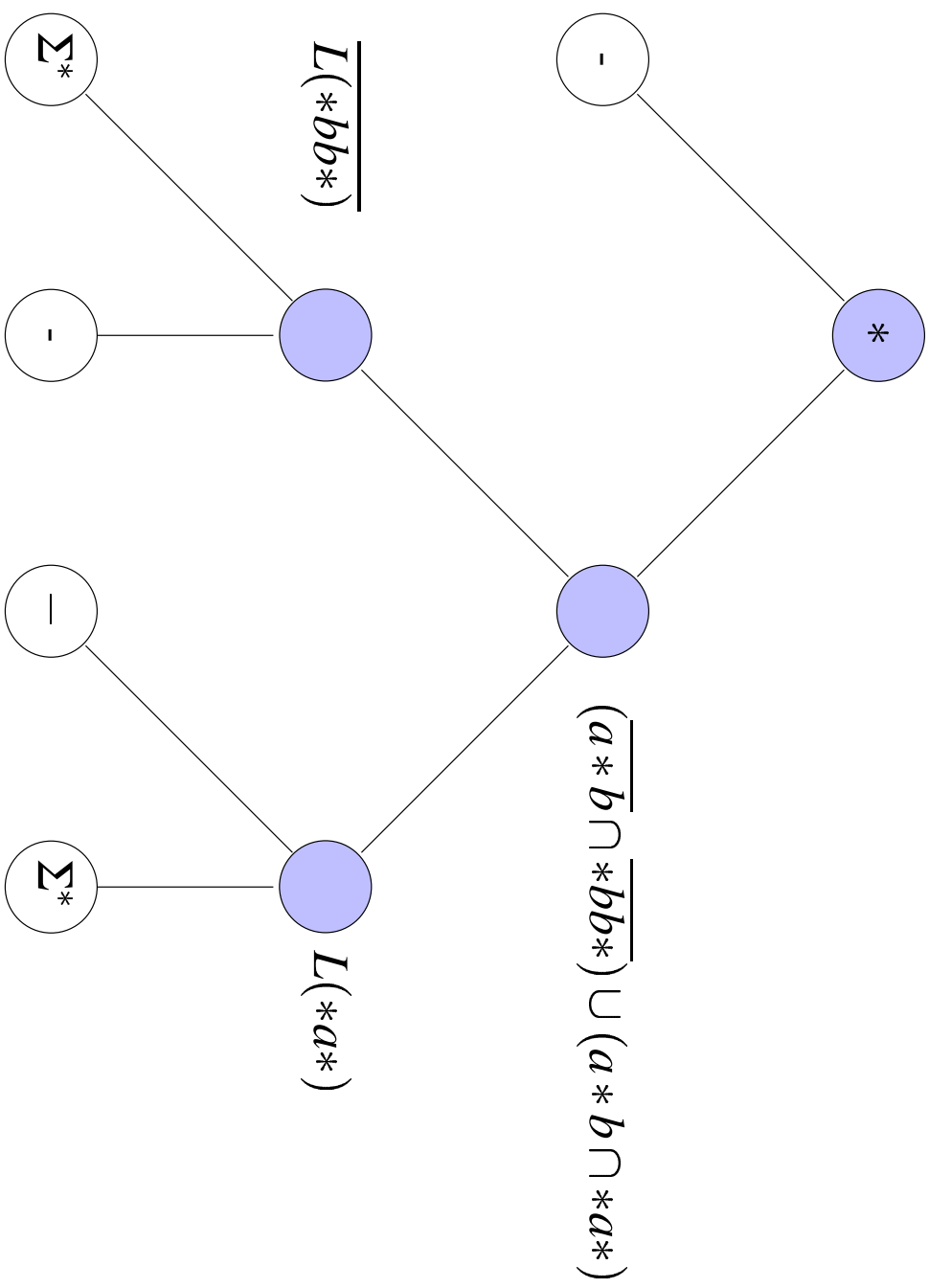
Decision Trees over Regular Patterns

Leaves labeled + and –, inner nodes labeled with regular patterns.

Such a decision tree T **accepts** the following language $L(T)$:

- If T has only one node, then $L(T) = \Sigma^*$ or $L(T) = \emptyset$ if label is + or –, resp.
- Otherwise, let α be the label of the root of T , and T_ℓ and T_r be the left and right subtrees of T .

$$L(T) = (\overline{L(\alpha)} \cap L(T_\ell)) \cup (L(\alpha) \cap L(T_r))$$



More on Regular Patterns (Lange/Nessel TCS 2003)

- Characterizable by **decision lists over r.p.**
- **Unknown** **descriptive** complexity trade-off
- language class NOT identifiable, BUT
- decision list r.p. languages of *degree one* which contain a limited number of nodes (patterns) are identifiable.

Outlook

- Analyze heuristics and investigate the corresponding language classes.
- Use formal language ideas to get learnable language classes.
- Are there “deeper” reasons for learnability?