Computational Learning Theory

Learning Tree Patterns

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

Contents

- Rooted, Ordered, Labeled and Ranked Trees
- Formal tree languages and tree patterns.
- Learning tree languages from positive data

Formal Languages

- Σ : a finite set of symbols and called an alphabet
- Σ^* : the set of all finite strings consisting of the symbols in Σ.
  - An empty string is denoted by ε.
  - $\Sigma^+ = \Sigma^* - \{\varepsilon\}$
- A formal language L on Σ is a subset of Σ^*.

Example

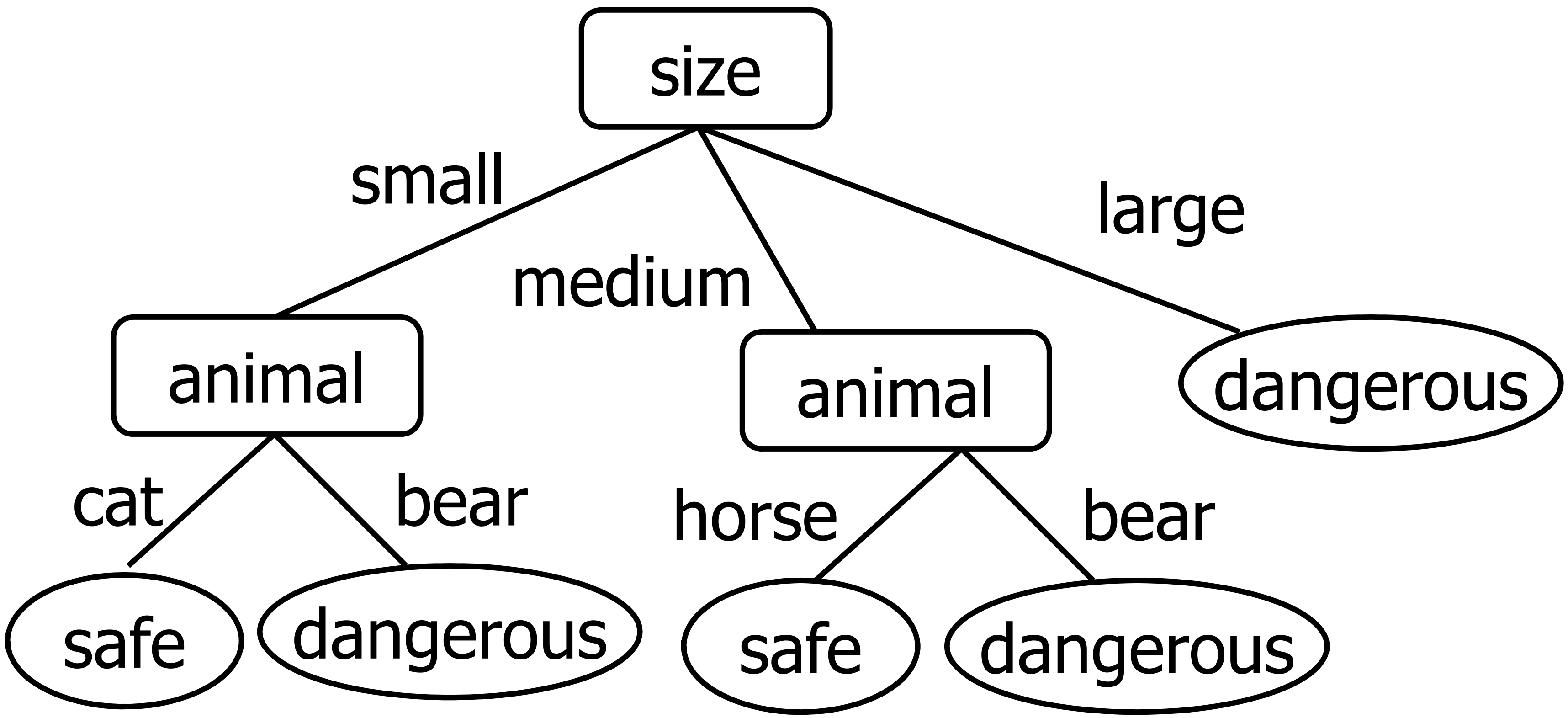
$\Sigma = \{$a, b$\}$

$\Sigma^* = \{\varepsilon$, a, b, aa, ab, ba, bb, aaa, aab,… $\}$

$L = \{$aab, abb, aaab, aabb, abab, abbb,… $\}$

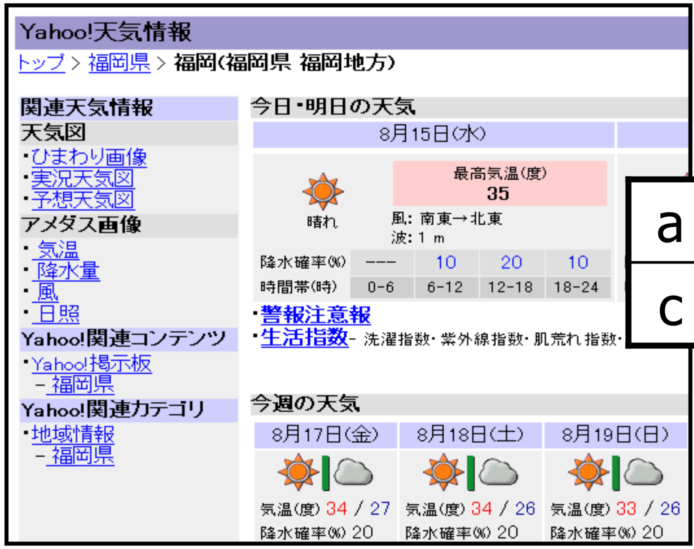
Trees (1)

- Trees are very popular data structure in computer science.



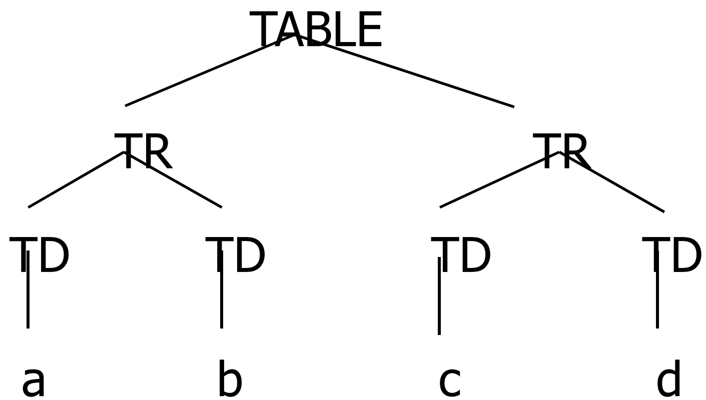
Trees (2)

- Trees are very popular data structure in computer science.



a	b
c	d

<TABLE>
<TR>
<TD>a</TD>
<TD>b</TD>
</TR>
<TR>
<TD>c </TD>
<TD>d</TD>
</TR>
</TABLE>



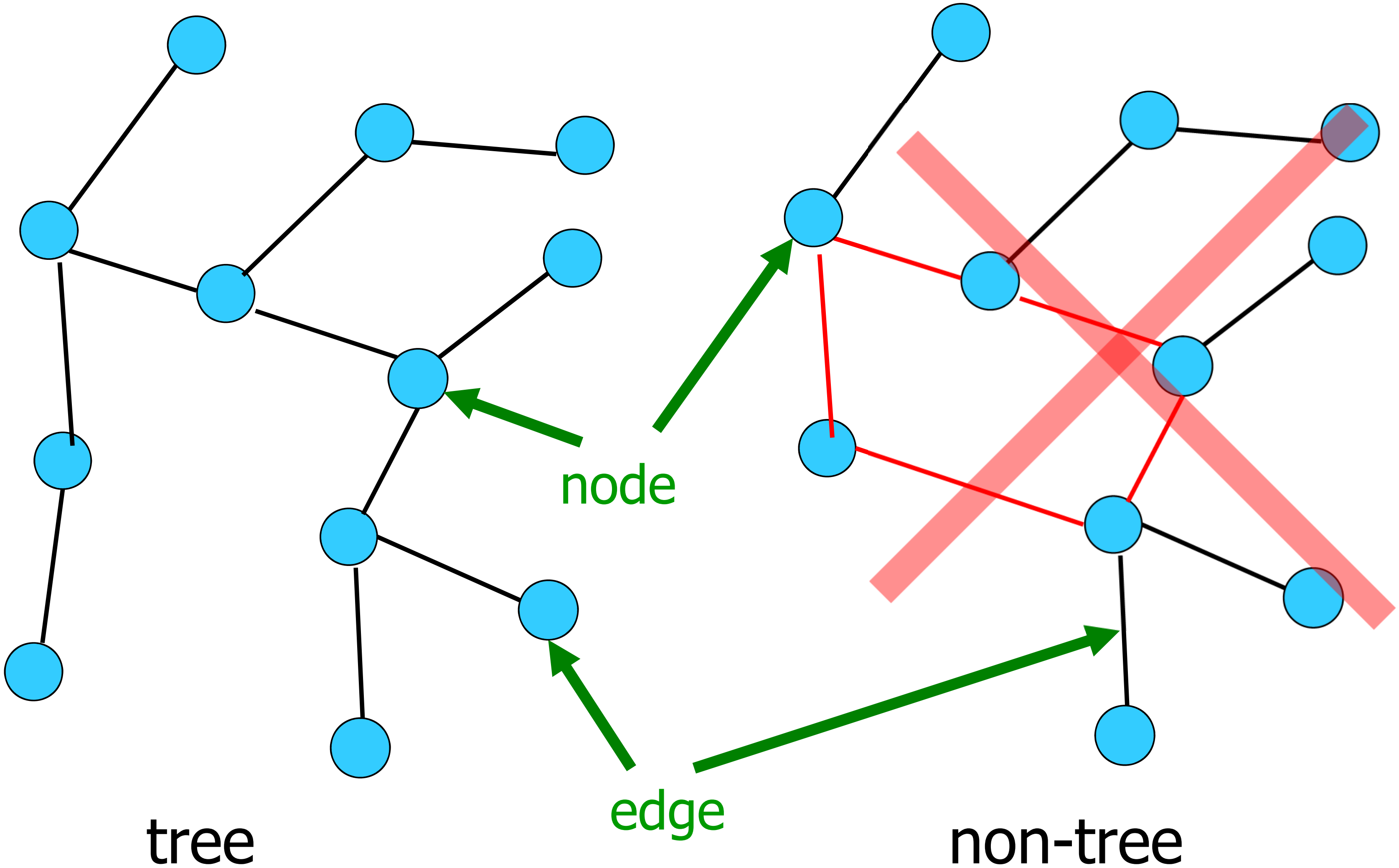
Typical Usage of Trees

- We introduced two types of usage of trees:
  - guidance of activity, e.g. representing classification, representing classification, and so on.
  - data with structure, e.g. parsing trees (results of parsing) …

- In this lecture we treat “tree” as data.

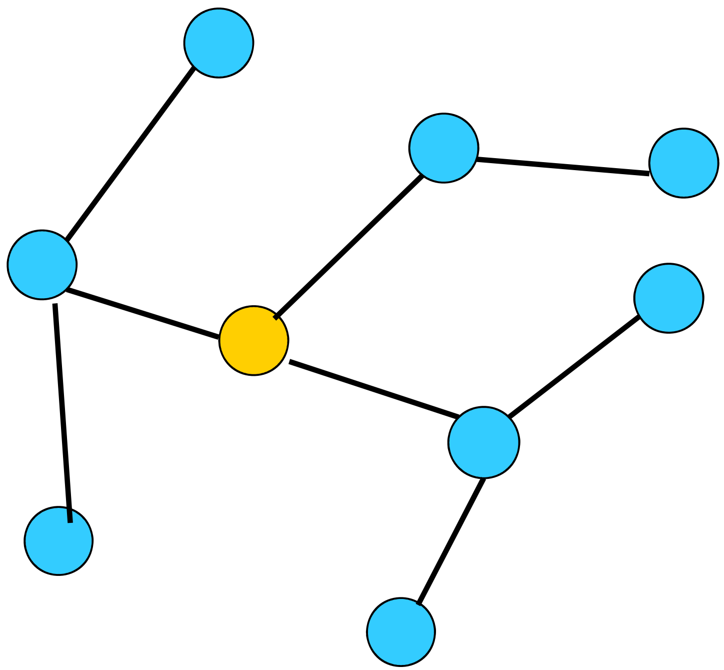
Definition of Trees

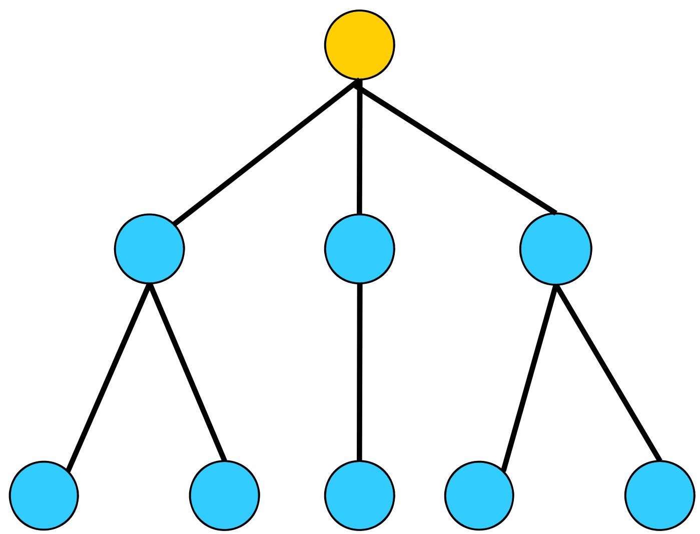
- Generally a tree is an acyclic graph.



Classes of trees (1)

- If a node in a tree T is selected, T is called a rooted tree.
- A rooted tree is regarded as a directed graph by giving the direction to every edge so that we can reach every leaf from the root.

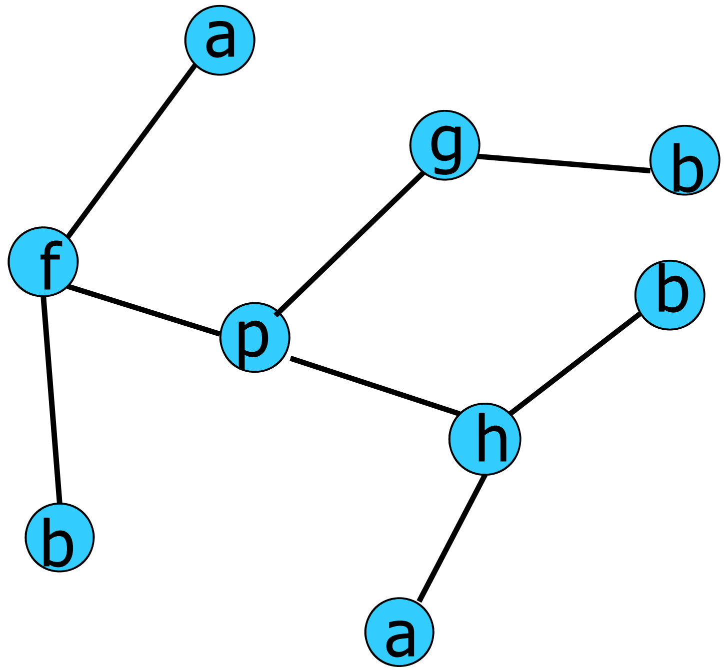




Classes of trees (2)

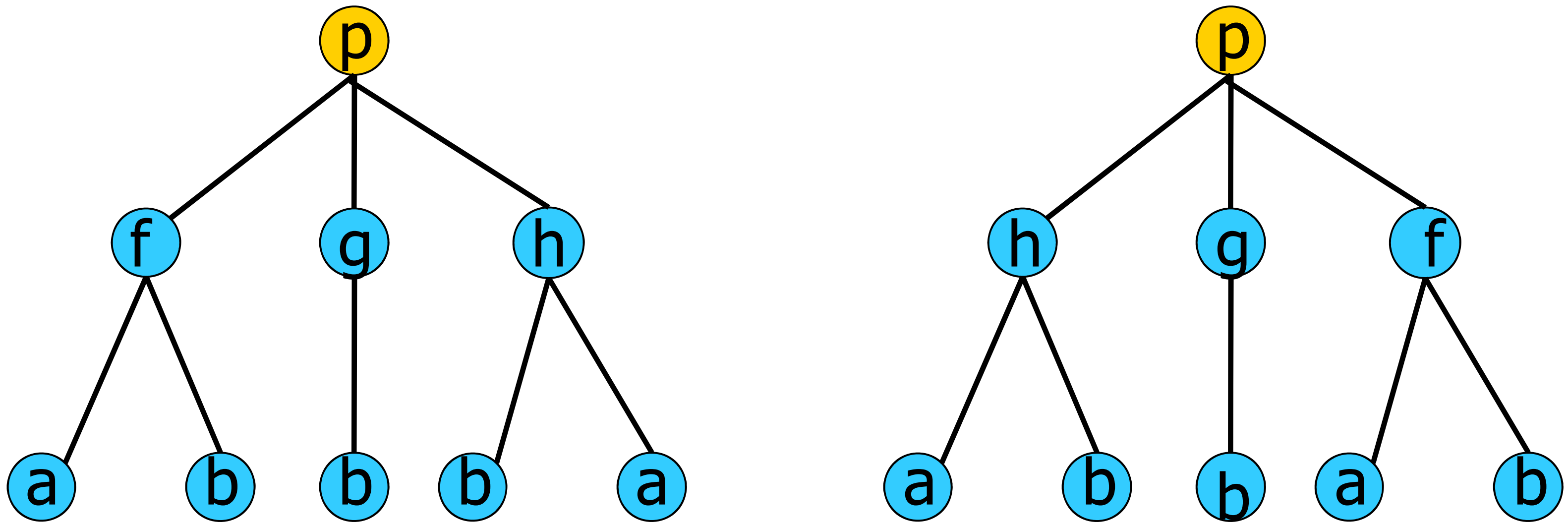
- Let Σ be a finite set of symbols and called an alphabet.
- A tree T is labeled if an label is attached to every node of a tree T.

$$\Sigma = \{a, b, f, g, h, p\}$$



Classes of trees (3)

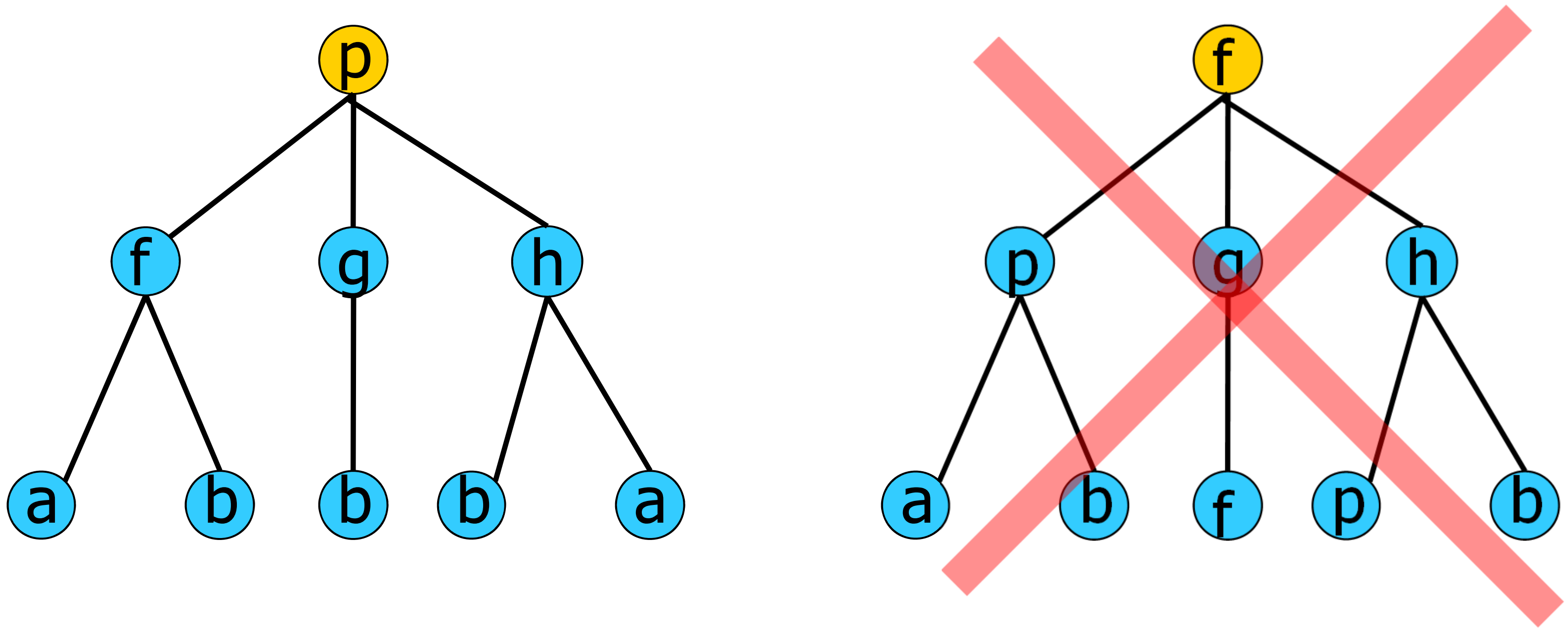
- A rooted and labeled tree *T* is ordered if, for every non-leaf node in *T*, an ordering is given to the set of its children.
  - That is, for every non-leaf node, the first child, the second child, … are defined.



Classes of trees (4)

- A rooted and labeled tree T is ranked if, for every non-leaf node in T, the number of its children is fixed by tha label attached to it.

$\Sigma = \{$a/0, b/0, f/2, g/1, h/2, p/3$\}$

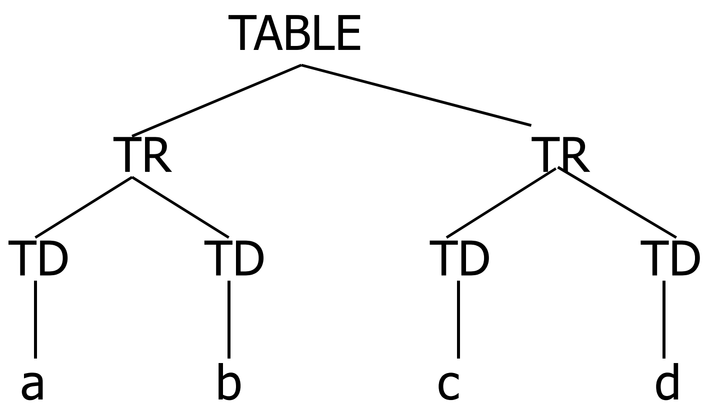


The class treated in this lecture

- We treat rooted, labeled, ordered, and ranked trees.
  - From now, a “tree” means that a rooted, labeled, ordered, and ranked tree.

Expressing Trees

- The relation between HTML expressions and dom trees shows that a tree is represented as an expression.



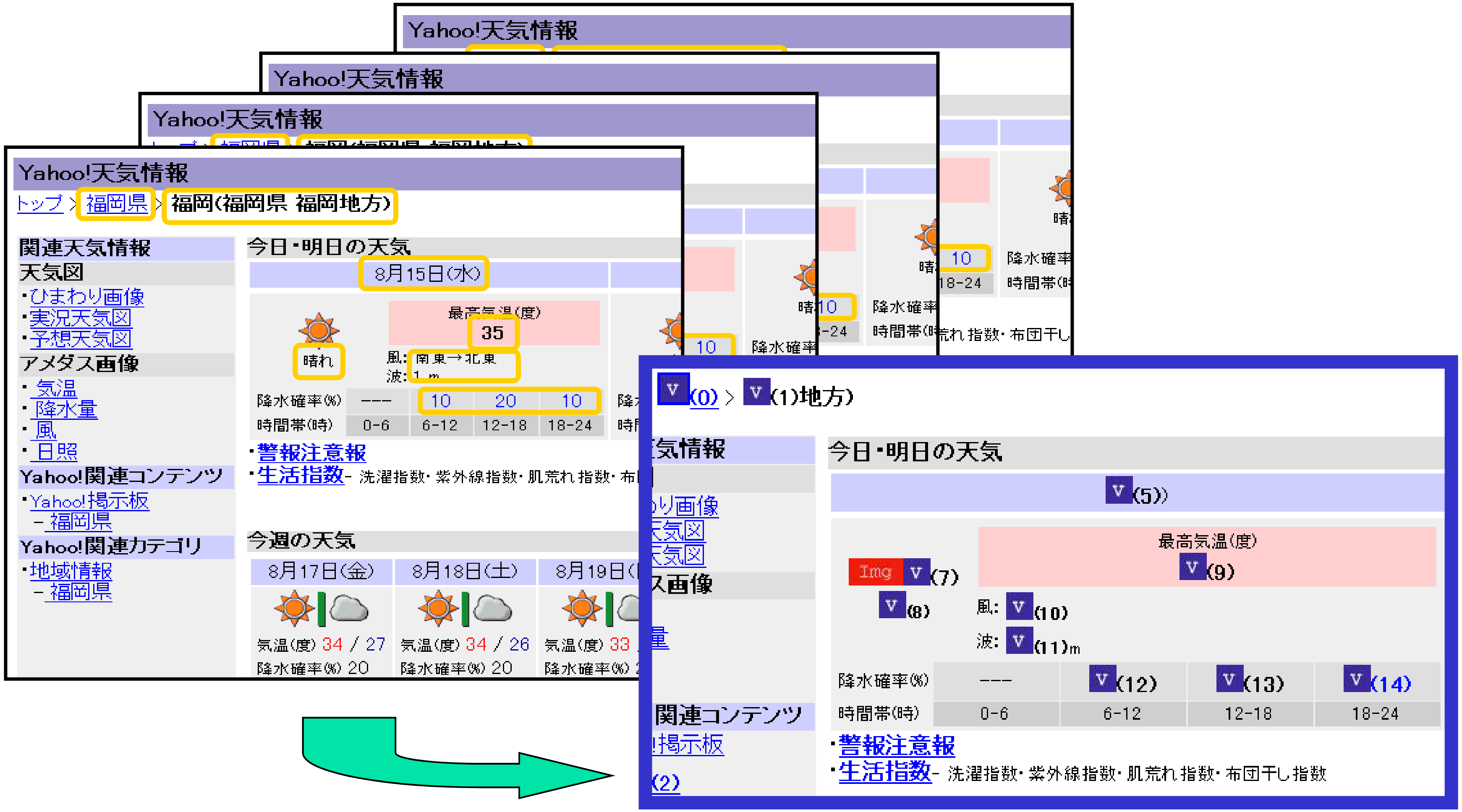

```
<TABLE>
                    <TR>
          <TD>a</TD>
<TD>b</TD>
                    </TR>
                    <TR>
          <TD>c </TD>
<TD>d</TD>
                    </TR>
 </TABLE>
```

- With referring mathematical logic, we use simpler expressions.

TABLE(TR(TD(a), TD(b)), TR(TD(c), TD(d)))

Application of Tree Patterns

- Extracting common “structure” from tree data



Formal Tree Languages

- Σ : a finite set of symbols and called an alphabet
  - To each symbol a non-negative integer called its rank is attached.
- T(Σ) : the set of all rooted, labeled, ordered, and ranked trees consisting of the symbols in Σ.
- A formal tree language *L* on Σ is a subset of T(Σ).

Examples

Σ = {a/0, f/2}

T(Σ) = {a, f(a,a), f(f(a,a),a), f(a,f(a,a)), f(f(a,a), f(a,a)),□ }

Σ = {z/0, s/1, p/3}

T(Σ) = {z, s(z), s(s(z)), p(z,z,z), p(s(z),z,z), s(p(z,z,z)),...}

Tree Patterns

- Let X be a countable set of variables
  - Assuming $\Sigma \cap X = \varnothing$
  - and rank(x) = 0 for every variable.
- A tree pattern π is an element of TL($\Sigma \cup X$)

Example

$\Sigma = \{a/0, f/2\}, X = \{x/0, y/0, \ldots\}$

a, f(x,a), f(f(a, x),a), f(f(a, x), y), f(x,f(a, x)),...

We sometime assume that every variable in a pattern is indexed, in the ordering of its first occurrence.

$\Sigma = \{a/0, f/2\}, X = \{x_1/0, x_2/0, x_3/0, \ldots\}$

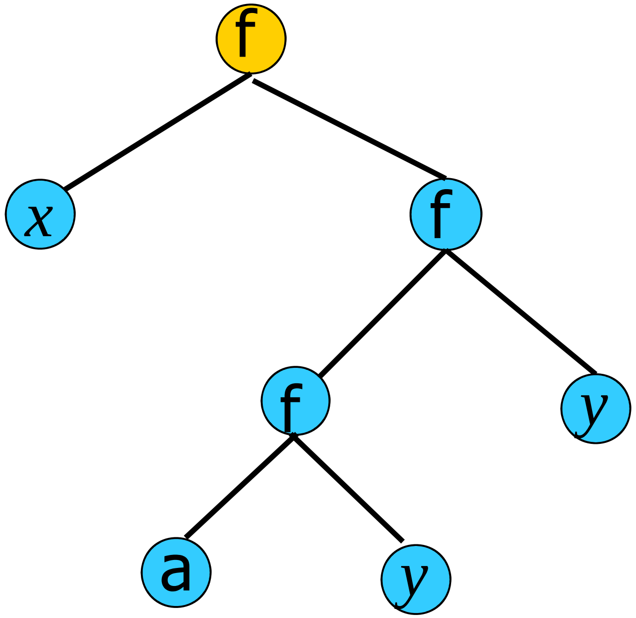
f(x_1,a), f(f(a, x_1), x_2), f(x_1, f(a, x_2)),...

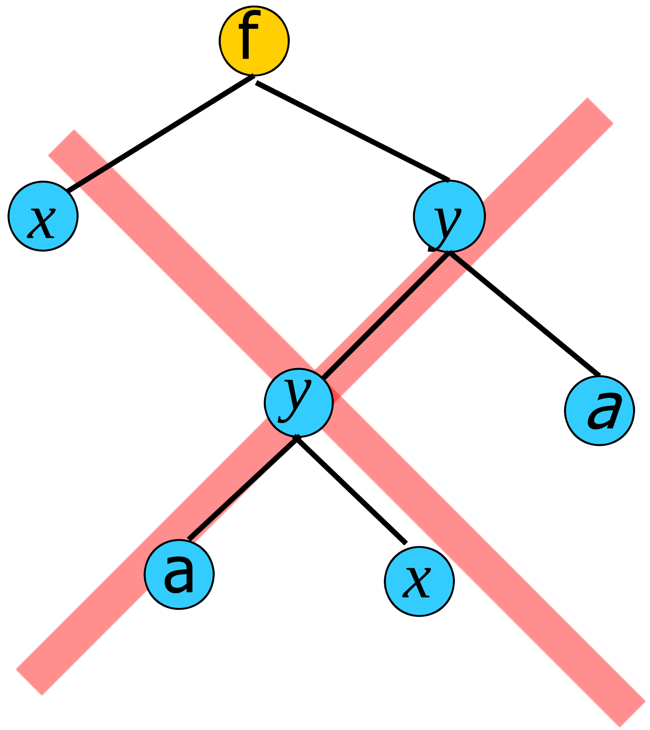
Tree Patterns (2)

- Since we let the rank of any variable be 0, it can be attached only to leaf nodes.

$\Sigma = \{a/0, f/2\}$, $X = \{x/0, y/0, \ldots\}$

$\pi = f(x, f(f(a, y), y))$





Defining tree languages with tree patterns

- A tree language defined with a tree pattern π is

$\{\sigma \mid \sigma = \pi\theta$ for some non-empty grounding substitution $\theta\}$

The language is denoted by L(π).

Examples

$\Sigma = \{a/0, f/2\}$, $X = \{x/0, y/0, \ldots\}$

$T(\Sigma) = \{$a, f(a,a), f(f(a,a),a), f(a,f(a,a)), f(f(a,a), f(a,a)),... $\}$

TL(f(x,a)) = {f(a,a), f(a,a), f(f(a,a),a), f(f(f(a,a),a),a),... }

TL(f(x,a)) = {f(a,a), f(a,a), f(f(a,a),a), f(f(f(a,a),a),a),... }

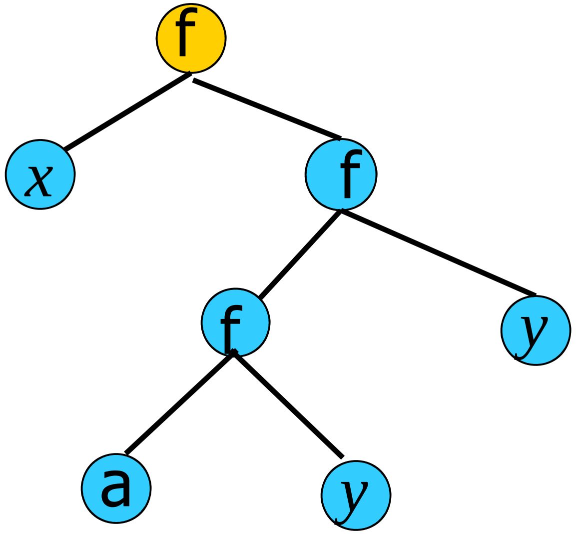
TL(f(x,f(a, x))) = {f(a,f(a, a)), f(f(a,a),f(a, f(a,a))),
f(f(f(a,a),a),f(a, f(f(a,a),a))),...

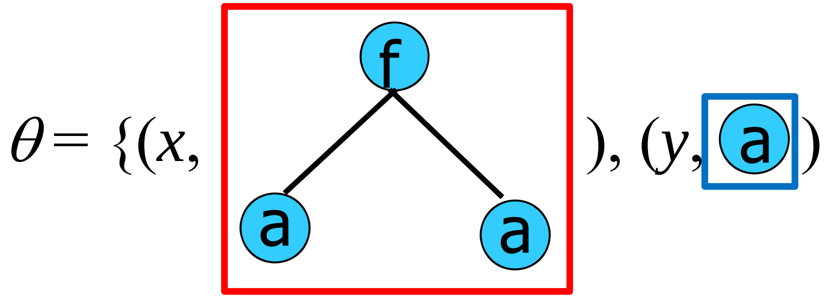
TL(f(x,f(a, y))) = {f(a,f(a, a)), f(a,f(a, f(a,a))), f(f(a,a),f(a,a)),
f(f(a,a),f(a, f(a,a))),
f(a,f(a, f(f(a,a),a)))),...}

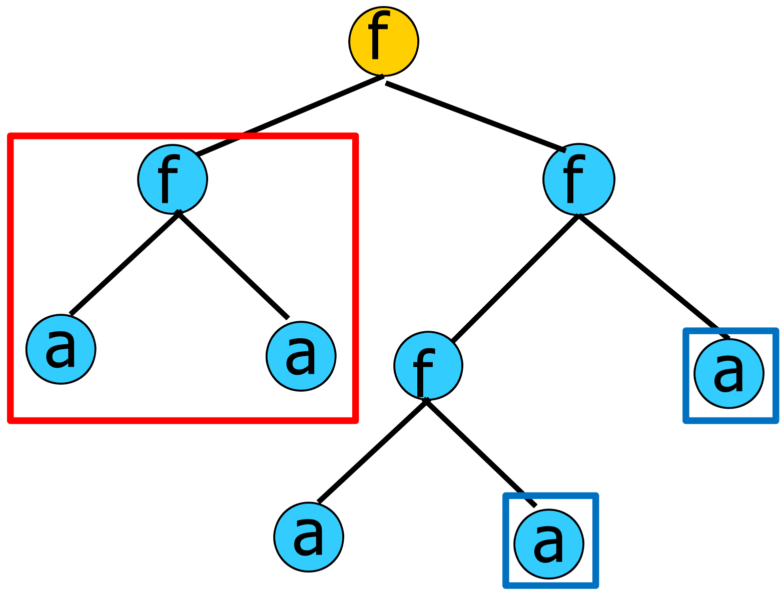
Substitution(2)

$\pi = \mathrm{f}(x,\mathrm{f}(\mathrm{f}(\mathrm{a}, y), y))$

$\theta = \{ (x, \mathrm{f}(\mathrm{a},\mathrm{a})), (y, \mathrm{a}) \}$







$\pi\,\theta = \mathrm{f}(\mathrm{f}(\mathrm{a},\mathrm{a}),\mathrm{f}(\mathrm{f}(\mathrm{a}, \mathrm{a}), \mathrm{a}))$

Substitution (1)

- A substitution is a set of pairs

$$\theta = \{\ (x_1, \tau_1), (x_2, \tau_2), \ldots, (x_n, \tau_n)\ \}$$

  where $x_1, x_2, \ldots, x_n$ are distinct variables and

  $\pi_1, \pi_2, \ldots, \pi_n$ are patterns.
- Applying a substitution θ to a tree pattern π is replacing every variable x_i in π with τ_i simultaneously. The result is denoted by $\pi\theta$.
- The rank of any variable is 0, but any pattern can be π_ι.

Examples

$\theta_1 = \{\ (x, \mathrm{f}(\mathrm{a},\mathrm{a})), (y, \mathrm{a})\ \}$

$\theta_2 = \{\ (x, \mathrm{f}(y,\mathrm{a})), (y, \mathrm{f}(\mathrm{a},y))\ \}$

$\mathrm{f}(x,x)\theta_1 = \mathrm{f}(\mathrm{f}(\mathrm{a},\mathrm{a}), \mathrm{f}(\mathrm{a},\mathrm{a}))$

$\mathrm{f}(x,\mathrm{f}(\mathrm{a}, y))\theta_1 = \mathrm{f}(\mathrm{f}(\mathrm{a},\mathrm{a}),\mathrm{f}(\mathrm{a}, \mathrm{a}))$

$\mathrm{f}(x,\mathrm{f}(\mathrm{a}, y))\theta_2 = \mathrm{f}(\mathrm{f}(y,\mathrm{a}),\mathrm{f}(\mathrm{a}, \mathrm{f}(\mathrm{a}, y)))$

$\mathrm{f}(x,x)\theta_2 = \mathrm{f}(\mathrm{f}(y,\mathrm{a}), \mathrm{f}(y,\mathrm{a}))$

Substitution (2)

- ~~A substitution θ={ $(x_1, \tau_1), (x_2, \tau_2), \ldots, (x_n, \tau_n)$ } is non-empty if all of $\tau_1, \tau_2, \ldots, \tau_n$ are in TL($\Sigma \cup X$).~~
- A substitution θ grounds a pattern π if $\pi\theta \in$TL(Σ). Such θ is called a grounding substitution for π.
- A substitution θ={ $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ } is variable renaming if $y_1, y_2, \ldots, y_n$ are distinct varaibles.
  - We regard two tree patterns equivalent when each one is obtained from the other by renaming variables.

Examples

Two tree patterns f(x,a) and f(y,a) are equivalent, and they are also equivalent to f(x_1,a).

Two patterns f(f(x,a),f(y,x)) and f(f(y,a),f(x,y)) are equivalent, and they are also equivalent to f(f(z,a),f(w,z)) and f(f(x_1,a),f(y_1, x_1))

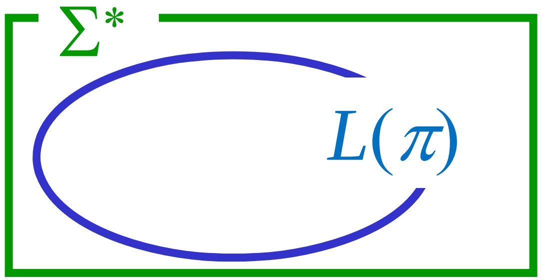
Learning tree pattern languages

Example

C = {f(a,f(f(a,a),a)),
f(f(a,a),f(f(a,a),a))),
f(f(a,a),f(f(f(a,a),a),f(a,a))),
f(f(f(a,a),a),f(f(a,a),a)))}

Positive Presentations

$e_1, e_2, e_3, \ldots$

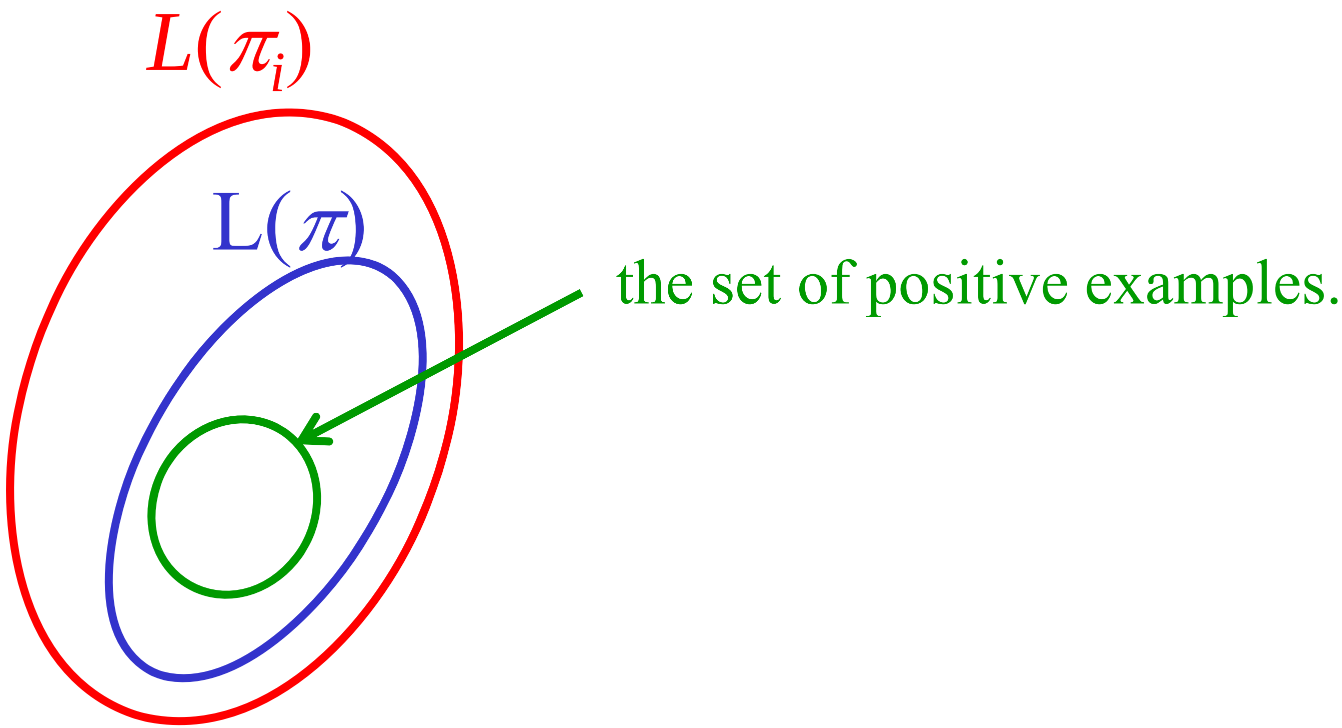


$\pi_1, \pi_2, \pi_3, \ldots$

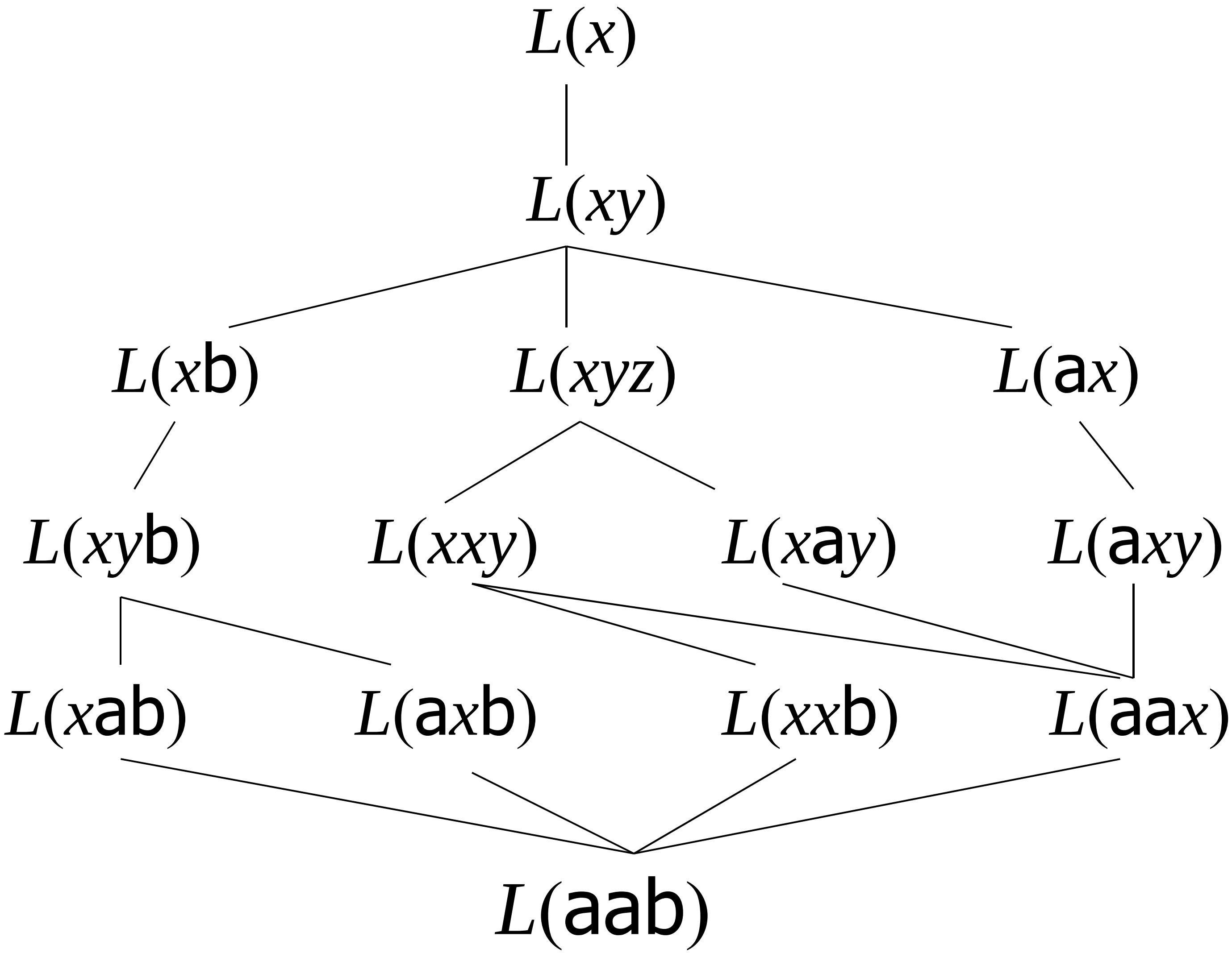
- A presentation of $L(\pi)$ is a infinite sequence consisting of positive and negative example.
- A presentation σ is positive if σ consists only of positive example $< s, +>$ and any positive example occurs at least once in σ.

Which patterns should be chosen?

- Intuitively, choose a minimal language which contains all of the positive examples at the moment.
  - That is, avoid over-generalization!

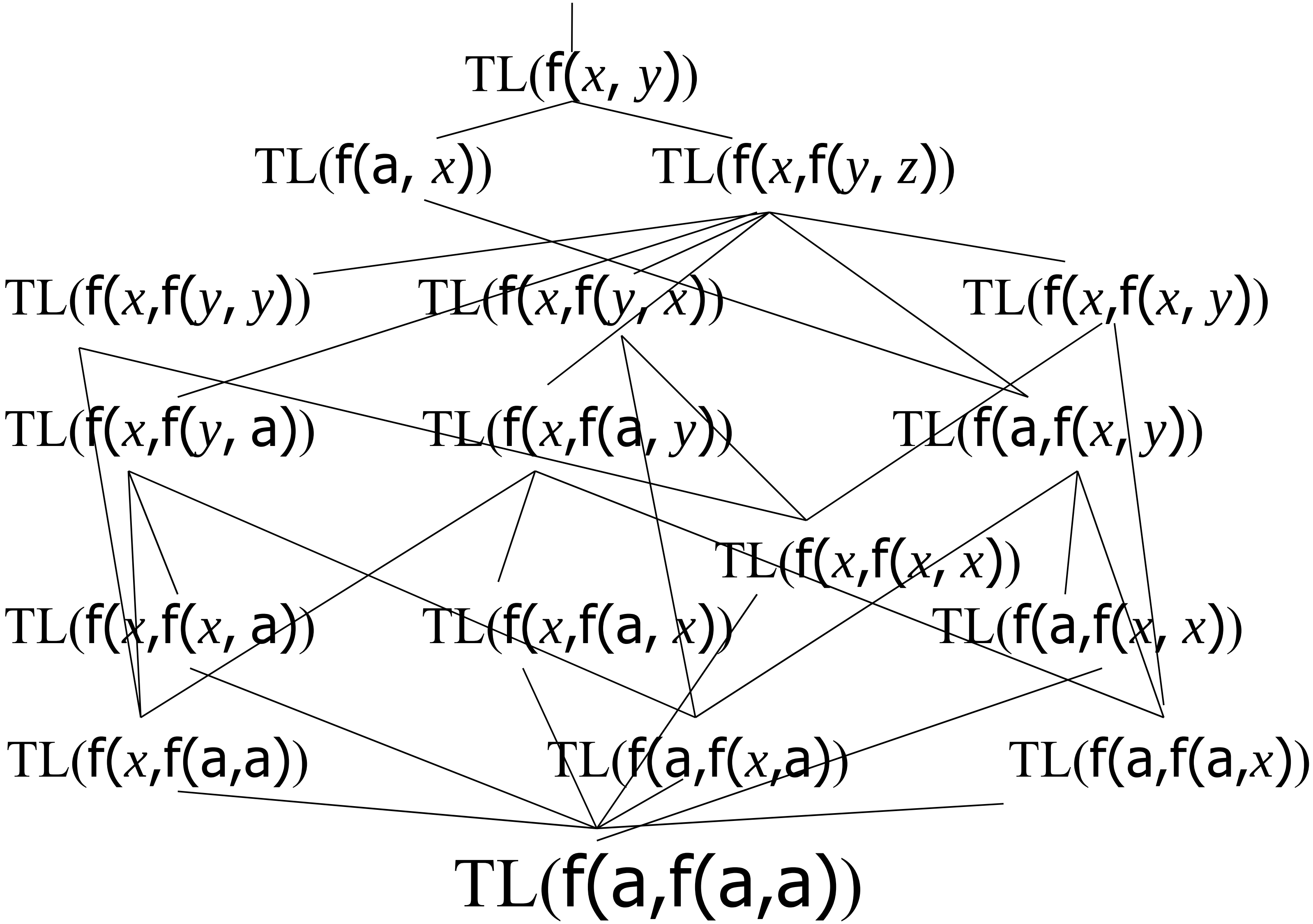


Hasse Diagram : String Case



Hasse Diagram : Tree Case

$\Sigma = \{$a/0, f/2$\}$, $X = \{x, y, \ldots\}$ TL (x)



Hasse Diagram : Tree Case

$\Sigma = \{$a/0, f/2$\}$, $X = \{x, y, \ldots\}$

TL (x)

TL(f(x, y))

TL(f(a, x)) TL(f(x,f(y, z))

TL(f(x,f(y, y)) TL(f(x,f(y, x)) TL(f(x,f(x, y))

TL(f(x,f(y, a)) TL(f(x,f(a, y)) TL(f(a,f(x, y))

TL(f(x,f(x, x))

TL(f(x,f(x, a)) TL(f(x,f(a, x)) TL(f(a,f(x, x))

TL(f(x,f(a,a)) TL(f(a,f(x,a)) TL(f(a,f(a,x))

TL(f(a,f(a,a))

C4: Finite thickness

- A class L(G) of languages has the finite thickness if for all $w \in \Sigma^*$ there are only a finite number of languages in L(G) which contain w.

Theorem [Angluin] A class L(G) of languages is identifiable in the limit from positive presentation if if L(G) of languages has the finite thickness.

Analysis of Tree Patterns

Example $\Sigma = \{$a/0, f/2$\}$, $X = \{x$/0, y/0,…$\}$

TL(f(x,f(f(y,a), y)))

={f(a,f(f(a,a),a)),

f(f(a,a),f(f(a,a),a)),

f(f(a,a),f(f(f(a,a),a),f(a,a))),

f(f(f(a,a),a),f(f(a,a),a))), …

f(f(f(f(a,a),a),a),f(f(a,a),a))), …}

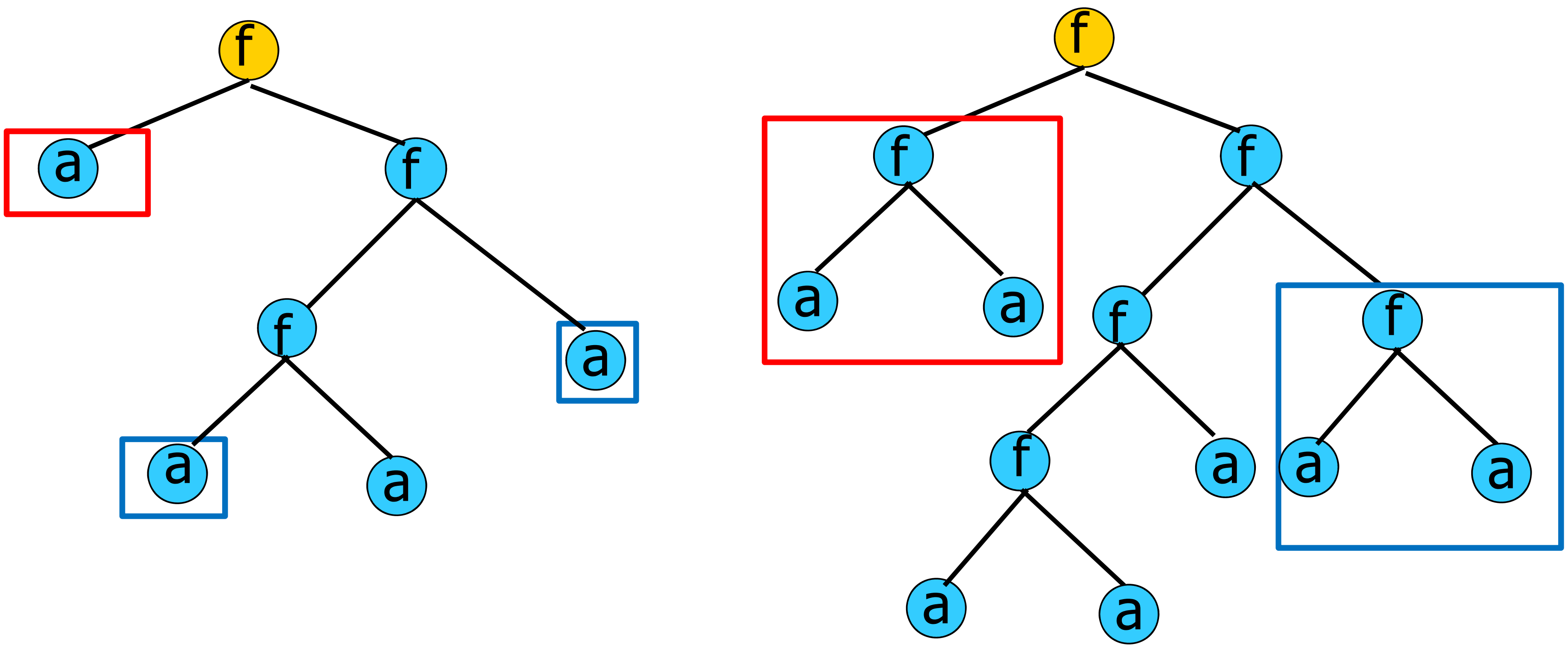
Anti-Unification of Trees

f(x,f(f(y,a), z))

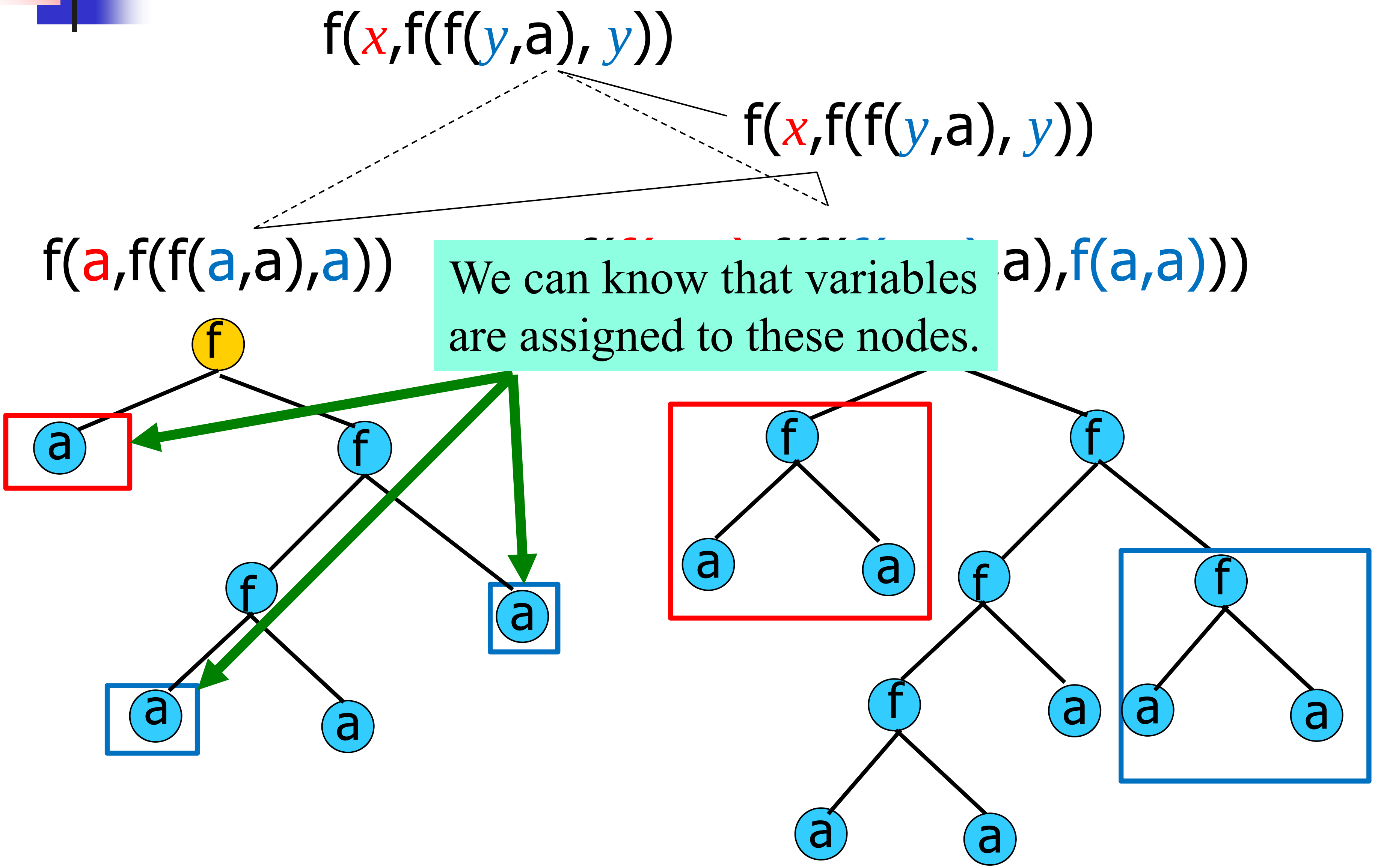
f(x,f(f(y,a), y))

f(a,f(f(a,a),a))

f(f(a,a),f(f(f(a,a),a),f(a,a)))



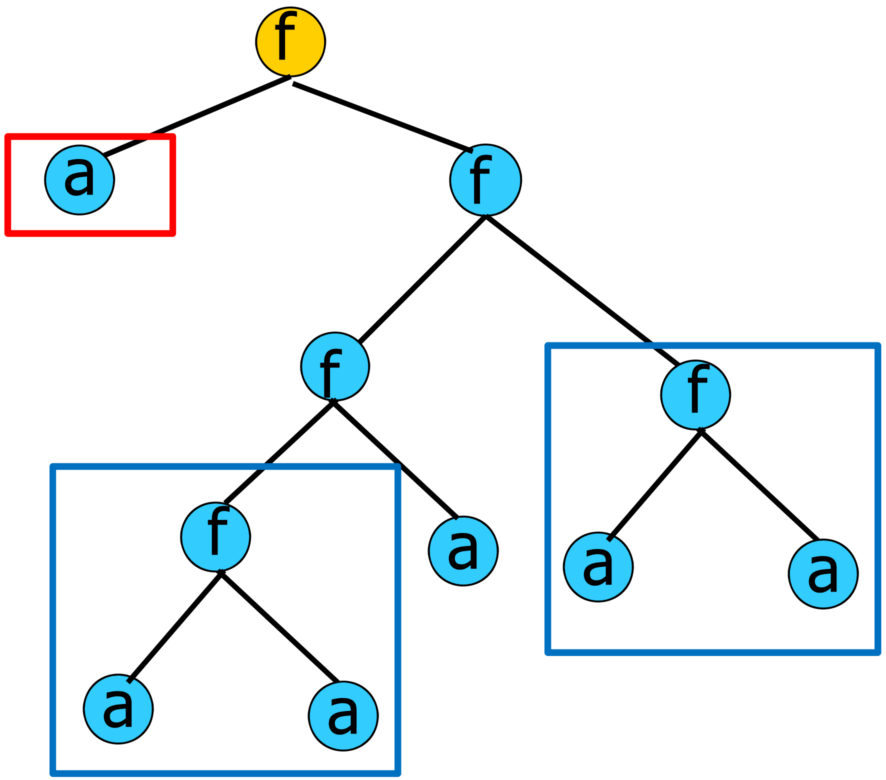
Anti-Unification of Trees

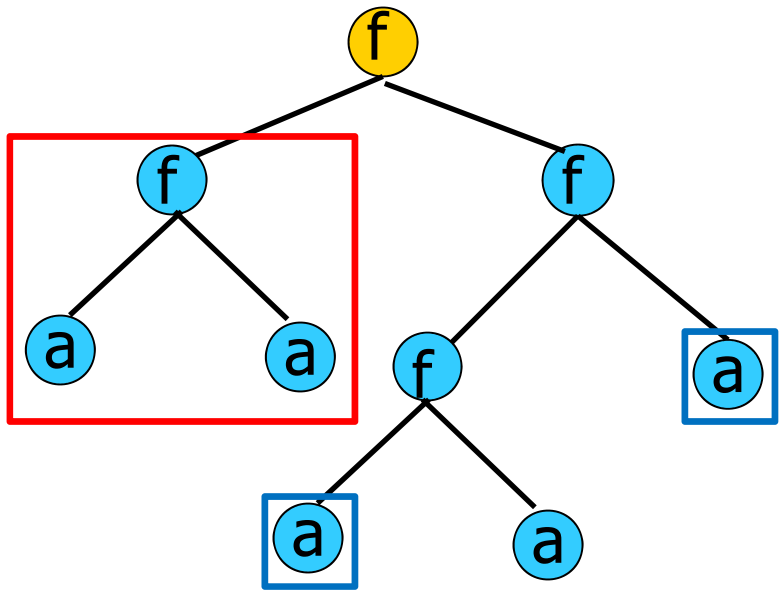


Anti-Unification of Trees

$f(x, f(f(y, a), y))$

$f(a, f(f(f(a,a), a), f(a,a)))$ $f(f(a,a), f(f(a,a), a))$



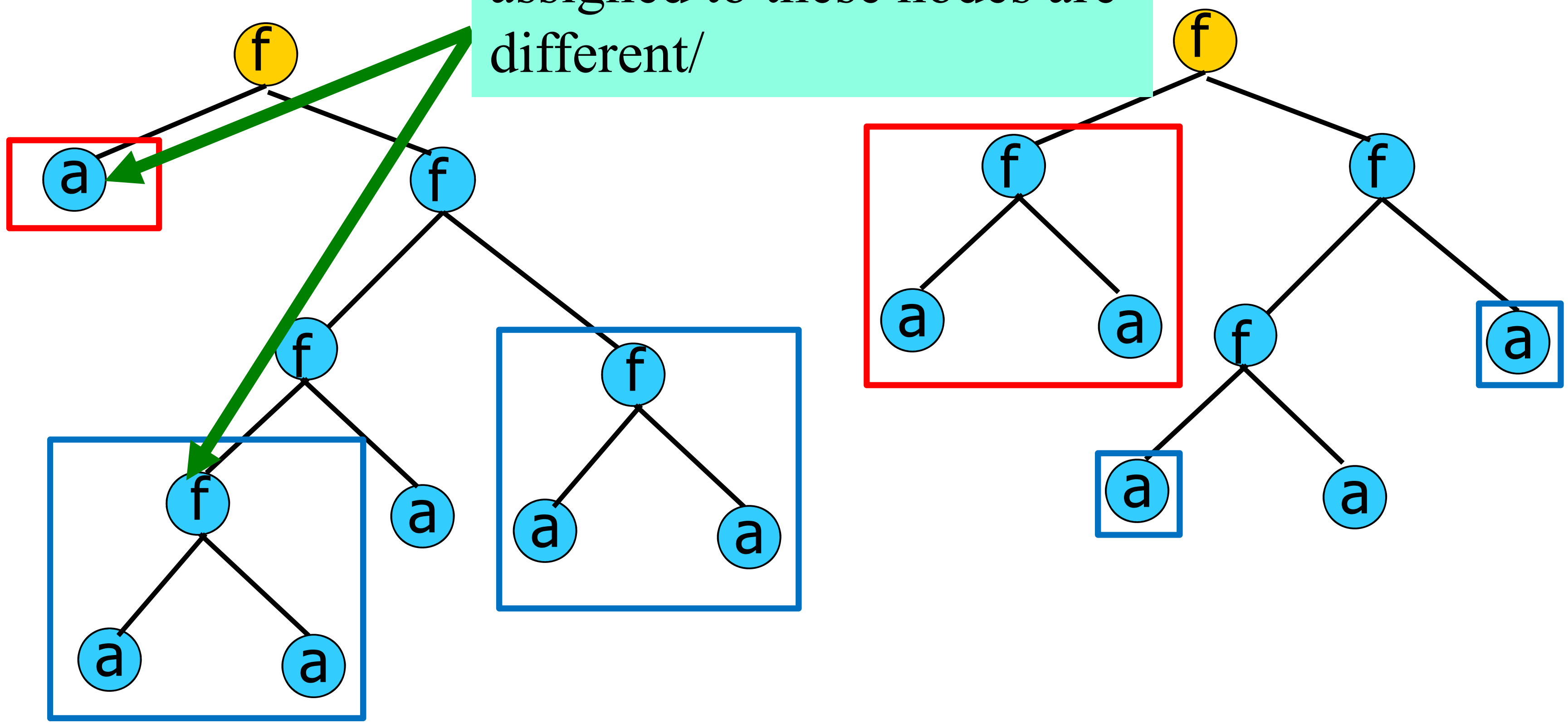


Anti-Unification of Trees

$f(x, f(f(y, a), y))$

$f(a, f(f(f(a,a),a), f$ … $(f(a,a),a))$

We can know that variables assigned to these nodes are different/



Characteristic Set of $L(\pi)$

- Let π be a pattern which contains variables $x_1, x_2, \ldots, x_n$. Consider the following substitutions:

  $\sigma_1 = \{(x_1, t), (x_2, \mathsf{a}), \ldots, (x_n, \mathsf{a})\}$,

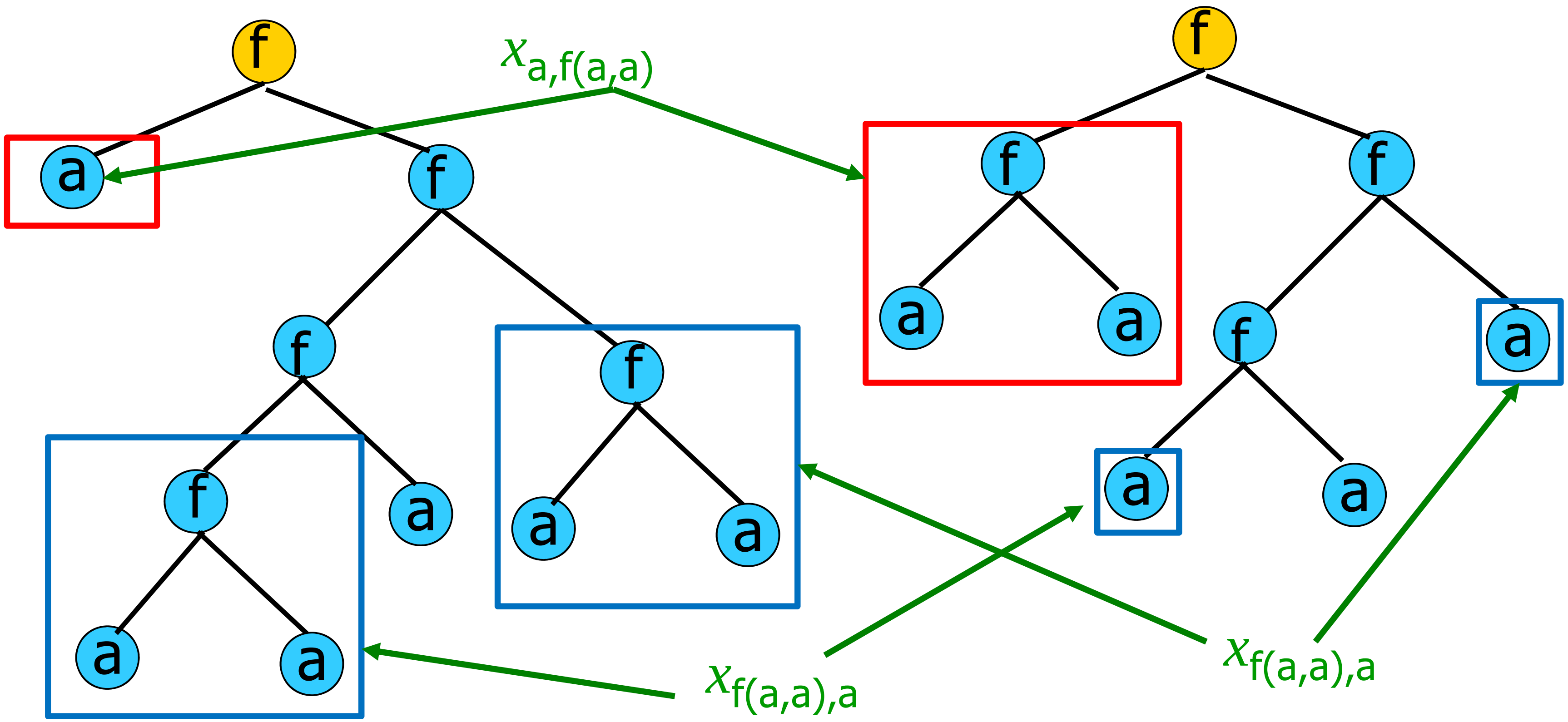
  $\ldots$

  $\sigma_n = \{(x_1, \mathsf{a}), (x_2, \mathsf{a}), \ldots, (x_n, t)\}$

  where t is a tree different from a, e.g. f(a,a) .
- The set $\{\pi\sigma_1, \ldots, \pi\sigma_n\}$ is a characteristic set of $L(\pi)$.

Anti-Unification Algorithm

1. Compare two trees from their roots.
2. If different labels are attached to nodes on a same position, replace the labels with a variable with an index which indicates the difference.



Anti-Unification Algorithm

$<f(a,f(f(f(a,a),a),f(a,a))),\ \ f(f(a,a),f(f(a,a),a))>$

$\Rightarrow <f(a,f(f(f(a,a),a),f(a,a))),\ f(f(a,a),f(f(a,a),a))>$

$\Rightarrow <f(x_{a,f(a,a)},f(f(f(a,a),a),f(a,a))),\ f(x_{a,f(a,a)},f(f(f(a,a),a),a))>$

$\Rightarrow <f(x_{a,f(a,a)},f(f(f(a,a),a),f(a,a))),\ f(x_{a,f(a,a)},f(f(a,a),a))>$

$\Rightarrow <f(x_{a,f(a,a)},f(f(f(a,a),a),f(a,a))),\ f(x_{a,f(a,a)},f(f(a,a),a))>$

$\Rightarrow <f(x_{a,f(a,a)},f(f(x_{f(a,a),a},a),f(a,a))),\ f(x_{a,f(a,a)},f(f(x_{f(a,a),a},a),a))>$

$\Rightarrow <f(x_{a,f(a,a)},f(f(x_{f(a,a),a},a),f(a,a))),\ f(x_{a,f(a,a)},f(f(x_{f(a,a),a},a),a))>$

$\Rightarrow <f(x_{a,f(a,a)},f(f(\ x_{f(a,a),a},a),\ x_{f(a,a),a})),$

$f(x_{a,f(a,a)},f(f(x_{f(a,a),a},a),\ x_{f(a,a),a}))>$

$\Downarrow$

$f(x,f(f(y,a),\ y))>$

Identification of tree patterns

Theorem By applying the anti-unification to all given examples at each moment, the learning machine EX-identifies the class of all tree pattern languages in the limit from positive presentations.

Origin of Learning Tree Patterns

G. Plotkin: Automatic Methods of Inductive Inference, 1971.

AUTOMATIC METHODS OF INDUCTIVE INFERENCE

G. PLOTKIN

PhD Thesis

Edinburgh University, 1971

G. Plotkin

The Royal Society:

Plotkin has contributed to Artificial Intelligence, Logic, Linguistics and especially to Computer Science. In AI he worked on hypothesis-formation and universal unification; in Logic, on frameworks for arbitrary logics; in Linguistics, on formalising Situation Theory.

His main general contribution has been to establish a semantic framework for Computer Science, especially programming languages. Particular significant results are in the lambda-calculus (elementary models, definability, call-by-value), non-determinism (powerdomain theory), semantic formalisms (structured operational semantics, metalanguages), and categories of semantic domains (coherent, pro-finite, concrete). Further contributions concern the semantic paradigm of full abstraction, concurrency theory (event structures), programming logic and type theory.

Example[Plotkin71]

ID	Class	Size	Color	Animal
a	dangerous	small	black	bear
b	dangerous	medium	black	bear
c	dangerous	large	black	dog
d	safe	small	black	cat
e	safe	medium	black	horse
f	dangerous	large	black	horse
g	dangerous	large	brown	horse

$$\forall x.\ \mathrm{black}(x) \rightarrow \mathrm{dangerous}(x)$$

$$\forall x.\ \mathrm{large}(x) \wedge \mathrm{horse}(x) \rightarrow \mathrm{dangerous}(x)$$