Computational Learning Theory

Learning EFS

Akihiro Yamamoto 山本 章博
http://www.iip.ist.i.kyoto-u.ac.jp/member/akihiro/
akihiro@i.kyoto-u.ac.jp

EFS(cont.)

Definite Clause (Rules) and EFS

- An definite clause is a formula of the form

$$p(\pi_1,\ldots,\pi_n) \leftarrow q_1(\tau_{11},\ldots), q_2(\tau_{21},\ldots),\ldots, q_k(\tau_{k1},\ldots)$$

where $\pi_1, \pi_2,\ldots, \tau_{11},\ldots, \tau_{k1},\ldots$ are patterns. The definite clause is interpreted as

"for any substitution θ, if $(\tau_{11}\theta,\ldots) \in Q_1, (\tau_{21}\theta,\ldots) \in Q_2,\ldots,$ $q_k(\tau_{k1}\theta,\ldots) \in Q_k$ then $(\pi_1\theta, \pi_2\theta,\ldots, \pi_n\theta) \in P$"

  - A clause $p(\pi_1,\ldots,\pi_n) \leftarrow$ which has no conditions is sometimes called a unit clause.

- A finite set of definite clause is called an elementary formal system (EFS). [Smullyan 61]

Examples

Some examples of definite clauses are

$$p(ax) \leftarrow r(x)$$

$$r(b)$$

$$p(axby) \leftarrow r(x), r(y)$$

$$q(ax, by) \leftarrow q(x, y)$$

$$\ldots$$

Example of Proof (1)

$$\dfrac{\dfrac{\leftarrow p(aaabbb) \qquad \dfrac{p(a\mathit{x}b)\leftarrow p(\mathit{x})}{p(aaabbb)\leftarrow p(aabb)}}{\leftarrow p(aabb)} \qquad \dfrac{p(a\mathit{x}b)\leftarrow p(\mathit{x})}{p(aabb)\leftarrow p(ab)}}{\dfrac{\leftarrow p(ab) \qquad \leftarrow p(ab)}{\leftarrow}}$$

$\leftarrow p(aaabbb)$ $\qquad$ $p(a\mathit{x}b)\leftarrow p(\mathit{x})$

$\leftarrow p(aabb)$ $\qquad$ $p(a\mathit{x}b)\leftarrow p(\mathit{x})$

$\leftarrow p(ab)$ $\qquad$ $p(ab)\leftarrow$

$\leftarrow$

Defining a language by proofs

- A ground atomic formula $p(s_1,\ldots,s_n)$ is provable from an EFS S if

  there is a proof which derives an empty clause from is $\leftarrow p(s_1,\ldots,s_n)$ and S.

- We define a language with a proof from an EFS.

  $\mathrm{P} = L(p, S) = \{ s \mid p(s) \text{ is provable from } S\}$

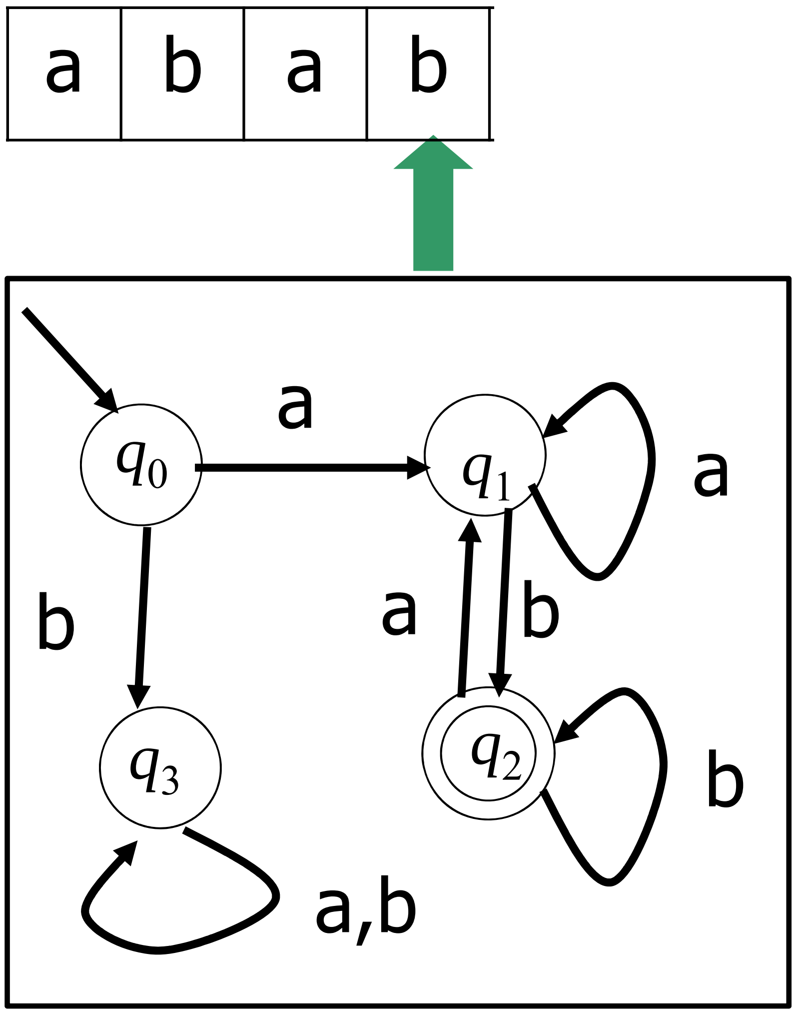
Example

$\mathrm{S} : \mathrm{p}(\mathrm{a}x\mathrm{b}) \leftarrow \mathrm{p}(x)$

$\quad \mathrm{p}(\mathrm{ab}) \leftarrow$

$\mathrm{P} = L(\mathrm{p}, \mathrm{S}) = \{\mathrm{ab}, \mathrm{aabb}, \mathrm{aaabbb}, \mathrm{aaabbb}, \ldots\}$

EFS v.s. FA (1)



$\leftarrow q_0(abab)$ $\quad$ $q_0(ax) \leftarrow q_1(x)$

$\leftarrow q_1(bab)$ $\quad$ $q_1(bx) \leftarrow q_2(x)$

$\leftarrow q_2(ab)$ $\quad$ $q_2(ax) \leftarrow q_1(x)$

$\leftarrow q_1(b)$ $\quad$ $q_1(b) \leftarrow$

$\leftarrow$

EFS v.s. FA (2)

- The EFS which simulates an EFS consists of clauses of the form

$$p(cx) \leftarrow q(x) \qquad c \in \Sigma$$

or

$$p(c) \leftarrow$$

CFG vs. EFS (1)

- Productions and derivation

$$P \to \mathrm{ab},\ \ P \to \mathrm{a}P\mathrm{b}$$

$$P \Rightarrow \mathrm{a}P\mathrm{b} \Rightarrow \mathrm{aa}P\mathrm{bb} \Rightarrow \mathrm{aaabbb}$$

- EFS and proof

$\leftarrow \mathrm{p(aaabbb)}$ $\qquad$ $\mathrm{p(a}x\mathrm{b)} \leftarrow \mathrm{p}(x)$

$\leftarrow \mathrm{p(aabb)}$ $\qquad$ $\mathrm{p(a}x\mathrm{b)} \leftarrow \mathrm{p}(x)$

$\leftarrow \mathrm{p(ab)}$ $\qquad$ $\mathrm{p(ab)} \leftarrow$

$\leftarrow$

CFG vs. EFS (2)

- For every production rule

$$P \rightarrow w_1\, Q_1 w_2\, Q_2 \ldots Q_n w_{n+1} \qquad\qquad P, Q \in N,\ w_i \in \Sigma^*$$

we define a definite clause

$$\mathrm{p}(w_1\, x_1 w_2\, x_2 \ldots x_n w_{n+1}) \leftarrow \mathrm{q}_1(x_1),\ \mathrm{q}_2(x_2), \ldots,\ \mathrm{q}_n(x_n)$$

Example

$$P \rightarrow \mathrm{a}P\mathrm{b} \quad \Rightarrow \quad \mathrm{p}(\mathrm{a}x\mathrm{b}) \leftarrow \mathrm{p}(x)$$

$$P \rightarrow \mathrm{ab} \quad \Rightarrow \quad \mathrm{p}(\mathrm{ab}) \leftarrow$$

Learning EFS

Learning EFS languages

Example 1

C = {aaaab, aaaaaab, aab, b, aaaaaaaab}

D = {a, bbbb, abba, baaaaba, babbb}

Example 2

C = {aabb, aaabbb, ab, aaaabbbb}

D = {a, b, bbbb, abb, baaaaba, babbb}

Example 3

C = {abaabb, aabb, ababaabb,aabbaabb}

D = {a, b, bbbb, abb, baaaaba, babbb}

Learning EFS

- Fix an effective enumeration of EFS on $\Sigma \cup X$:
  $S_1, S_2, \ldots,$

$k = 1, S = S_1$
for $n = 1$ forever
  receive $e_n = \langle s_n , b_n \rangle$
  while ($0 \leq \exists j \leq n$
    $(e_j = \langle s_j , + \rangle$ and $s_j \notin L(S))$ and
    $(e_j = \langle s_j , - \rangle$ and $s_j \in L(S))$
      $S = S'$ for an appropriate S'; k++
  output S

Enumerating EFS

- A simple method to enumerate EFS is just like the enumeration of FA.
- We define the size of an EFS *S* as the total number of symbols in *S* but except "← ", " (", ")"and " , ".

Example size({p(a*x*b)←p(*x*), p(ab)←})= 9

Enumeration of EFS

$\Sigma = \{a, b\}$

size(S)

2 $\{ p(a) \leftarrow \}, \{ p(b) \leftarrow \}, \{ p(x) \leftarrow \}$

3 $\{ p(aa) \leftarrow \}, \{ p(ab) \leftarrow \}, \{ p(ba) \leftarrow \}, \{ p(bb) \leftarrow \},$
$\{ p(xy) \leftarrow \}, \{ p(xx) \leftarrow \}, \{ p(ax) \leftarrow \}, \{ p(bx) \leftarrow \},$
$\{ p(xa) \leftarrow \}, \{ p(xb) \leftarrow \},$

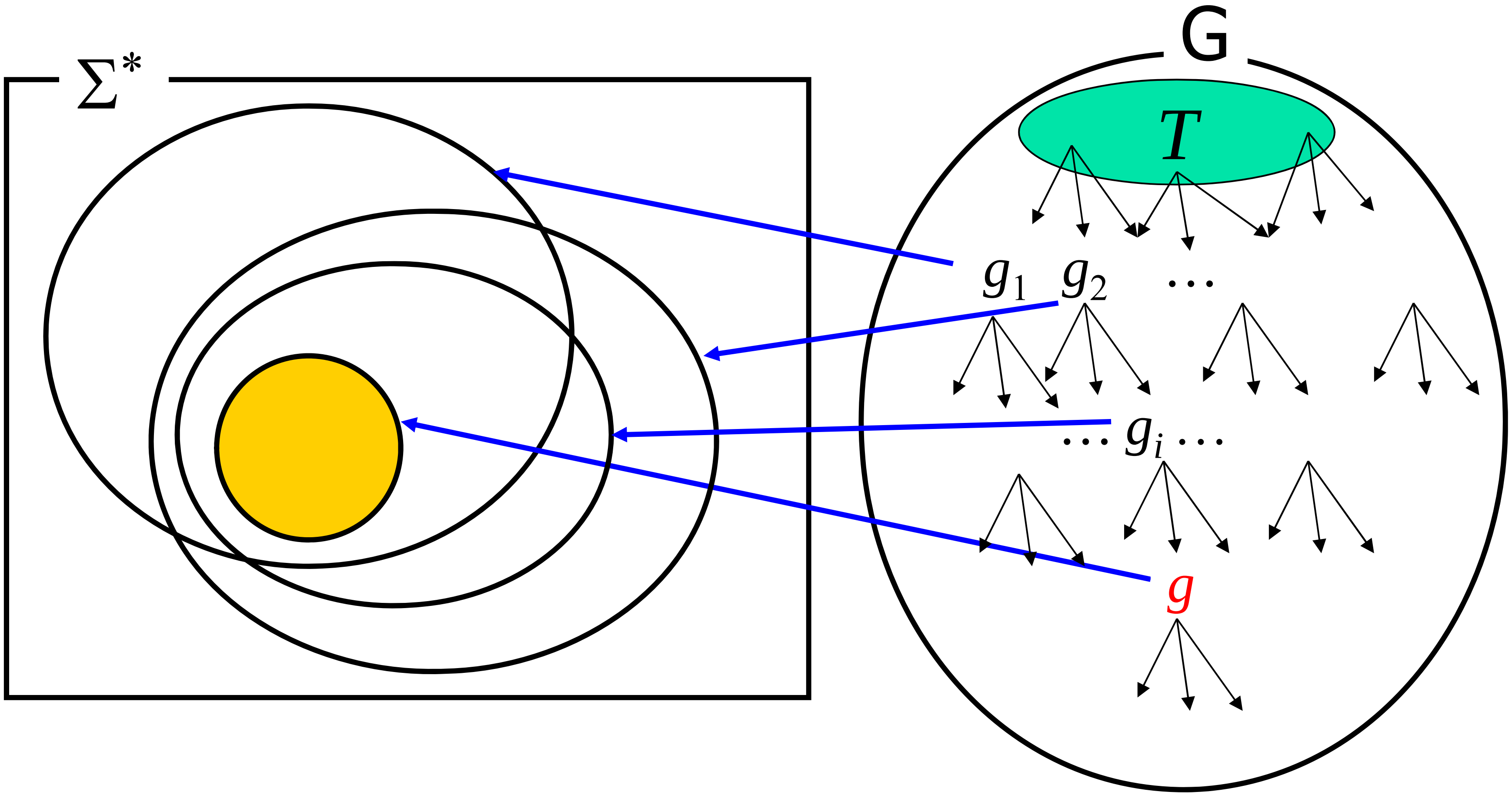
4 $\{p(a) \leftarrow p(a) \}, \ldots, \{p(x) \leftarrow p(x)\},$
$\{p(aaa) \leftarrow \},\ldots,$
$\{p(a) \leftarrow , p(b) \leftarrow\} ,\ldots,\{p(b) \leftarrow , p(x) \leftarrow\}$

…

Refinement Operators

Key idea: Refinement Operator(1)

- Let G be a set of grammar and L(G) be the class of languages represented by grammar in G.



Refinement Operator(2)

- A refinement operator ρ defines, from a given grammar g, set of grammar satisfying:
  1. $\rho(g)$ is recursively enumerable,
  2. for all $h \in \rho(g)$ $L(h) \subseteq L(g)$, and
  3. ere is no sequence $g_1, g_2, \ldots, g_n$ of grammars such that $g_{i+1} = \rho(g_i)$ and $g_1 = g_n$.

Refinement of Patterns

- For patterns on Σ.

$\sigma_x = \{ x := x\, y \}$ where y is a fresh variable

$\theta_{xc} = \{ x := c \}$ where c is in Σ

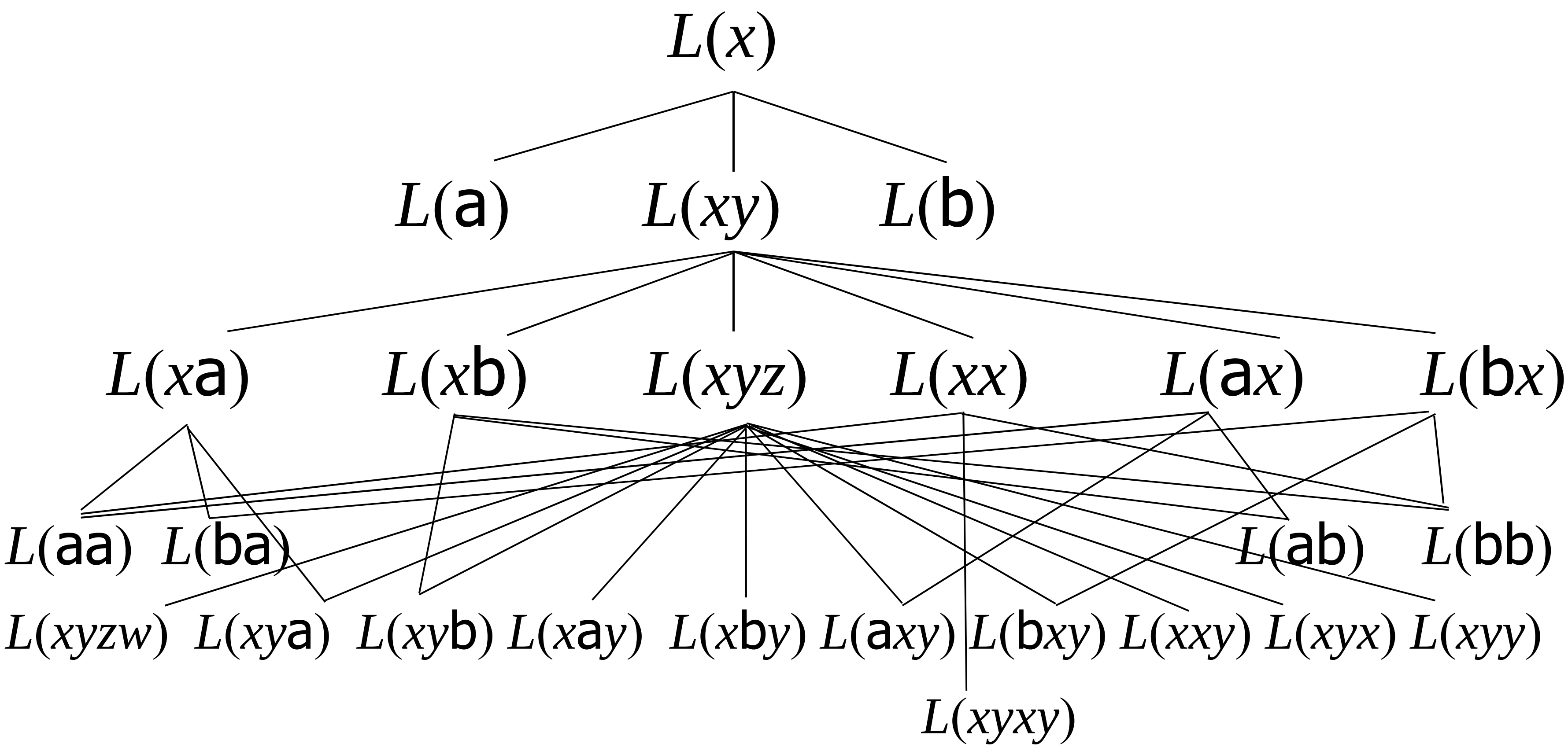
$\delta_{xy} = \{ x := z, y := x \}$ where z is a fresh variable

$\rho(\pi) = \{ \pi\sigma_x \mid x$ is a variable occurring in $\pi \}$

$\cup \{ \pi\theta_{xc} \mid x$ is a variable occurring in π and c is in $\Sigma \}$

$\cup \{ \pi\delta_{xy} \mid x$ and y are variables occurring in $\pi \}$

Hasse Diagram (General Version)



Generating Patterns with Refinement

- Let C be a set of positive examples and D be a set of negative examples.
- Assume the set of variables $X = \{x_1, x_2, \ldots, x_n, \ldots\}$

Let $P := \{ x_1 \}$, $Q := \varnothing$

/* P is for keeping candidates, and Q is for minimal candidates.*/

while $P \neq \varnothing$ **do**

choose π from P

$P' := \varnothing$

for each $\pi' \in \rho(\pi)$

if $C \subseteq L(\pi')$ and $L(\pi') \cap D = \varnothing$

$P' := P' \cup \{\pi'\}$

if $P' = \varnothing$

$Q := Q \cup \{\pi\}$

else

$P := P - \{\pi\} \cup P'$

Some Required Properties

- ρ should be locally finite :

  $\rho(g)$ is a finite and the enumeration of its elements terminates in finite time.

- ρ should be semantically complete :

  For every language $L(h)$ such that $L(h) \subset L(g)$, there is a sequence $g_1, g_2, \ldots, g_n$ such that

  $g_1 = g,\ g_{i+1} = \rho(g_i)\ \ (i = 1, \ldots, n-1)$, and $L(g_n) = L(h)$.

- There exist finitely many maximal grammars:

  For every grammar g, there exits a sequence $g_1, g_2, \ldots, g_n$ such that

  g_1 is maximal, $g_{i+1} = \rho(g_i)\ (i = 1, \ldots, n-1)$, and $g_n = g$.

Refinement of EFS

- Because an EFS is a set of definite clauses, we define the refinement operator for EFSs by
  - defining the refinement of operator of definite clause
  - and then defining the refinement operator of the set of definite clauses.

Refinement of definite clauses

- For a definite clause $C = A \leftarrow B_1,\ldots,, B_n$

$\sigma_x = \{ x := x\, y\}$ where y is a fresh variable

$\theta_{xc} = \{ x := c \}$ where c is in Σ

$\delta_{xy} = \{ x := z, y := x\}$ where z is a fresh variable

$\rho(C) =$ $\{C\, \sigma_x \mid x$ is a variable occurring in $C \}$

$\cup$ $\{C\, \theta_{xc} \mid x$ is a variable occurring in C and c is in $\Sigma \}$

$\cup$ $\{C\, \delta_{xy} \mid x$ and y are variables occurring in $C \}$

$\cup$ $\{ A \leftarrow B_1,\ldots,, B_n, p(x_1,\ldots,x_k) \mid$ where $x_1,\ldots,x_k$ are mutually distinct variables occuring in $A\}$

Refinement of EFS

- For a set S of definite clauses

$$\rho(S) = \{S \cup \{D\} \mid D \in \rho(C) \text{ for some } C \in S\} \cup \{S - \{C\} \mid C \in S\}$$

- The top element is a set of clauses of the form

$$\mathrm{T}: \left\{ \begin{array}{l} p_1(x_1,\ldots,x_{n1}) \leftarrow \\ p_2(x_1,\ldots,x_{n2}) \leftarrow \\ p_3(x_1,\ldots,x_{n3}) \leftarrow \\ \ldots \end{array} \right.$$

- $\rho^n(P)$: The set of EFS which can be obtained by applying ρ repeatedly at most n times.

A successful case

- If we give some restrictions to EFS S, we can simply extend the learning algorithm for patterns.
- An example of such a restriction is:

  The number of definite clauses in S is bounded up to a given N and every clause is of the form

  $$p(\pi_1,\ldots,\pi_n) \leftarrow q_1(x_1), q_2(x_2), \ldots, q_k(x_k)$$

  where $x_1, x_2, \ldots, x_k$ appears in $\pi_1, \ldots, \pi_n$.

- The latter condition is just saying that S corresponds to a CFG.

A key property of refinement

- For EFS S and T,

$$T \in \rho(S) \Rightarrow L(S) \supseteq L(T)$$

- The definition of $\rho(S)$ is rather mathematical, and a more practical method for finding hypotheses can be formalized with not using $\rho(S)$ but $\rho(C)$.
  - Starting with $\top$, if a definite clause C generates any negative example, replace C with all of the clauses in $\rho(C)$.

Learning EFS

$S = \mathsf{T}$

for $n = 1$ forever

receive $e_n = \langle s_n , b_n \rangle$

while ($0 \leq \exists j \leq n \; e_j = \langle s_j , - \rangle$ and $s_j \in L(S)$)

delete a clause C in S and add all clauses

in $\rho(C)$

output S

Example

$S: \ \mathrm{p}(\mathsf{a}x\mathsf{b}) \leftarrow \mathrm{p}(y)$

$\quad \mathrm{p}(\mathsf{ab}) \leftarrow$

$E_1 = \langle \mathrm{p}(\mathsf{aabb}), + \rangle$, $E_2 = \langle \mathrm{p}(\mathsf{ab}), + \rangle$

$E_3 = \langle \mathrm{p}(\mathsf{abb}), - \rangle$

Example

$E_1 = \langle p(\text{aabb}), +\rangle$, $E_2 = \langle p(\text{ab}), +\rangle$

$E_3 = \langle p(\text{bba}), -\rangle$

$S =$ ~~$p(x) \leftarrow$~~

$p(xy) \leftarrow$, $p(\text{a}) \leftarrow$, $p(\text{b}) \leftarrow$, $p(x) \leftarrow p(x)$

Example

$E_1 = \langle p(aabb), +\rangle$, $E_2 = \langle p(ab), +\rangle$

$E_3 = \langle p(bba), -\rangle$

$S =$ ~~$p(x) \leftarrow$~~

~~$p(xy) \leftarrow$~~, $p(a) \leftarrow$, $p(b) \leftarrow$, $p(x) \leftarrow p(x)$

$p(xyz) \leftarrow$, $p(ay) \leftarrow$, $p(by) \leftarrow$, $p(xa) \leftarrow$, $p(xb) \leftarrow$

$p(xy) \leftarrow p(x)$, $p(xy) \leftarrow p(y)$

Example

$E_1 = \langle p(aabb), + \rangle$, $E_2 = \langle p(ab), + \rangle$

$E_3 = \langle p(bba), - \rangle$

$S =$ ~~$p(x) \leftarrow$~~

~~$p(xy) \leftarrow$~~, $p(a) \leftarrow$, $p(b) \leftarrow$, $p(x) \leftarrow p(x)$

~~$p(xyz) \leftarrow$~~, $p(ay) \leftarrow$, $p(by) \leftarrow$, $p(xa) \leftarrow$, $p(xb) \leftarrow$

$p(xy) \leftarrow p(x)$, $p(xy) \leftarrow p(y)$

$p(xxz) \leftarrow$, $p(xyx) \leftarrow$, $p(xyy) \leftarrow$, $p(xyz) \leftarrow p(x)$,

$p(xyz) \leftarrow p(y)$, $p(xyz) \leftarrow p(z)$,

$p(ayz) \leftarrow$, $p(byz) \leftarrow$, …, $p(xya) \leftarrow$, $p(xyb) \leftarrow$,

Example

$E_1 = \langle$p(aabb), $+\rangle$, $E_2 = \langle$p(ab), $+\rangle$

$E_3 = \langle$p(bba), $-\rangle$

$S =$ ~~p(*x*) ←~~

~~p(*xy*)←~~, p(a)←, p(b)←, p(*x*) ← p(*x*)

~~p(*xyz*)←~~, p(a*y*)←, ~~p(b*y*)←~~, p(*x*a)←, p(*x*b)←

p(*xy*)←p(*x*), p(*xy*)←p(*y*)

p(*xxz*)←, p(*xyx*)←, p(*xyy*)←, p(*xyz*)←p(*x*),

p(*xyz*)←p(*y*), p(*xyz*)←p(*z*),

p(a*yz*)←, p(b*yz*)←, …, p(*xy*a)←, p(*xy*b)←,

p(ba)←, p(bb)←, ~~p(b*xy*)←~~, p(b*y*)←p(*y*),

Example

$E_1 = \langle p(aabb), +\rangle$, $E_2 = \langle p(ab), +\rangle$

$E_3 = \langle p(bba), -\rangle$

$S =$ ~~p(*x*) ←~~

~~p(*xy*)←~~, p(a)←, p(b)←, p(*x*) ← p(*x*)

~~p(*xyz*)←~~, ~~p(a*y*)←~~, p(b*y*)←, ~~p(*x*a)←~~, p(*x*b)←

p(*xy*)←p(*x*), p(*xy*)←p(*y*)

p(*xxz*)←, p(*xyx*)←, p(*xyy*)←, p(*xyz*)←p(*x*),

p(*xyz*)←p(*y*), p(*xyz*)←p(*z*),

p(a*yz*)←, p(b*yz*)←, …, p(*xy*a)←, p(*xy*b)←,

p(aa)←, p(ab)←, ~~p(a*xy*)←~~, p(a*y*)←p(*y*),

p(ba)←, ~~p(ab)←~~, ~~p(*xy*a)←~~, p(*x*a)←p(*x*)

Example

$E_1 = \langle p(aabb), +\rangle$, $E_2 = \langle p(ab), +\rangle$

$E_3 = \langle p(bba), -\rangle$

$S =$ $p(a)\leftarrow$, $p(b)\leftarrow$, $p(x) \leftarrow p(x)$

$p(by)\leftarrow$, $p(xa)\leftarrow$, $p(xy)\leftarrow p(x)$, $p(xy)\leftarrow p(y)$

$p(xxz)\leftarrow$, $p(xyx)\leftarrow$, $p(xyy)\leftarrow$, $p(xyz)\leftarrow p(x)$,

$p(xyz)\leftarrow p(y)$, $p(xyz)\leftarrow p(z)$,

$p(ayz)\leftarrow$, $p(byz)\leftarrow$, …, $p(xya)\leftarrow$, $p(xyb)\leftarrow$,

$p(aa)\leftarrow$, $p(ab)\leftarrow$, $p(ay)\leftarrow p(y)$,

$p(bb)\leftarrow$, $p(xyb)\leftarrow$, $p(xb)\leftarrow p(x)$

Example

$E_1 = \langle p(aabb), +\rangle$, $E_2 = \langle p(ab), +\rangle$

$E_3 = \langle p(bba), -\rangle$

$S =$ p(a)←, p(b)←, p(x) ← p(x)

p(by)←, p(xa)←, p(xy)←p(x), p(xy)←p(y)

p(xyx)←, p(xyy)←, p(xyz)←p(x),

p(xyz)←p(y), p(xyz)←p(z),

p(ayz)←, p(byz)←, …, p(xya)←, p(xyb)←,

p(aa)←, p(ab)←, p(ay)←p(y),

p(bb)←, p(xyb)←, p(xb)←p(x)

p(ayy)←, p(byy)←, p(aa$z$)←, p(bb$z$)←, p($xyxyz$)←,

p(xxz)←p(x), p(xxz)←p(z),

Example

$E_1 = \langle p(\mathrm{aabb}), +\rangle$, $E_2 = \langle p(\mathrm{ab}), +\rangle$

$E_3 = \langle p(\mathrm{bba}), -\rangle$

$S =$ $p(\mathrm{a})\leftarrow$, $p(\mathrm{b})\leftarrow$, $p(x) \leftarrow p(x)$

$p(\mathrm{b}y)\leftarrow$, $p(x\mathrm{a})\leftarrow$, $p(xy)\leftarrow p(x)$, $p(xy)\leftarrow p(y)$

$p(xyx)\leftarrow$, $p(xyy)\leftarrow$, $p(xyz)\leftarrow p(x)$,

$p(xyz)\leftarrow p(y)$, $p(xyz)\leftarrow p(z)$,

$p(\mathrm{a}yz)\leftarrow$, $p(\mathrm{b}yz)\leftarrow$, …, $p(xy\mathrm{a})\leftarrow$, $p(xy\mathrm{b})\leftarrow$,

$p(\mathrm{aa})\leftarrow$, $p(\mathrm{ab})\leftarrow$, $p(\mathrm{a}y)\leftarrow p(y)$,

$p(\mathrm{bb})\leftarrow$, $p(xy\mathrm{b})\leftarrow$, $p(x\mathrm{b})\leftarrow p(x)$

$p(\mathrm{a}yy)\leftarrow$, $p(\mathrm{b}yy)\leftarrow$, $p(\mathrm{aa}z)\leftarrow$, $p(\mathrm{bb}z)\leftarrow$, $p(xyxyz)\leftarrow$,

$p(xxz)\leftarrow p(x)$, $p(xxz)\leftarrow p(z)$,

Example

$E_1 = \langle$p(aabb), $+\rangle$, $E_2 = \langle$p(ab), $+\rangle$

$E_3 = \langle$p(bba), $-\rangle$

$S =$ …

p(ayb)←p(y),

p(ab)←,

…

References

- Shapiro, E.Y.: Inductive Inference of Theories From Facts, YALEU / DCS / TR -192 (1981).
- Shapiro, E.Y.: Algorithmic Program Debugging, MIT Press (1983).
- S. Arikawa, T. Shinohara, A. Yamamoto: Learning Elementary Formal Systems, *Theoretical Computer Science*, 95, 97-113, (1992).