Calculs inductifs de fonctions sur des graphes décomposés de largeur de clique bornée

Frédérique Carrère (carrere@labri.fr)
LaBRI

April 1, 2007

Preliminaries

- We consider simple graphs, undirected or directed, defined by three-uple of the form (V, E, γ), where V denotes the set of vertices, E the set of edges and γ is a function from V into a finite alphabet.
- We are interested in tuples of inductive functions of the form $D_p(G, x_1, \ldots, x_p)$ where G is a graph and $x_1, \ldots, x_p$ are vertices of the graph
(for example, the distance between two vertices of G).

Problem

- put on each vertex some information depending on G and on itself such that one can compute $D_p(G, x_1, \ldots, x_p)$ just from the information put on $x_1, \ldots, x_p$.
- use only a distributed data structure:
  no centralized data structure
- Minimize the quantity of information on each vertex:
  (for example with $O(nlog(n))$ bits, one could put on each vertex the distance to all others)

Definitions

Definition

Let G be a graph.

An *f-labelling scheme* for G is a mapping $lab : V \longrightarrow \{0,1\}^*$ such that for every vertex x of G, the length of the word $lab(x)$ is at most $f(|V|)$.
($lab(x)$ is the data attached to x).

Definition

A numerical function g, defined on the vertices of G, is computable from the labeling scheme lab of G if there exists an algorithm to compute $g(x_1, \ldots, x_p)$ from the words $\{lab(x_1), \ldots, lab(x_p)$.

Clique-width

graphs families	maximal label length	
arbitrary graphs	$O(n)$	Gavoille-Peleg-Pérennès-Raz'04
bounded tree-width graphs	$O(log^2(n))$	Gavoille-Peleg-Pérennès-Raz'04
bounded clique-width graphs	$O(log^2(n))$	Courcelle-Vanicat'03
interval graphs	$O(log(n))$	Gavoille-Paul'03

Monadic second orde logic

Theorem (Courcelle-Vanicat'03)

- G a graph of clique width at most k,
- f a monadic second-order optimization function (like distance),
  one can associate with each vertex u of G
  a piece of information of size $O(log^2(n))$,
  such that one can compute $f(u, v)$ in time $O(log^2(n))$.

  The preprocessing time is $O(nlog^2(n))$.

MSO Logic

- this result uses very powerfull logical tools (monadic second order logic, tree automatas)
- these tools are not easy to implement
- these tools induce constants of exponential size

Clique-width at most k

Graphs of clique width at most k

k integer, $k \geq 2$

- $\gamma(V)$ is the set $\{1, \ldots, k\}$
  $\longrightarrow$ partition of V in $S_1, \ldots, S_k$
- three graphs operations:

  $\oplus \longrightarrow$ disjoint sum,

  $add_{i,j} \longrightarrow$ add all edges $S_i \times S_j$

  $ren_{i \to j} \longrightarrow$ relabel the vertices of S_i with j.

Terms

The graphs of bounded clique width are defined by terms (or trees) in $T(F, C)$, where:

- C is a set of k constants
- F is a set of binary graph operations of the form:
  $G = \otimes_{L,R}(G_1, G_2)$, with $L, R \subset \{1, \ldots, k\} \times \{1, \ldots, k\}$

  $\longrightarrow$ add the edges $S_i \times S_j$, for all (i, j) in L,
  $\longrightarrow$ relabel S_i with j, for all (i, j) in R,

Contexts

Contexts are terms of $T(F, C \cup \{u\})$ with a unique occurrence of u.

Substitution in a context

- $c \bullet t = c[t] \longrightarrow$ the result is a tree
- $c \circ c' = c[c'] \longrightarrow$ the result is a context

Note:
if $f \in F$, t is a tree, c, c' are contexts,
- $f(c, t)$ and $f(t, c)$ are trees
- $f(c, c')$ is not a context (two occurrences of u)

Systems of inductive fonctions

tuples of **numerical** functions $\{D_0(t), D_1(t,x), D_2(t,x,y)\}$ defined on **terms** $t \in T(F,C)$ [resp. on **contexts**] and nodes x, y of t $[x, y \neq u]$, such that:

- if $t = f(t_1, t_2)$ and $x \in t_1$:

$$D_2(\, f(t_1,t_2), x, y) =$$
$$\begin{cases} \phi'_f(\, D_2(t_1,x,y),\ D_1(t_1,x),\ D_1(t_1,y),\ D_0(t_1),\ D_0(t_2)), & \text{if } y \in t_1 \\ \phi_f(\, D_1(t_1,x),\ D_1(t_2,y),\ D_0(t_1),\ D_0(t_2)), & \text{if } y \in t_2 \end{cases}$$

$$D_1(\, f(t_1,t_2), x) = \psi_f(\, D_1(t_1,x),\ D_0(t_1),\ D_0(t_2)),$$

$$D_0(\, f(t_1,t_2)) = \xi_f(\, D_0(t_1),\ D_0(t_2)).$$

Systems of inductive fonctions

tuples of **numerical** functions $\{D_0(t), D_1(t,x), D_2(t,x,y)\}$ defined on **terms** $t \in T(F,C)$ [resp. on **contexts**] and nodes x, y of t $[x, y \neq u]$, such that:

- if $t = c_1 \bullet t_2$ and $x \in c_1$:

$$D_2(\, c_1 \bullet t_2, x, y) =$$
$$\begin{cases} \phi'_\bullet(\, D_2(c_1,x,y),\ D_1(c_1,x),\ D_1(c_1,y),\ D_0(c_1),\ D_0(t_2)), & \text{if } y \in c_1\ , \\ \phi_\bullet(\, D_1(c_1,x), D_1(t_2,y),\ D_0(c_1),\ D_0(t_2)), & \text{if } y \in t_2 \end{cases}$$

$$D_1(\, c_1 \bullet t_2, x) = \psi_\bullet(\, D_1(c_1,x),\ D_0(c_1),\ D_0(t_2)),$$

$$D_0(\, c_1 \bullet t_2) = \xi_\bullet(\, D_0(c_1),\ D_0(t_2)).$$

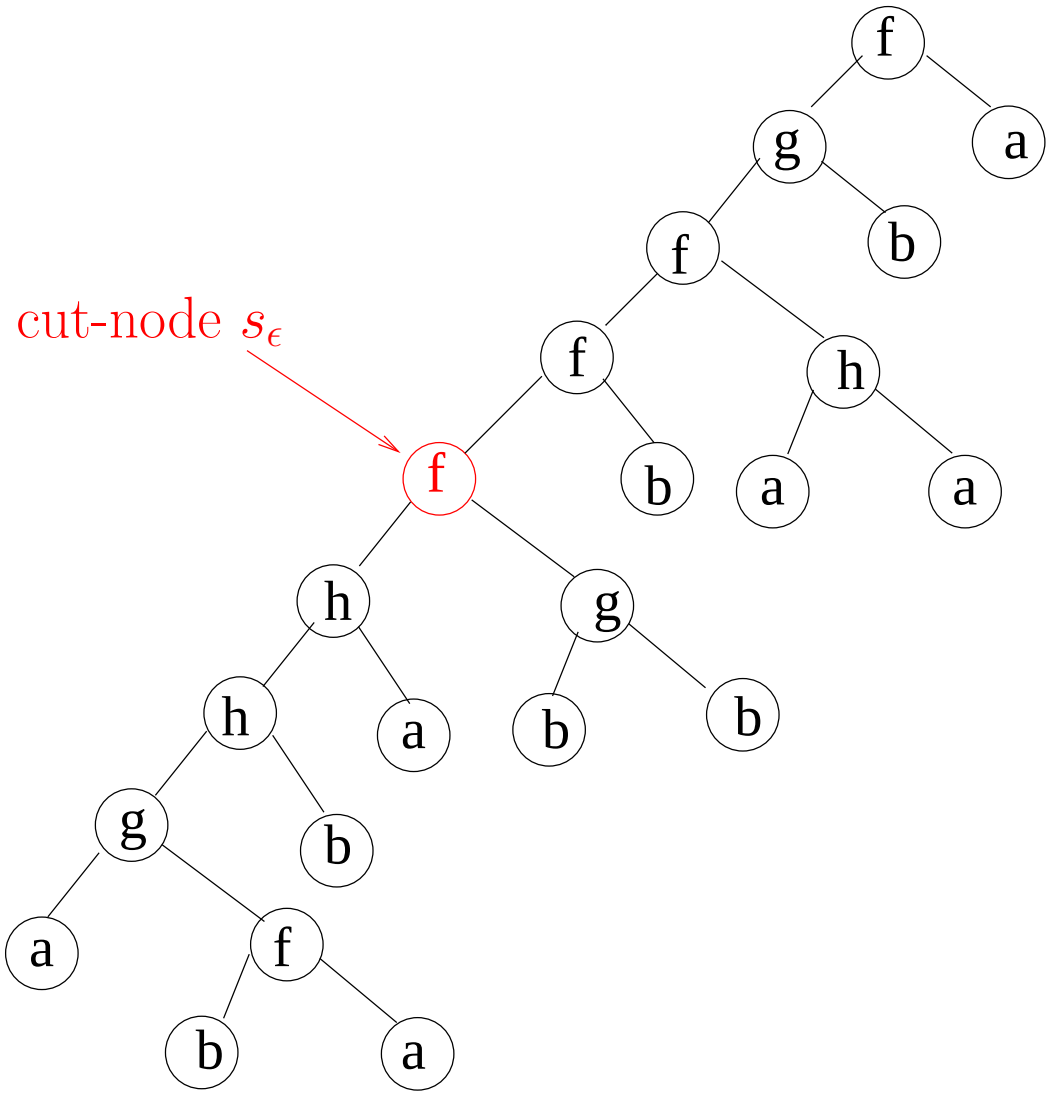
Cut-nodes

Proposition (Courcelle-Vanicat03)

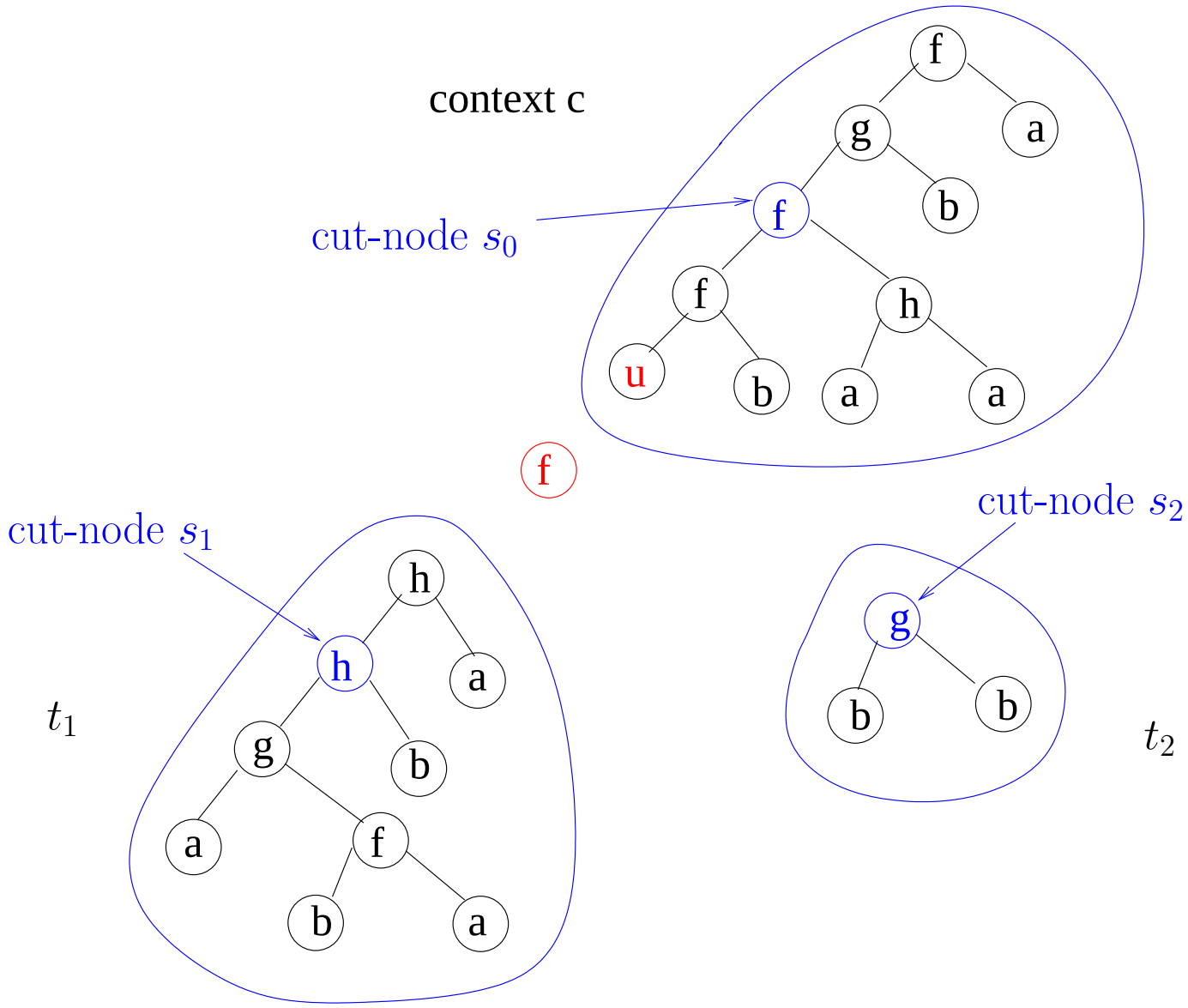
1. Let $t \in T(F, C)$, of size $|t| = n = 2p + 1$, $p \geq 1$.
   there exists a *cut-node* with label f such that:
   $t = c_0[f(t_1, t_2)]$
   with $|c_0| \leq p$ and $|t_i| \leq p + 1$ for $i = 1, 2$.

2. Let c a context of size $|c| = n = 2p + 1$, $p \geq 1$.
   there exists a *cut-node* with label f such that:
   $c = c_1[f(c_2, t)]$ or $c = c_1[f(t, c_2)]$
   with $|c_1| \leq p$, $|c_2| \leq p + 1$ and $|t| \leq 2p - 1$.

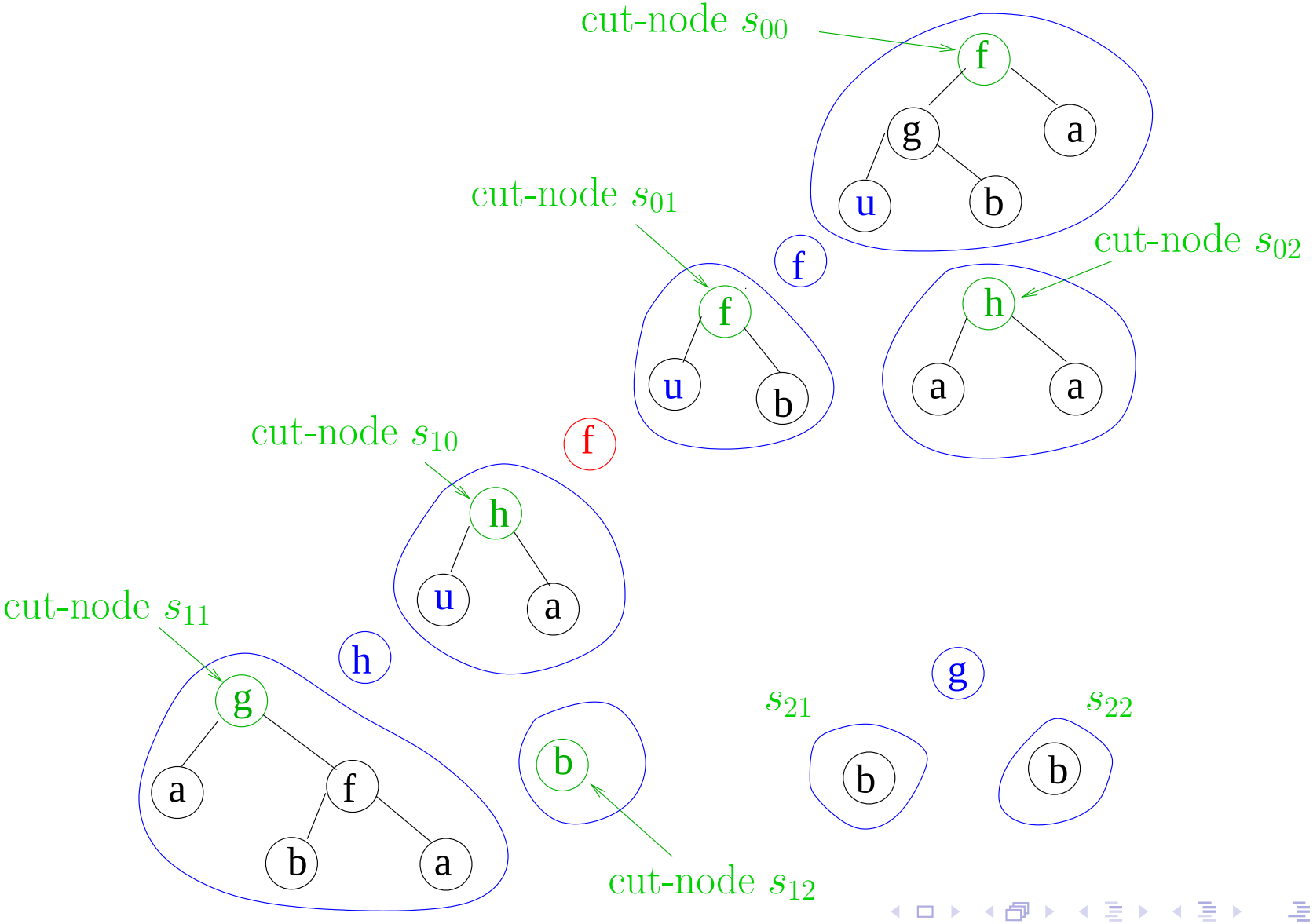
cut-nodes



cut-nodes


context c
cut-node s_0
f
g
a
b
f
h
u
b
a
a
f
cut-node s_1
h
h
a
g
b
a
f
b
a
t_1
cut-node s_2
g
b
b
t_2

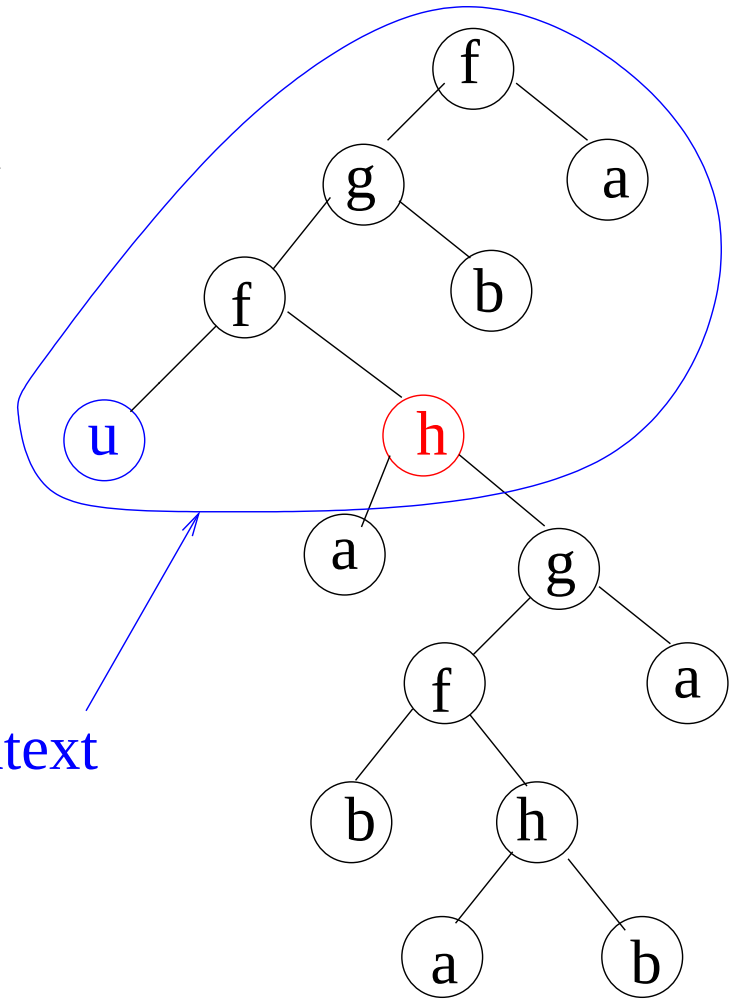
cut-nodes


cut-node s_{00}
f
g
a
u
b
cut-node s_{01}
f
f
u
b
cut-node s_{02}
h
a
a
cut-node s_{10}
f
h
u
a
cut-node s_{11}
h
g
a
f
b
a
b
cut-node s_{12}
g
s_{21}
b
s_{22}
b

Tree partition

- Fact 1:

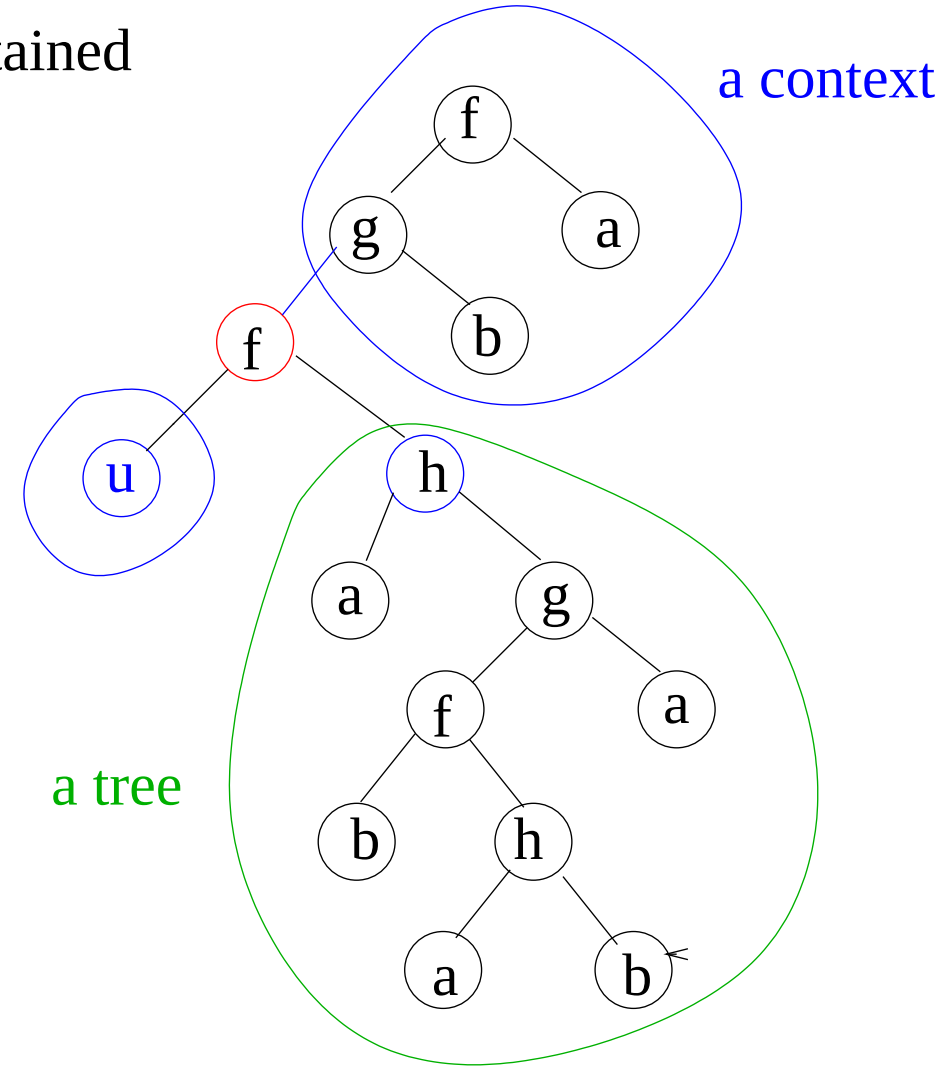
All the blocks obtained
by partitioning t
are trees
or contexts



Tree partition

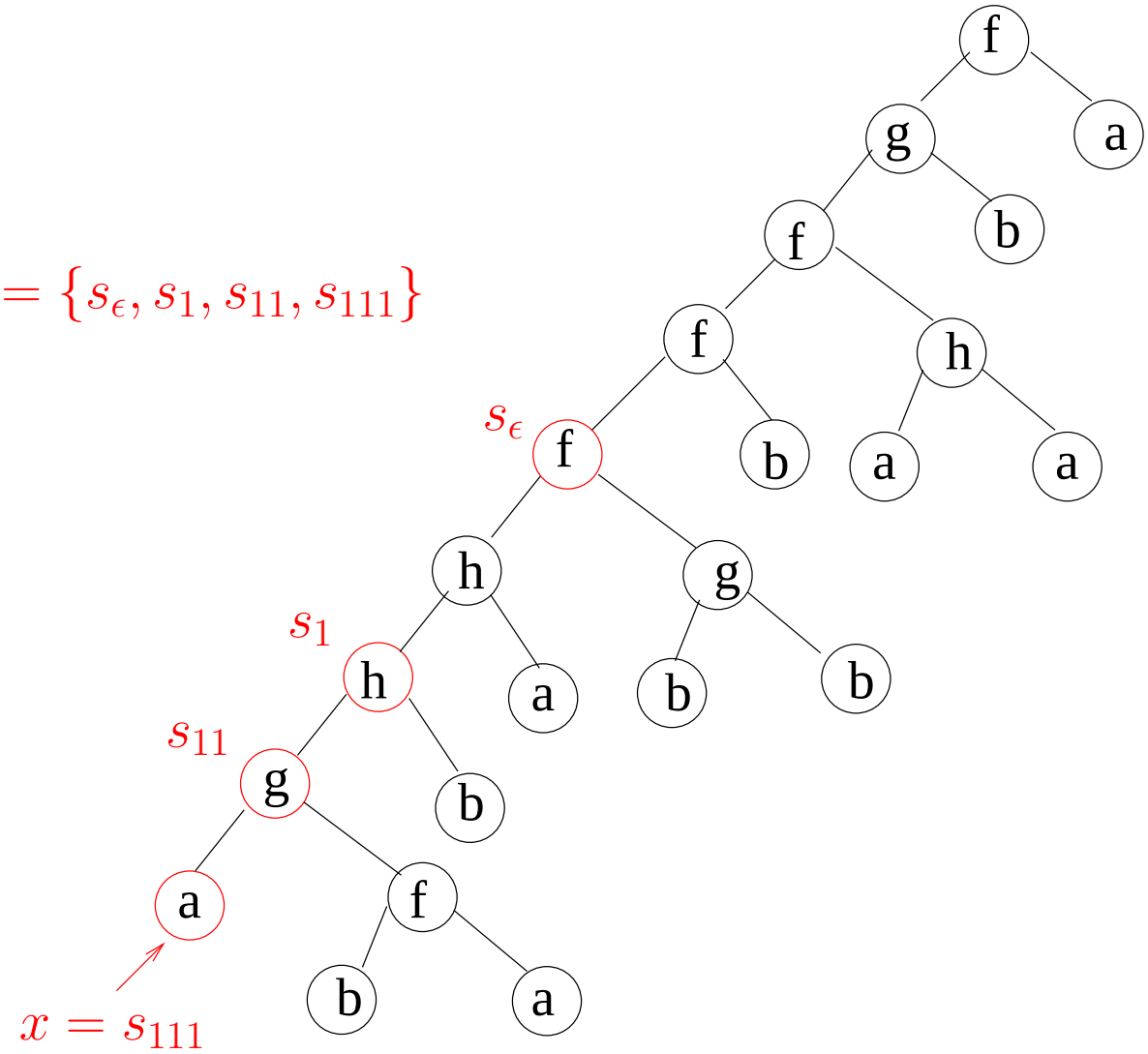
- Fact 1:

All the blocks obtained
by partitioning t
are trees
or contexts



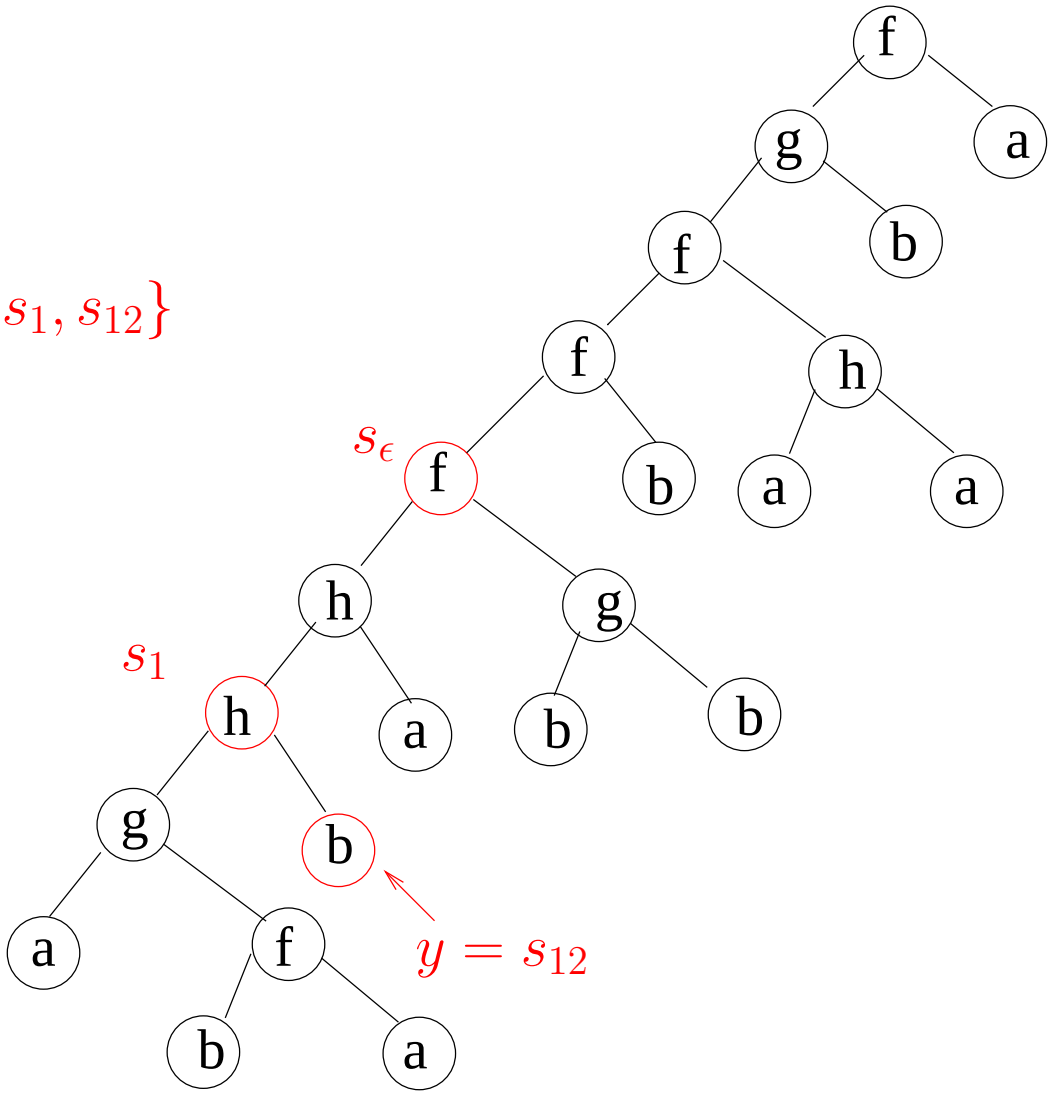
Relevant cut-nodes for x

$sep(x) = \{s_\epsilon, s_1, s_{11}, s_{111}\}$



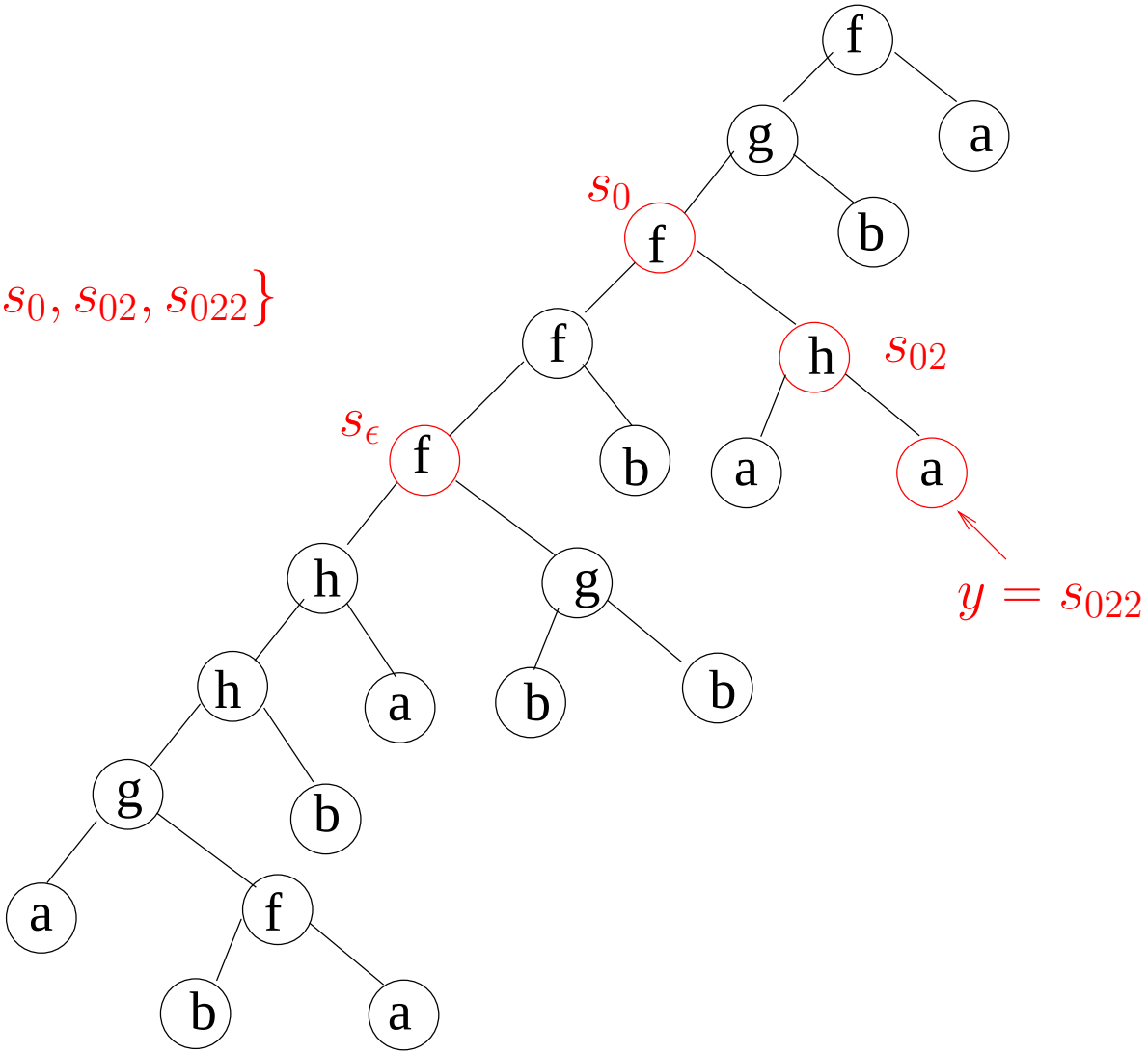
Relevant cut-nodes for y

$sep(y) = \{s_\epsilon, s_1, s_{12}\}$



Relevant cut-nodes for z

$sep(y) = \{s_\epsilon, s_0, s_{02}, s_{022}\}$



Partition of the tree

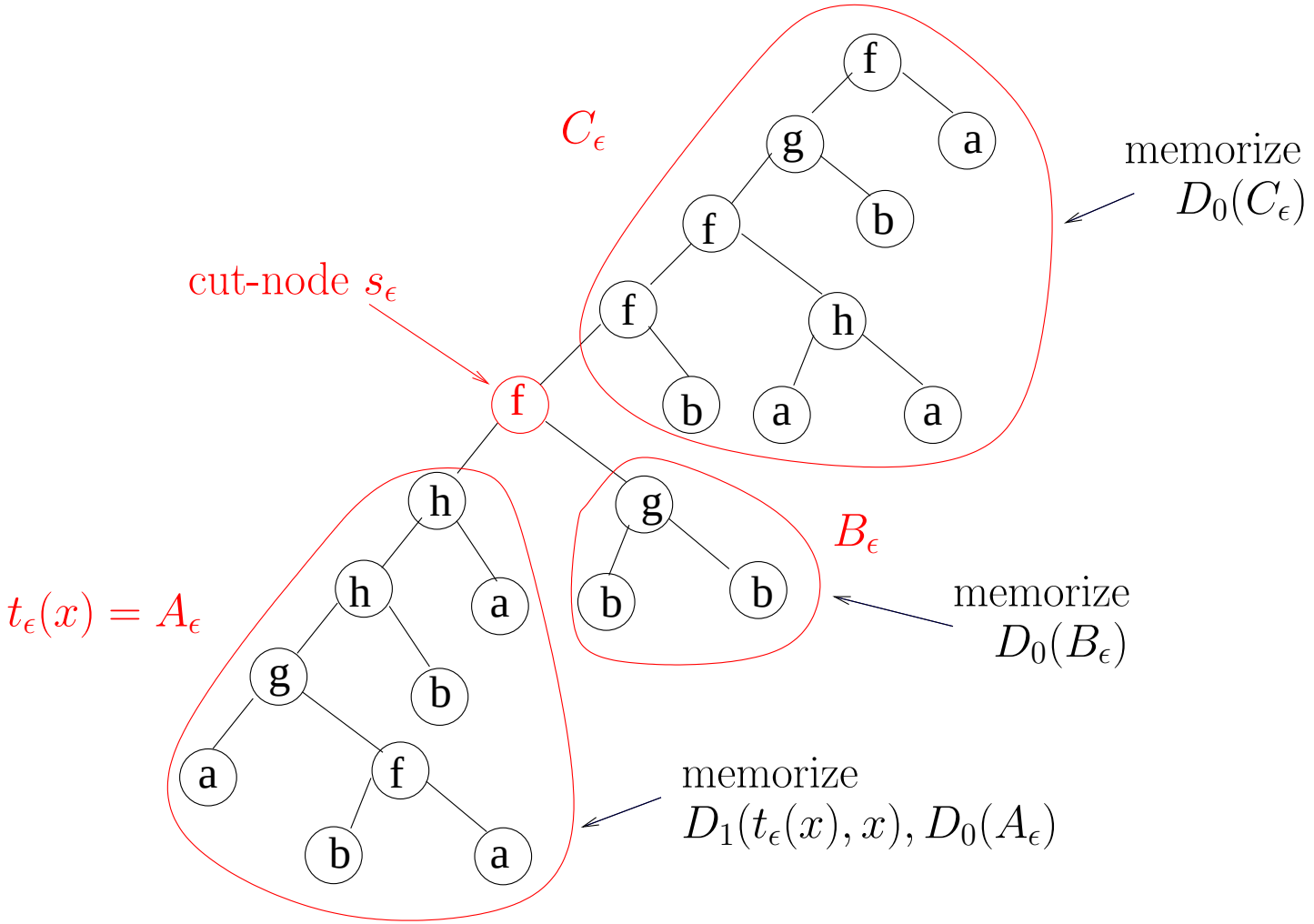
Fact 1

All the blocks obtained by partitionning t are trees or contexts.

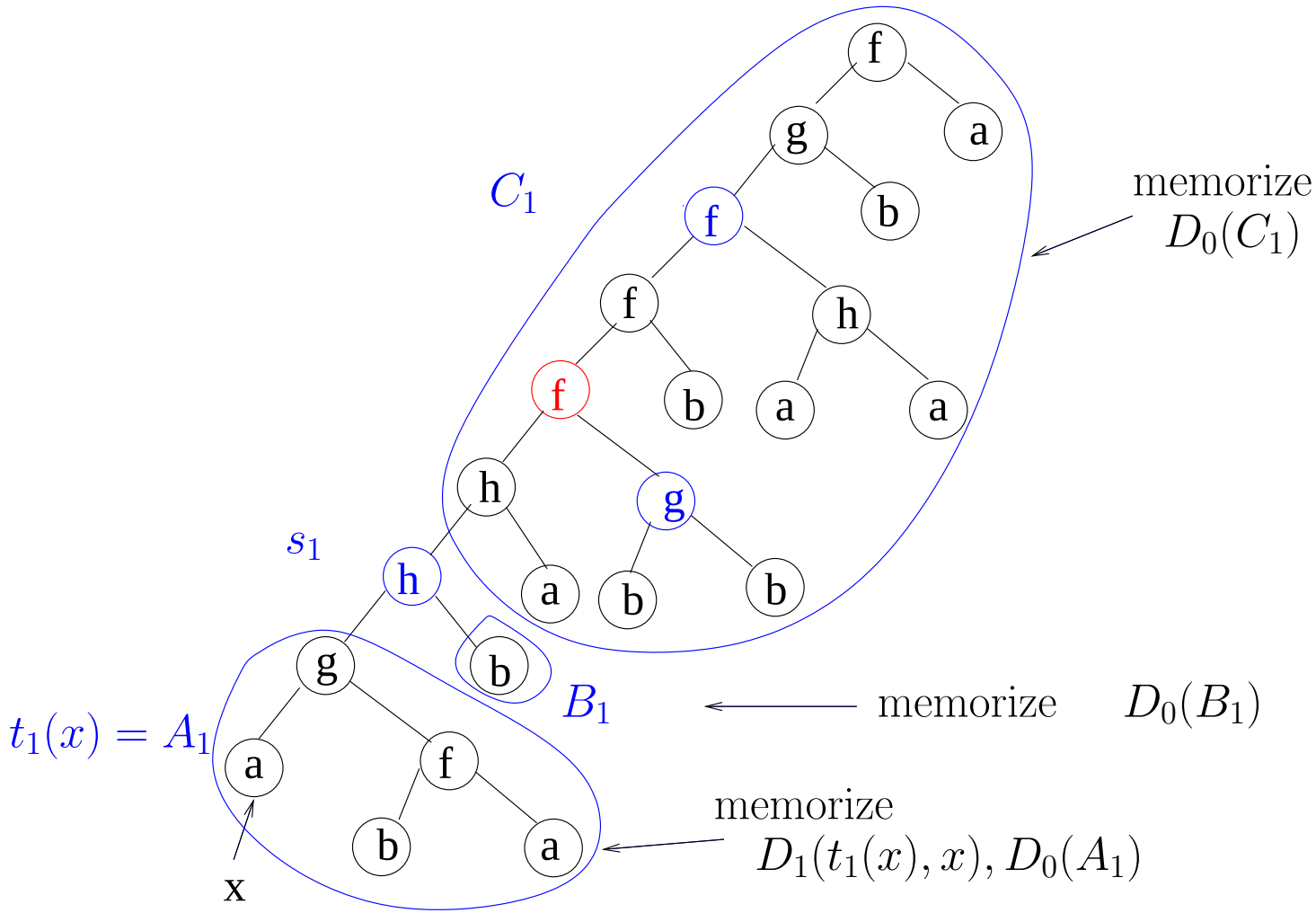
Fact 2

The length of the sequence $sep(x)$ is at most $3log(n)$.

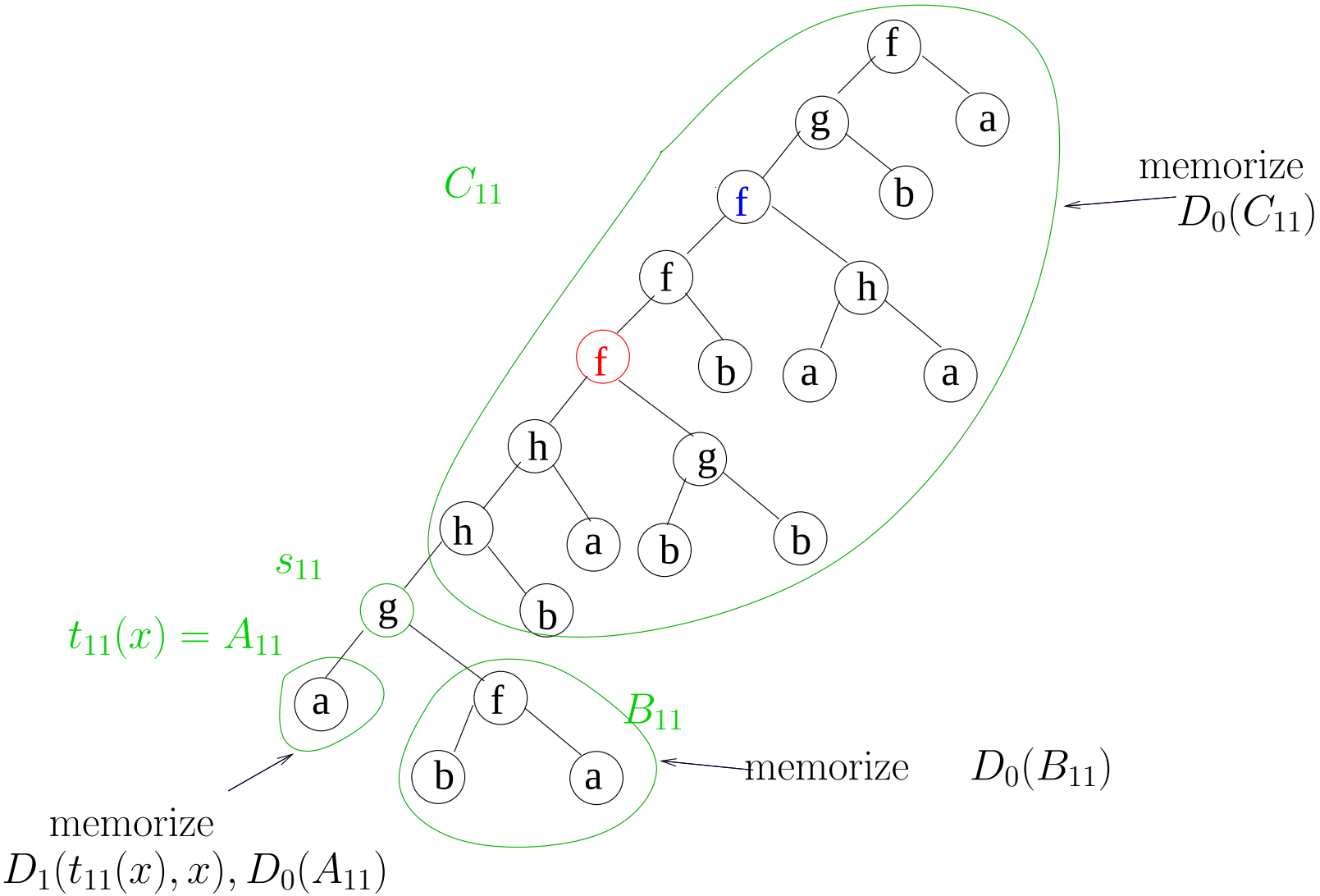
The information $lab(x)$



The information $lab(x)$



The information $lab(x)$



The information $lab(x)$

The label of x

For all s_w in $sep(x)$, we put in $lab(x)$:

$\{\ldots, s_w(x), D_1(T_w(x), x), D_0(A_w), D_0(B_w), D_0(C_w), \ldots\}$,

where $T_w(x)$ is A_w (resp. B_w, C_w), if x belongs to A_w (resp. B_w, C_w).

Example:

$$\begin{aligned} lab(x) = \{&s_\epsilon(x),\ D_1(A_\epsilon, x), D_0(A_\epsilon), D_0(B_\epsilon), D_0(C_\epsilon),\\ &s_1(x),\ D_1(A_1, x), D_0(A_1), D_0(B_1), D_0(C_1),\\ &s_{11}(x),\ D_1(A_{11}, x), D_0(A_{11}), D_0(B_{11}), D_0(C_{11})\} \end{aligned}$$

Partition of the tree

Fact 1

All the blocks obtained by partitionning t are trees or contexts.

Fact 2

The length of the sequence $sep(x)$ is at most $3log(n)$.

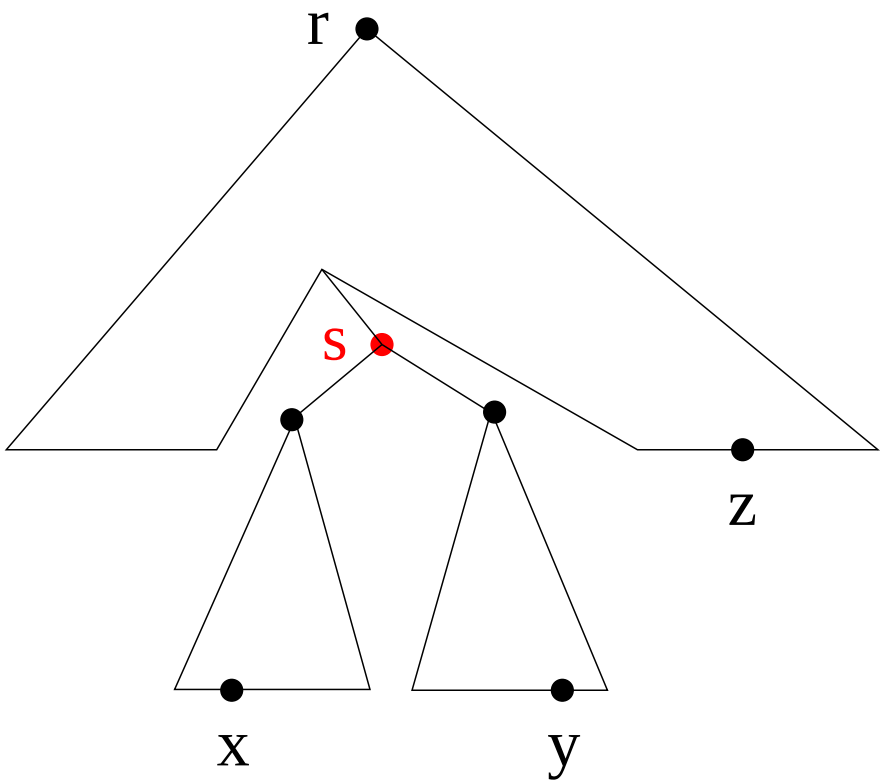
Fact 3

The set of values
$\{D_1(T_w(x)); x \text{ leaf of } t, w \text{ such that } s_w \in sep(x)\}$
can be computed in time $O(h(t) * n)$.

Separator of two vertices

Definition

The vertex z of t is a *separator* of x and y in t iff $t = c[f(t_1, t_2)]$ with x and y not in the same block c, t_1 ou t_2.



Partition of the tree

Fact 1

All the blocks obtained by partitionning t are trees or contexts.

Fact 2

The length of the sequence $sep(x)$ is at most $3log(n)$.

Fact 3

The set of values
$\{D_1(T_w(x)); x$ leaf of t, w such that $s_w \in sep(x)\}$
can be computed in time $O(h(t) * n)$.

Fact 4

For any leaves x et y, $x \neq y$, there exists a unique vertex in $Lab(x) \cap Lab(y)$ which is a separator for x and y in t.

Find a separator of x and y

Fact 4

For any leaves x et y, $x \neq y$, there exists a unique vertex in $Lab(x) \cap Lab(y)$ which is a separator for x and y in t.

Proof:

$sep(x) = \{s_\epsilon, s_i, \ldots, s_w, \ldots, s_m\}$

$sep(y) = \{s_\epsilon, s_j, \ldots, s_w, \ldots\ldots, s_{m'}\}$

Let w be the longest commun prefixe of m and m':

$t_w = t_{w0}[f(t_{w1}, t_{w2})]$

x and y are not in the same block t_{w0}, t_{w1} or t_{w2},

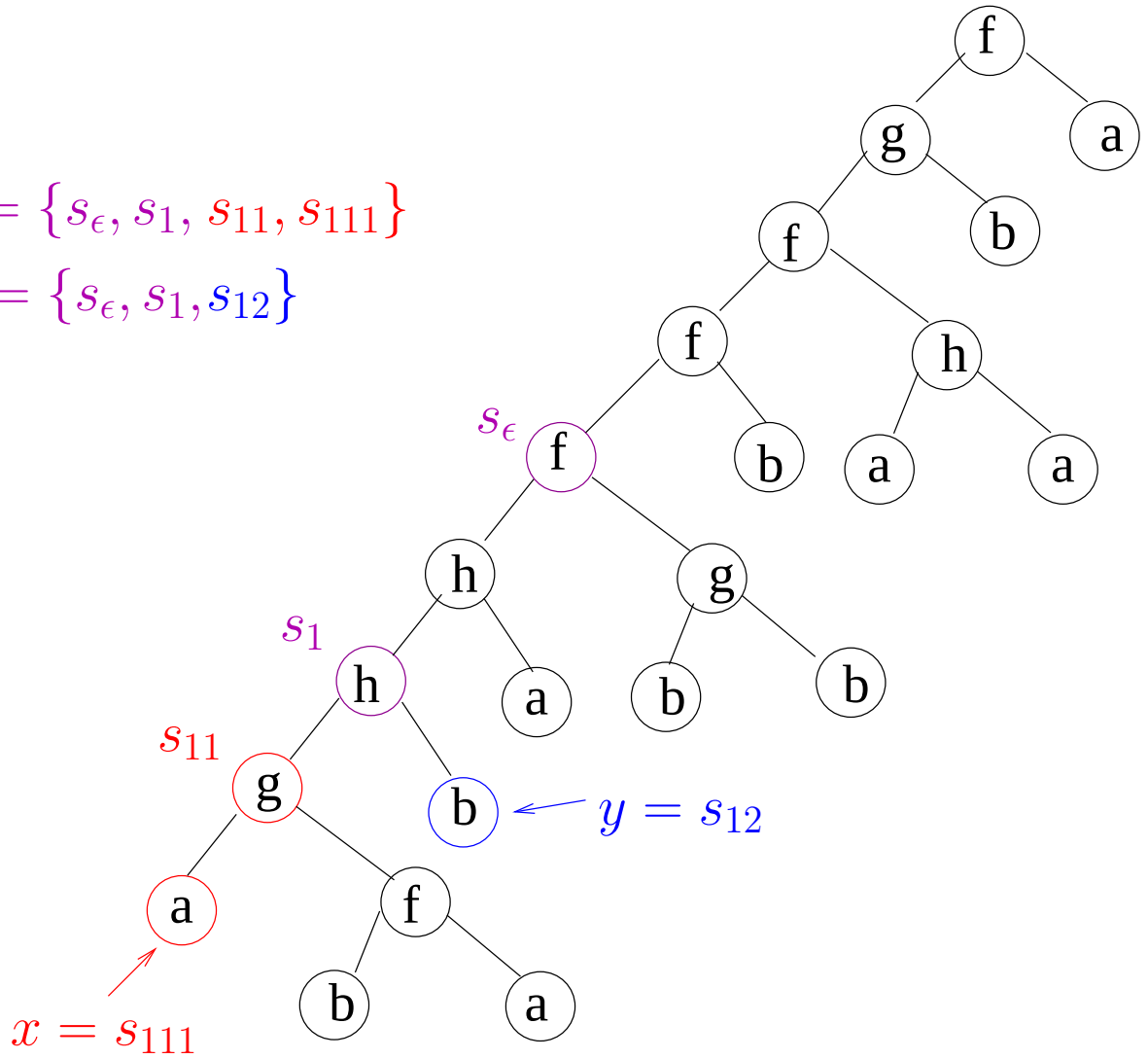
$\Longrightarrow$ w separator of x and y in t_w, which is a subtree of t

$\Longrightarrow$ w separator of x and y in t

Separator node

$sep(x) = \{s_\epsilon, s_1, s_{11}, s_{111}\}$

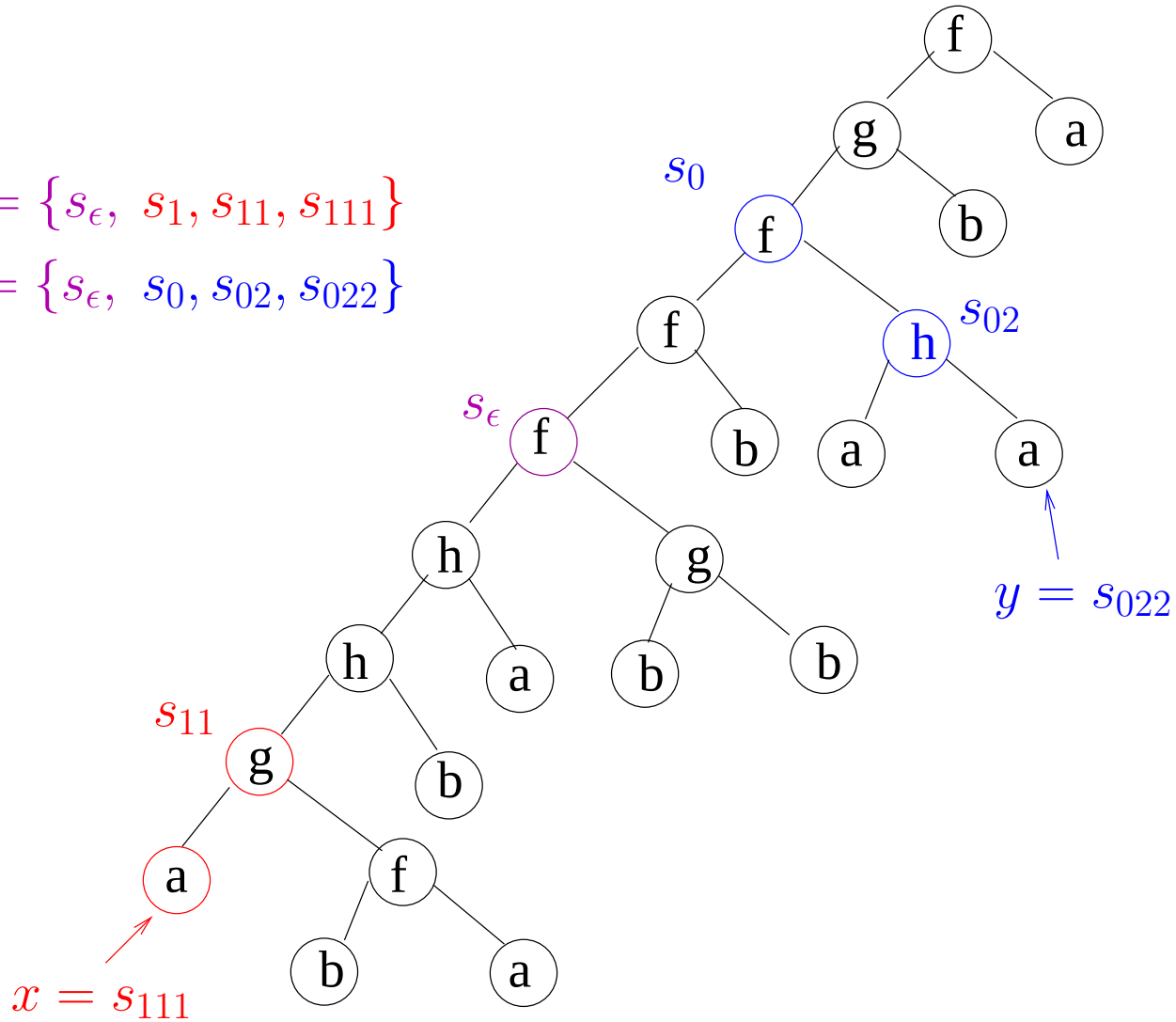
$sep(y) = \{s_\epsilon, s_1, s_{12}\}$



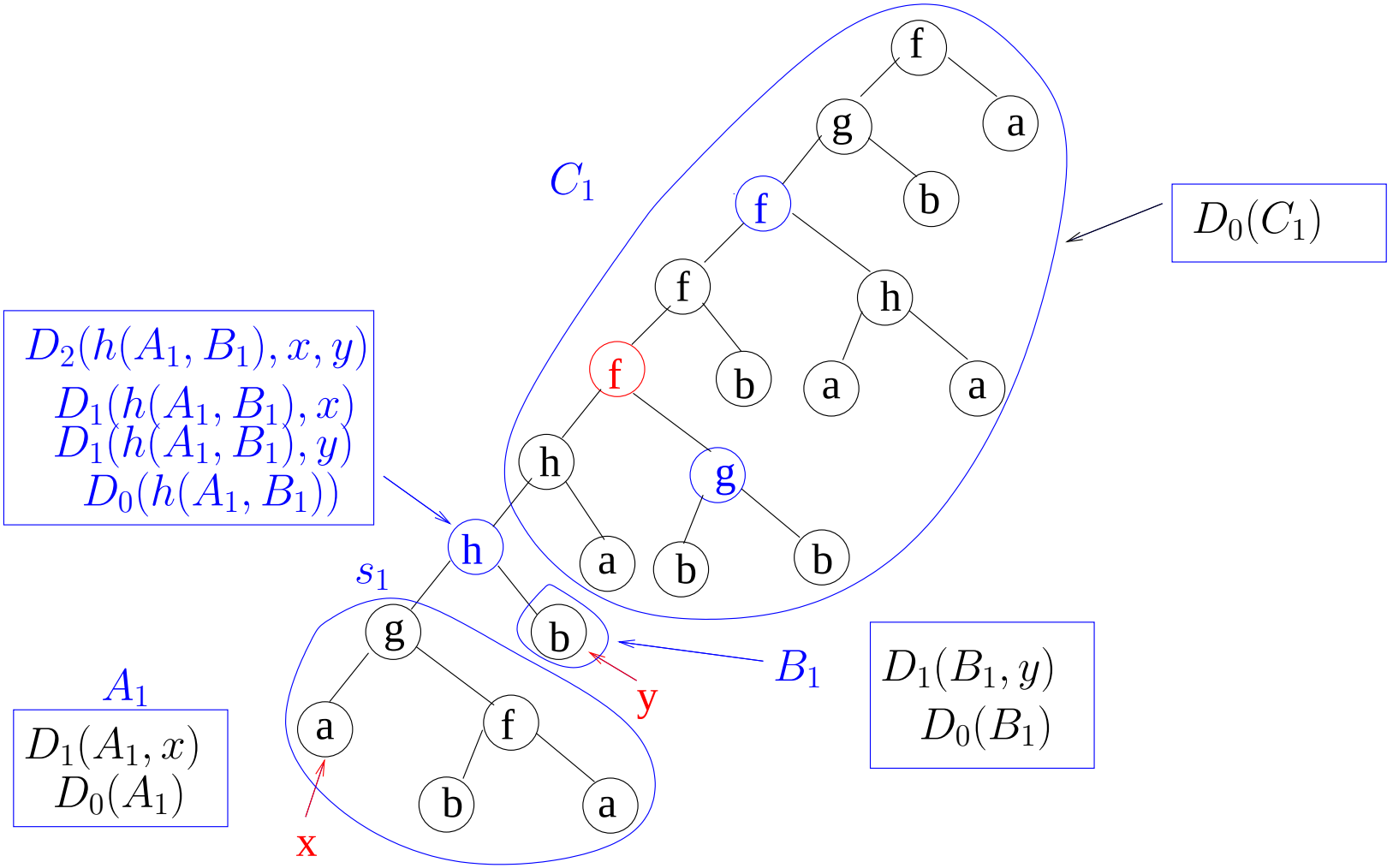
Separator node

$sep(x) = \{s_\epsilon,\ s_1, s_{11}, s_{111}\}$

$sep(y) = \{s_\epsilon,\ s_0, s_{02}, s_{022}\}$



Induction: first step



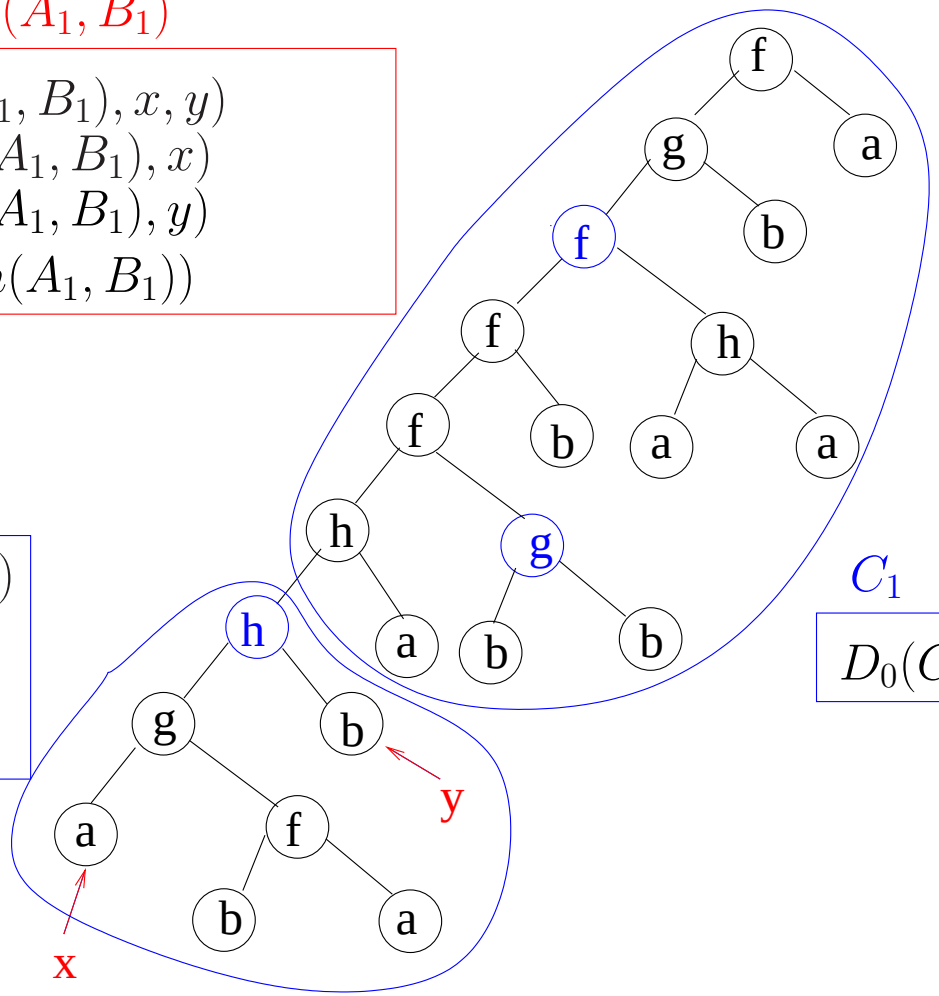
Induction: second step

$t = C_1 \bullet h(A_1, B_1)$

$D_2(C_1 \bullet h(A_1, B_1), x, y)$
$D_1(C_1 \bullet h(A_1, B_1), x)$
$D_1(C_1 \bullet h(A_1, B_1), y)$
$D_0(C_1 \bullet h(A_1, B_1))$

$h(A_1, B_1)$

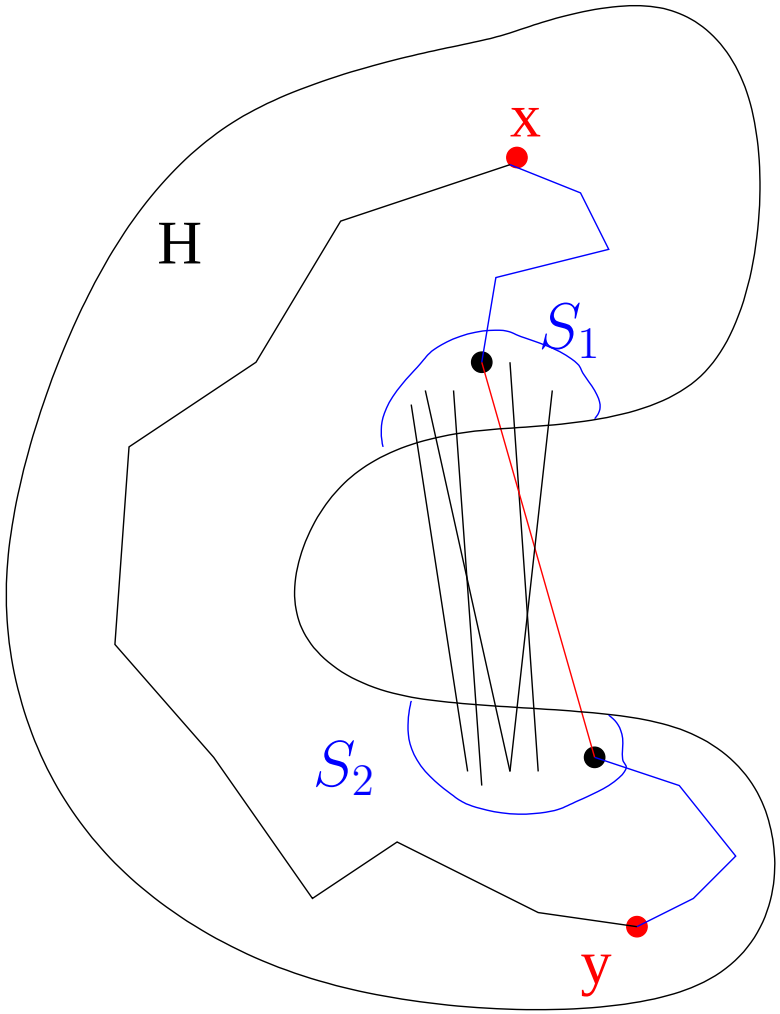
$D_2(h(A_1, B_1), x, y)$
$D_1(h(A_1, B_1), x)$
$D_1(h(A_1, B_1), y)$
$D_0(h(A_1, B_1))$



C_1

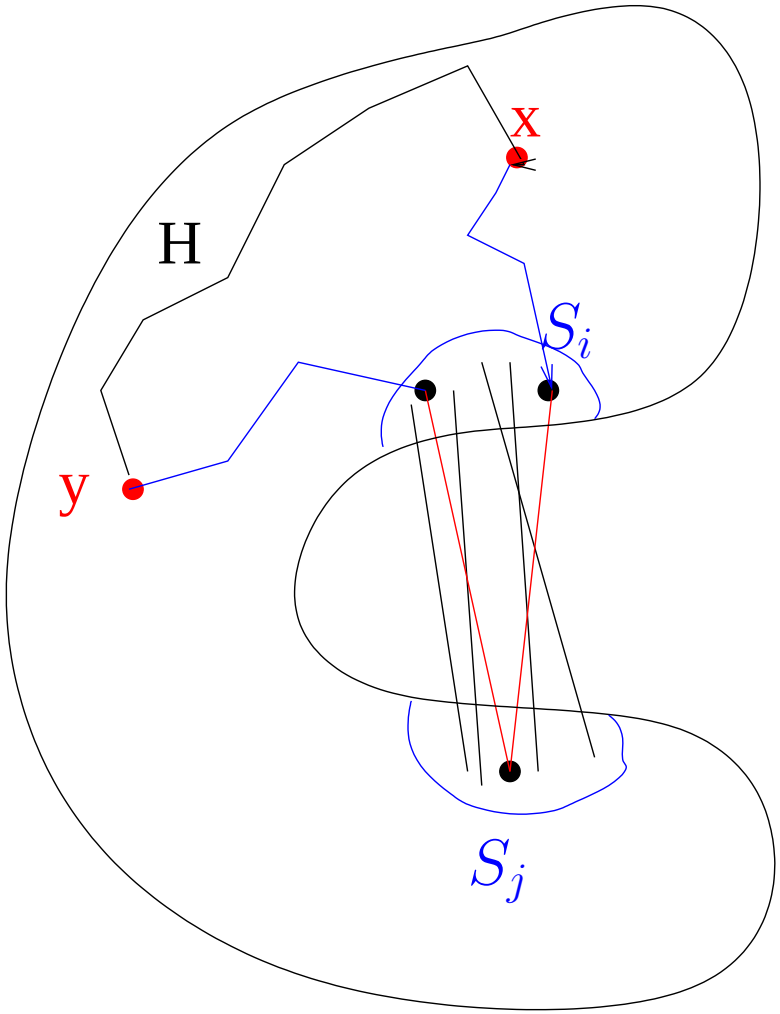
$D_0(C_1)$

distance: $G = add_{i,j}(H)$



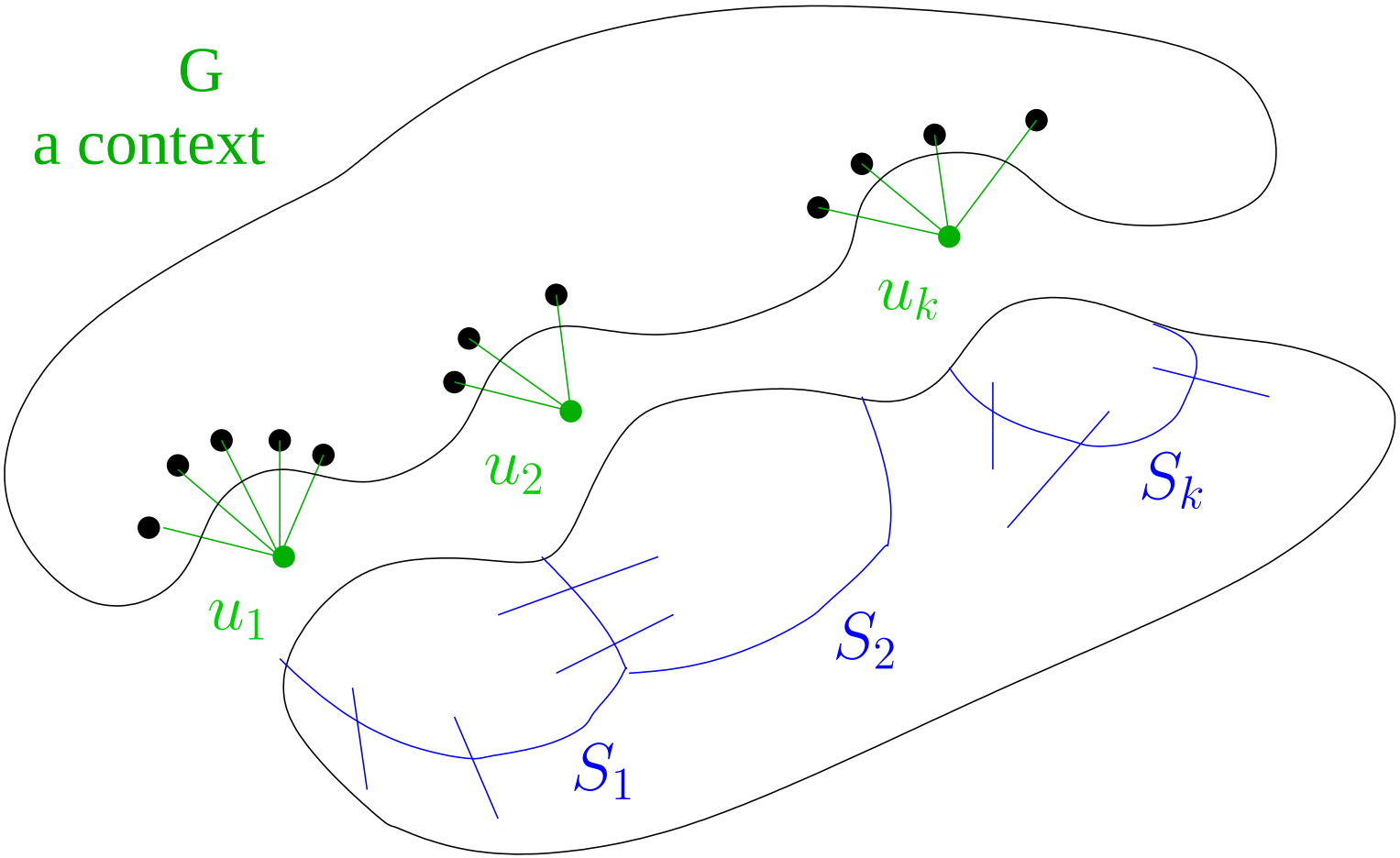
$$d_G(x,y) = min\{d_H(x,y),$$
$$d_H(x,S_i) \ +1+ \ d_H(y,S_j),$$
$$d_H(x,S_i) + 2 + d_H(y,S_i),$$
$$d_H(x,S_j) + 2 + d_H(y,S_j)\}$$

distance: $G = add_{i,j}(H)$



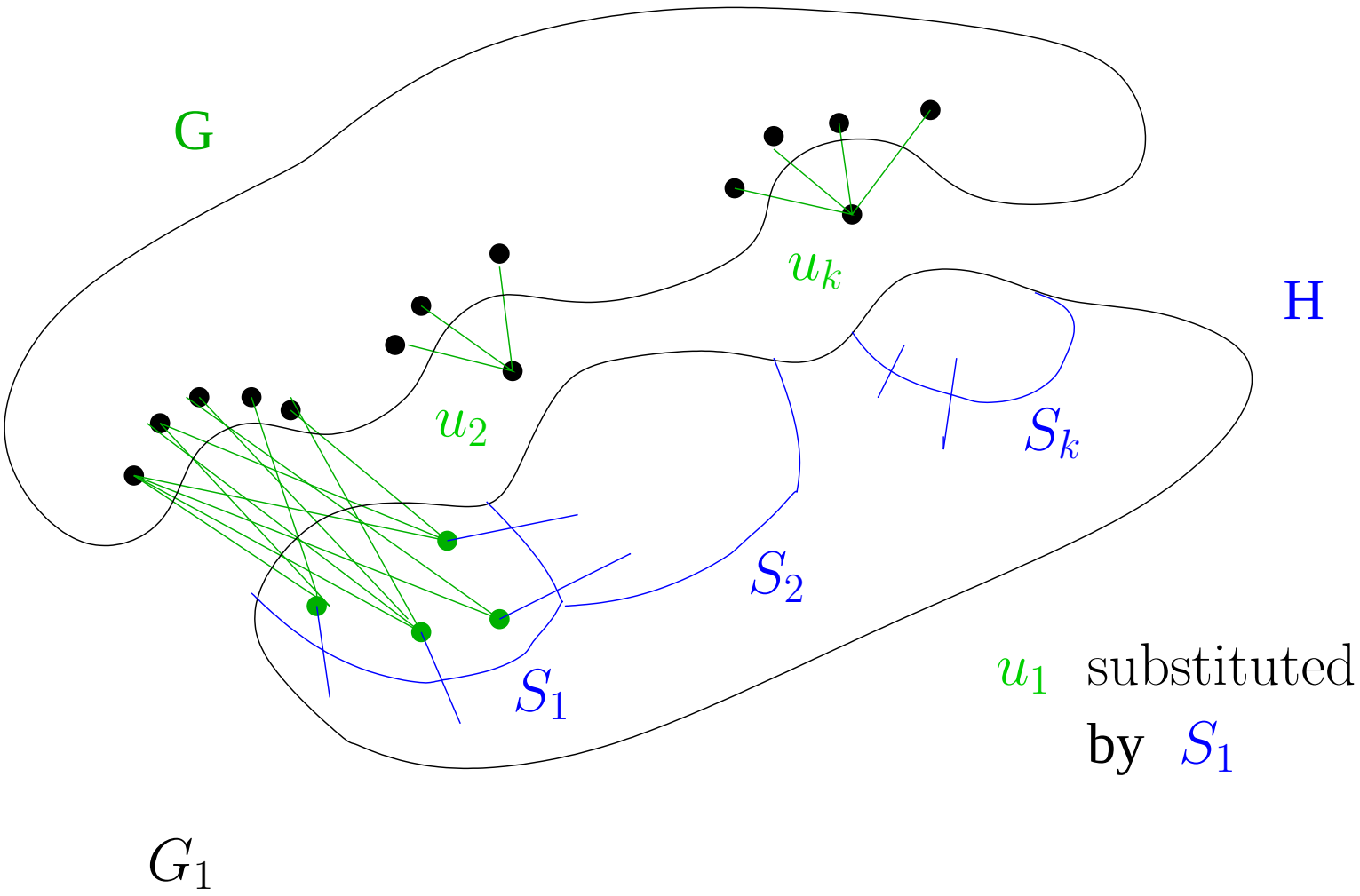
$$d_G(x,y) = min\{d_H(x,y),$$
$$d_H(x,S_i) + 1 + d_H(y,S_j),$$
$$d_H(x,S_i) \ +2+ \ d_H(y,S_i),$$
$$d_H(x,S_j) + 2 + d_H(y,S_j)\}$$

k-substitution

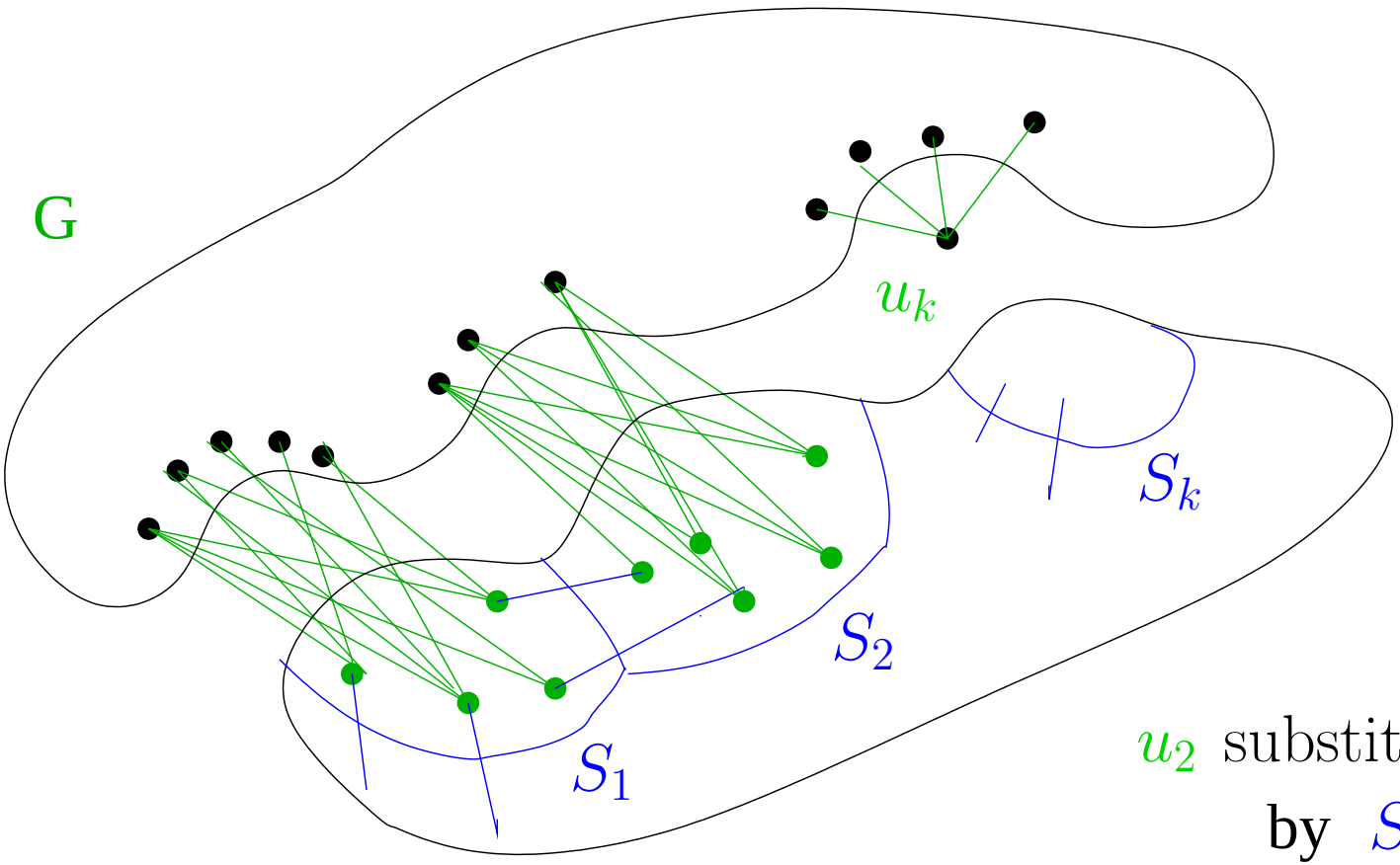


H graph of clique width k

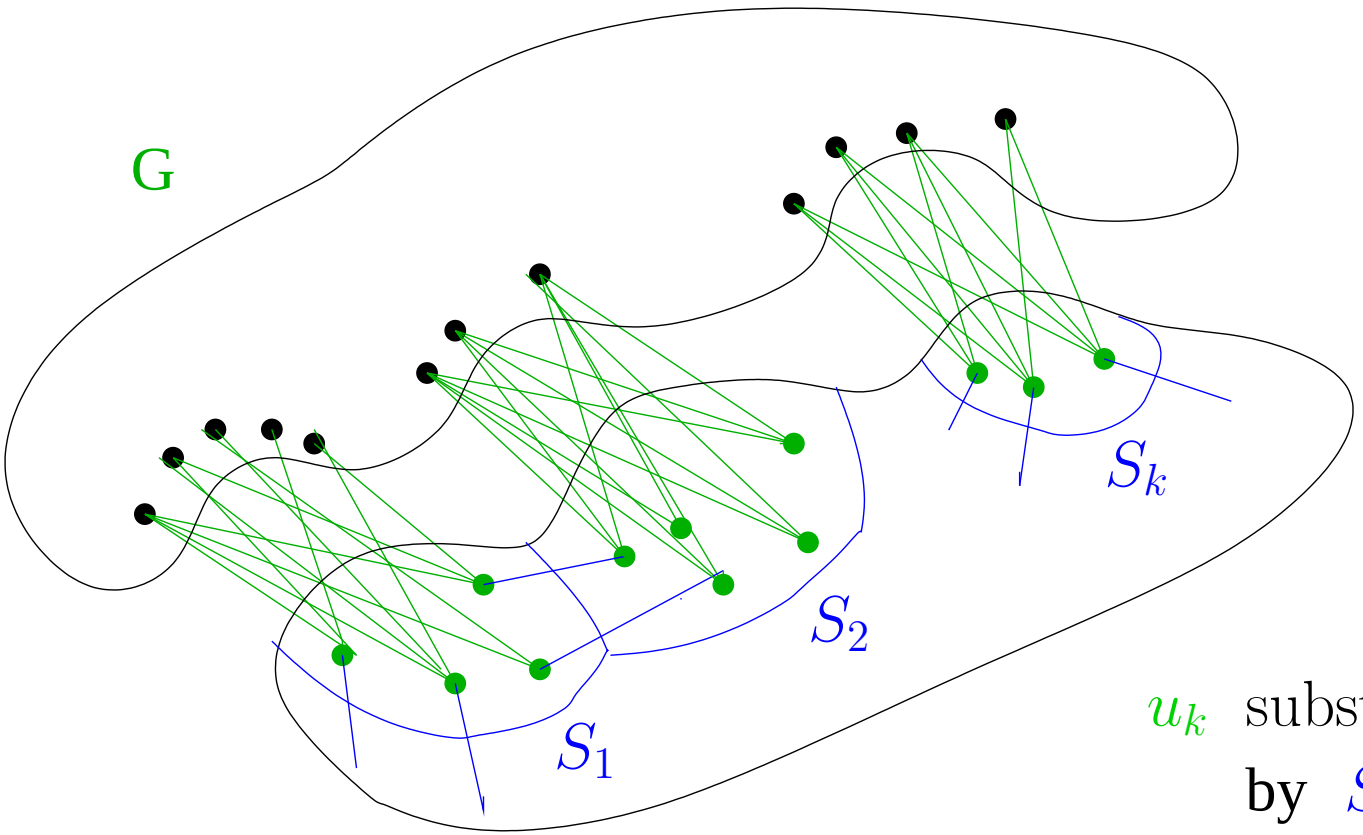
k-substitution



k-substitution



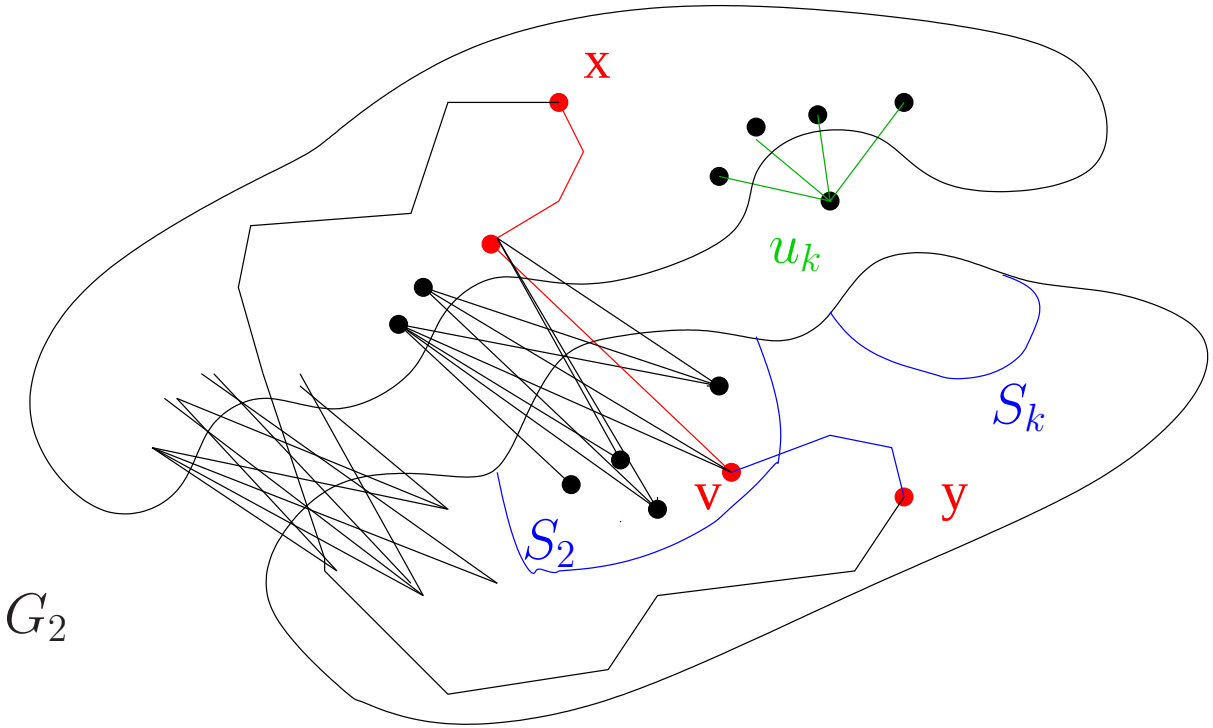
k-substitution



u_k substituted by S_k

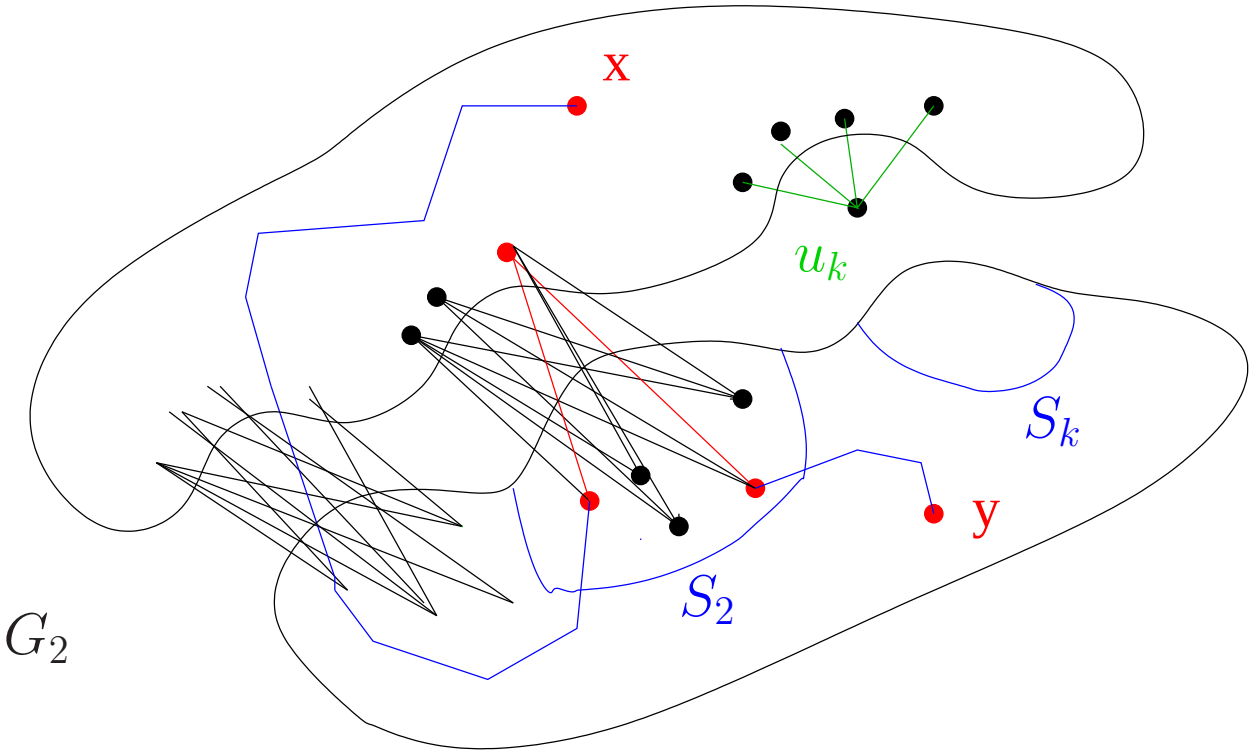
$$G' = G[H/u_1, u_2, \ldots, u_k]$$

distance in $G' = G[u_1, ldots, u_k/H]$



$$d_{G_2}(x,y) = min\{d_{G_1}(x,y),$$
$$d_{G_1}(x,u_2) + d_{G_1}(y,S_2),$$
$$d_{G_1}(y,u_2) + d_{G_1}(x,S_2),$$
$$d_{G_1}(x,S_2) + 2 + d_{G_1}(y,S_2)\}$$

distance in $G' = G[u_1, ldots, u_k/H]$



$$d_{G_2}(x,y) = min\{d_{G_1}(x,y),$$
$$d_{G_1}(x,u_2) + d_{G_1}(y,S_2),$$
$$d_{G_1}(y,u_2) + d_{G_1}(x,S_2),$$
$$d_{G_1}(x,S_2) + 2 \ + d_{G_1}(y,S_2),$$

The inductive functions

$$D_2(t, x, y) = d_G(x, y),$$

$$D_1(t, x) = \begin{cases} (d_G(x, S_1), d_G(x, S_2), \ldots, d_G(x, S_k)), & \text{if } t \text{ is a tree} \\ (d_G(x, S_1), d_G(x, S_2), \ldots, d_G(x, S_k), \\ \quad d_G(x, u_1), \ldots, d_G(x, u_k)), & \text{if } t \text{ is a context} \end{cases}$$

$$D_0(t) = \begin{cases} (d_G(S_1, S_1), d_G(S_1, S_2), \ldots, d_G(S_k, S_k)), & \text{if } t \text{ is a tree} \\ (d_G(S_1, S_1), d_G(S_1, S_2), \ldots, d_G(S_k, S_k), \\ \quad d_G(u_1, S_1), \ldots, d_G(u_k, S_k)), & \text{if } t \text{ is a context} \end{cases}$$

where $d_G(x, S_i) = min\{d_G(x, y), y \in S_i\}$
and $d_G(S_i, S_j) = min\{d_G(x, y), x \in S_i, y \in S_j\}$

Applications

- The distance can be computed in constant time from the labelling scheme
- The number of distinct shortest paths can be computed in constant time from some similar labelling scheme
- This results also applies with oriented, weighted graphs