Finding a Forest in a Tree

The matching problem for wide reactive systems

Giorgio Bacci, Marino Miculan, Romeo Rizzi

Dept. of Computer Science, Aalborg University

Breakfast Talk
(presented @ TGC 2014)

Outline of the talk

- Introduction
- The problem: forest matching
- NP-completeness
- Fixed-parameter Algorithm (the core only)
- Concluding remaks

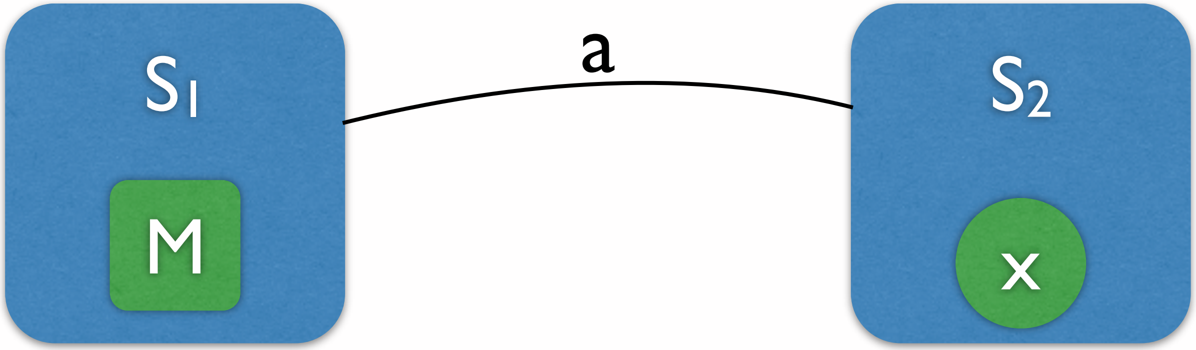
Motivations

- Reactive systems (aka reduction systems) are common in process calculi
- Defined by
  - a set of reduction rules $\quad l(x) \rightarrow r(x)$
  - a contextual closure

$$\frac{l(x) \rightarrow r(x) \quad a = C[l(x)\sigma] \quad b = C[r(x)\sigma]}{a \rightarrow b}$$

- However: not always easy to apply, especially in models of global computing

Ex: Message Passing

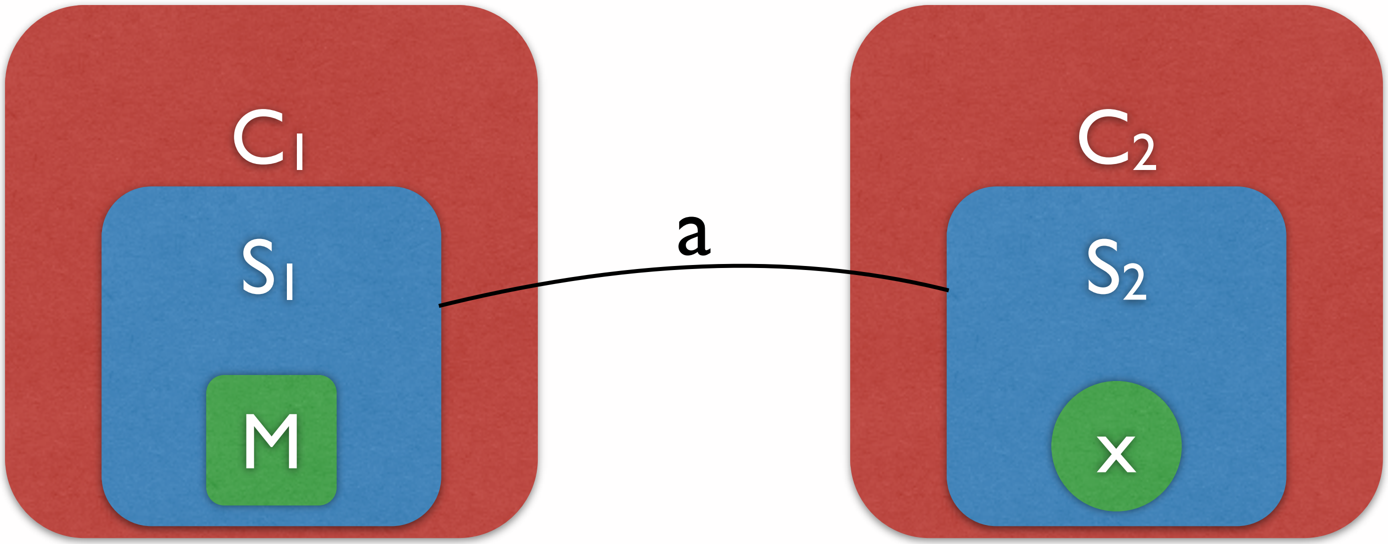


$$\bar{a}\langle M\rangle.P \mid a(x).Q \;\rightarrow\; P \mid Q[M/x]$$

Ex: Message Passing

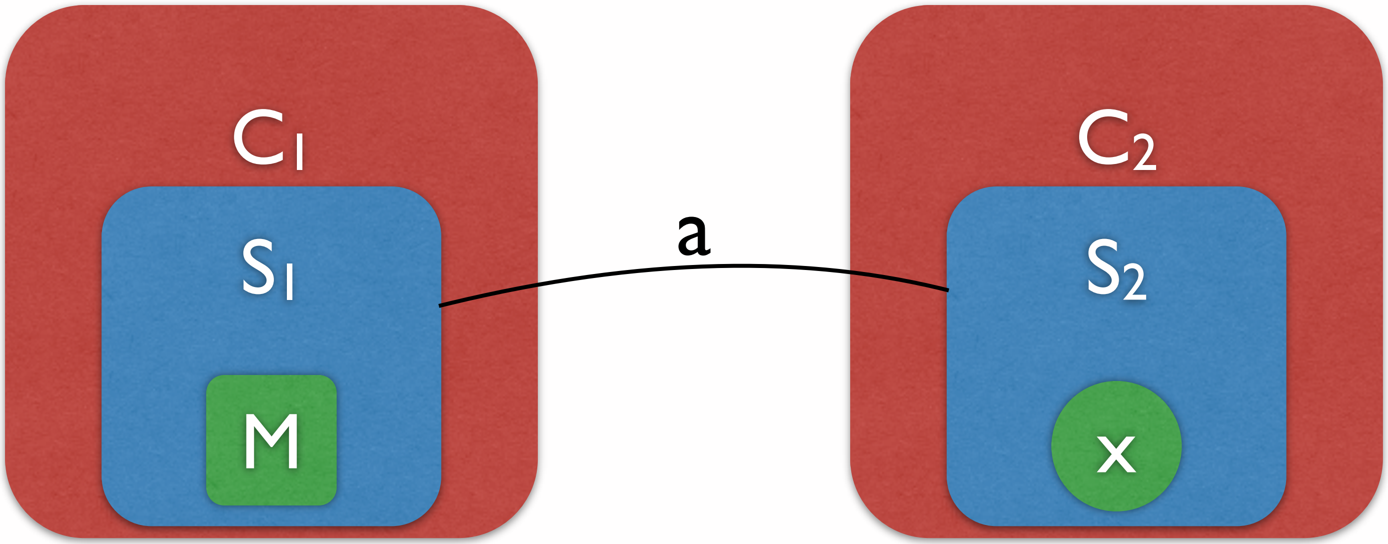
$\bar{a}\langle M\rangle.P \mid a(x).Q \;\rightarrow\; P \mid Q[M/x]$

Ex: Remote Message Passing



$$C_1[\bar{a}\langle M\rangle.P] \mid C_2[a(x).Q] \rightarrow C_1[P] \mid C_2[Q\{M/x\}]$$

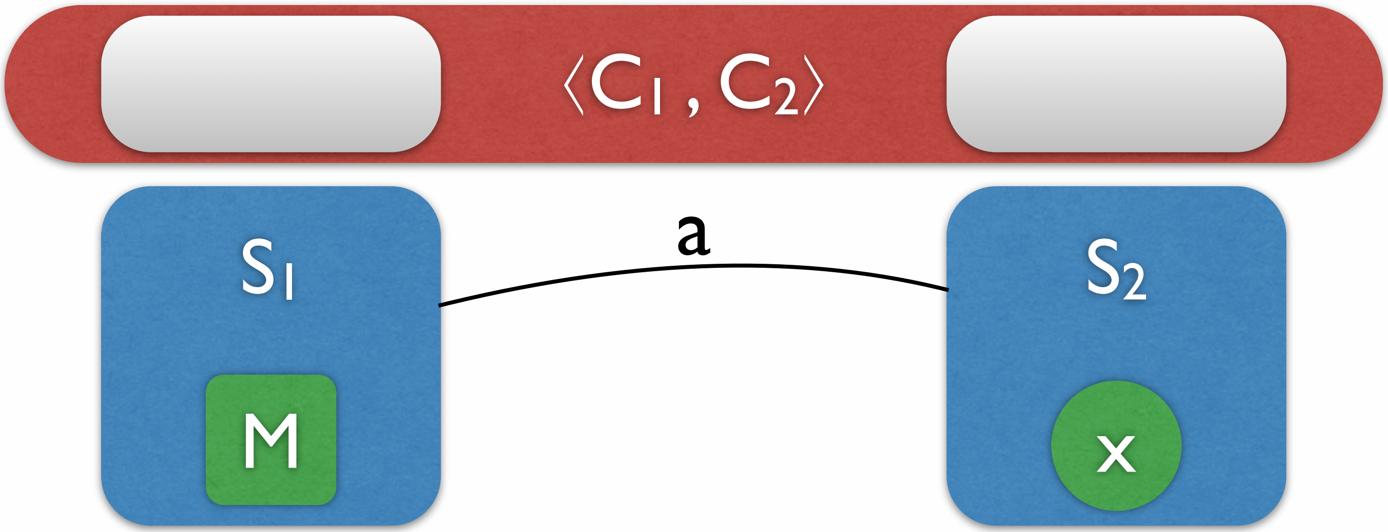
Ex: Remote Message Passing



$$C_1[\bar{a}\langle M\rangle.P] \mid C_2[a(x).Q] \;\rightarrow\; C_1[P] \mid C_2[Q\{M/x\}]$$

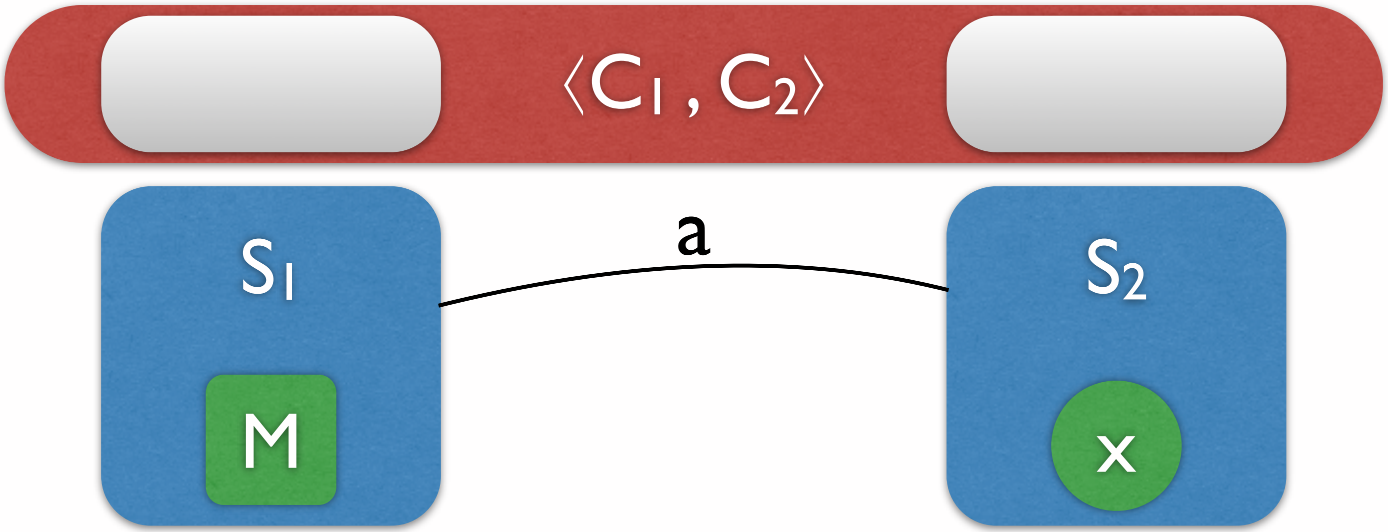
Infinite reduction rules are needed
(one for each pair C_1,C_2)

Ex: Remote Message Passing



$$\langle \bar{a}\langle M\rangle.P, a(x).Q\rangle \rightarrow \langle P, Q\{M/x\}\rangle$$

Ex: Remote Message Passing



$$\langle \bar{a}\langle M\rangle.P, a(x).Q\rangle \rightarrow \langle P, Q\{M/x\}\rangle$$

Just one **wide** reduction rule
Context has two holes

Wide reactive systems

A set of **wide** reaction rules

$$\langle l_1(\boldsymbol{x}_1),\ldots,l_n(\boldsymbol{x}_n)\rangle \to \langle r_1(\boldsymbol{y}_1),\ldots,r_n(\boldsymbol{y}_n)\rangle$$

$$\text{with } \{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n\} \subseteq \{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n\}$$

Contextual closure:

$$\frac{\langle l_1(\boldsymbol{x}_1),\ldots,l_n(\boldsymbol{x}_n)\rangle \to \langle r_1(\boldsymbol{y}_1),\ldots,r_n(\boldsymbol{y}_n)\rangle \\ a{=}C[l_1(\boldsymbol{x}_1)\sigma,\ldots,l_n(\boldsymbol{x}_n)\sigma] \qquad b{=}C[r_1(\boldsymbol{y}_1)\sigma,\ldots,r_n(\boldsymbol{y}_n)\sigma]}{a \to b}$$

Wide vs. Non-wide

Wide vs. Non-wide

- WRSs are more expressive than non-WRSs

Wide vs. Non-wide

- WRSs are more expressive than non-WRSs
  - non-wide are a particular case of wide (width = 1)

Wide vs. Non-wide

- WRSs are more expressive than non-WRSs
  - non-wide are a particular case of wide (width = 1)
  - WRSs need less rules (often, finite instead of infinite)

Wide vs. Non-wide

- WRSs are more expressive than non-WRSs
  - non-wide are a particular case of wide (width = 1)
  - WRSs need less rules (often, finite instead of infinite)
- Pattern matching in WRSs is more complex

Wide vs. Non-wide

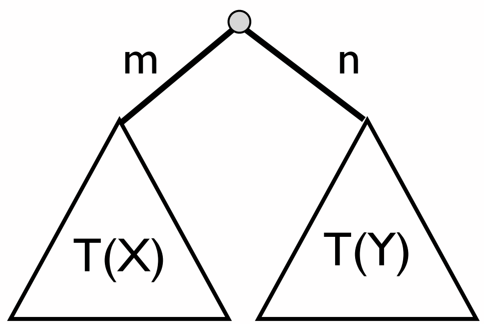
- WRSs are more expressive than non-WRSs
  - non-wide are a particular case of wide (width = 1)
  - WRSs need less rules (often, finite instead of infinite)
- Pattern matching in WRSs is more complex
  - The bad news: NP-complete

Wide vs. Non-wide

- WRSs are more expressive than non-WRSs
  - non-wide are a particular case of wide (width = 1)
  - WRSs need less rules (often, finite instead of infinite)
- Pattern matching in WRSs is more complex
  - The bad news: NP-complete
  - The good news: combinatorial explosion depends on redex width only (constant and usually ≤ 3)

Linear Context Trees

$T(X) ::=$	0	empty tree
	x	leaf $(x \in X)$
	$m[\,T(X)]$	labeled tree
	$T(X') \mid T(X'')$	siblings $(X = X' \uplus X'')$



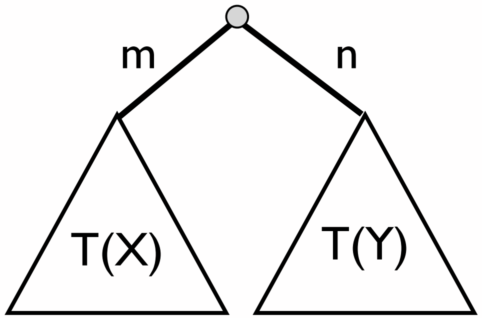
$m[\,T(X)] \mid n[T(Y)]$

Linear Context Trees

$T(X) ::=$	0	empty tree
	x	leaf $(x \in X)$
	$m[\,T(X)\,]$	labeled tree
	$T(X') \mid T(X'')$	siblings $(X = X' \uplus X'')$

Unordered

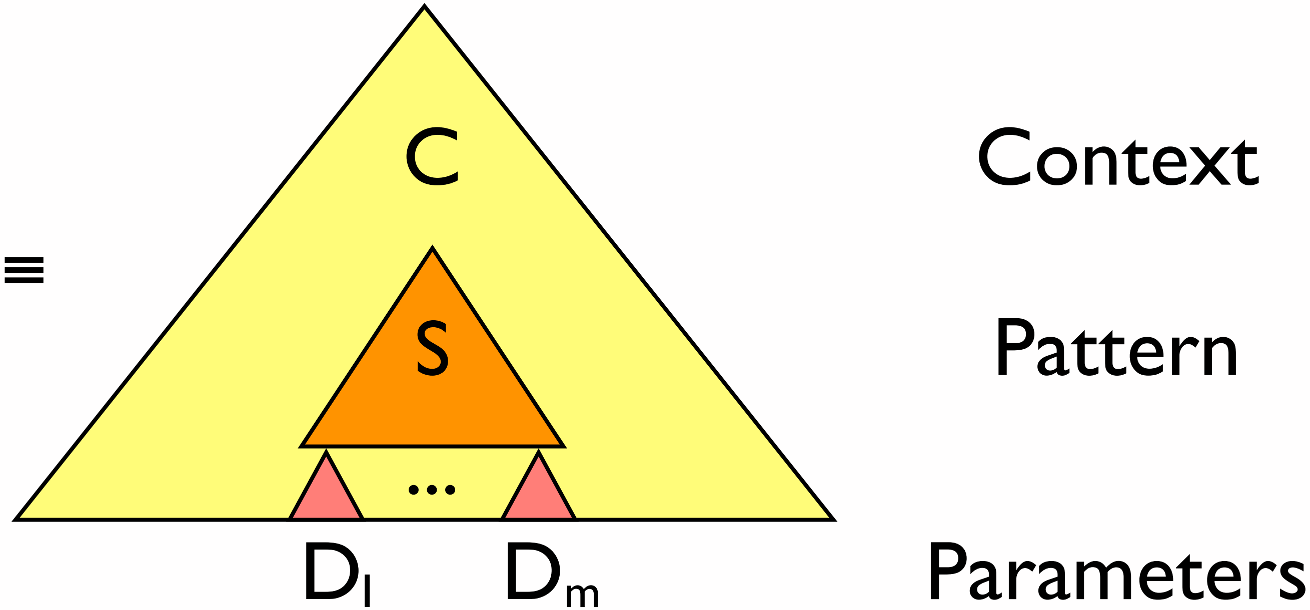
(up to structural congruence ≡)



$m[\,T(X)\,] \mid n[T(Y)]$

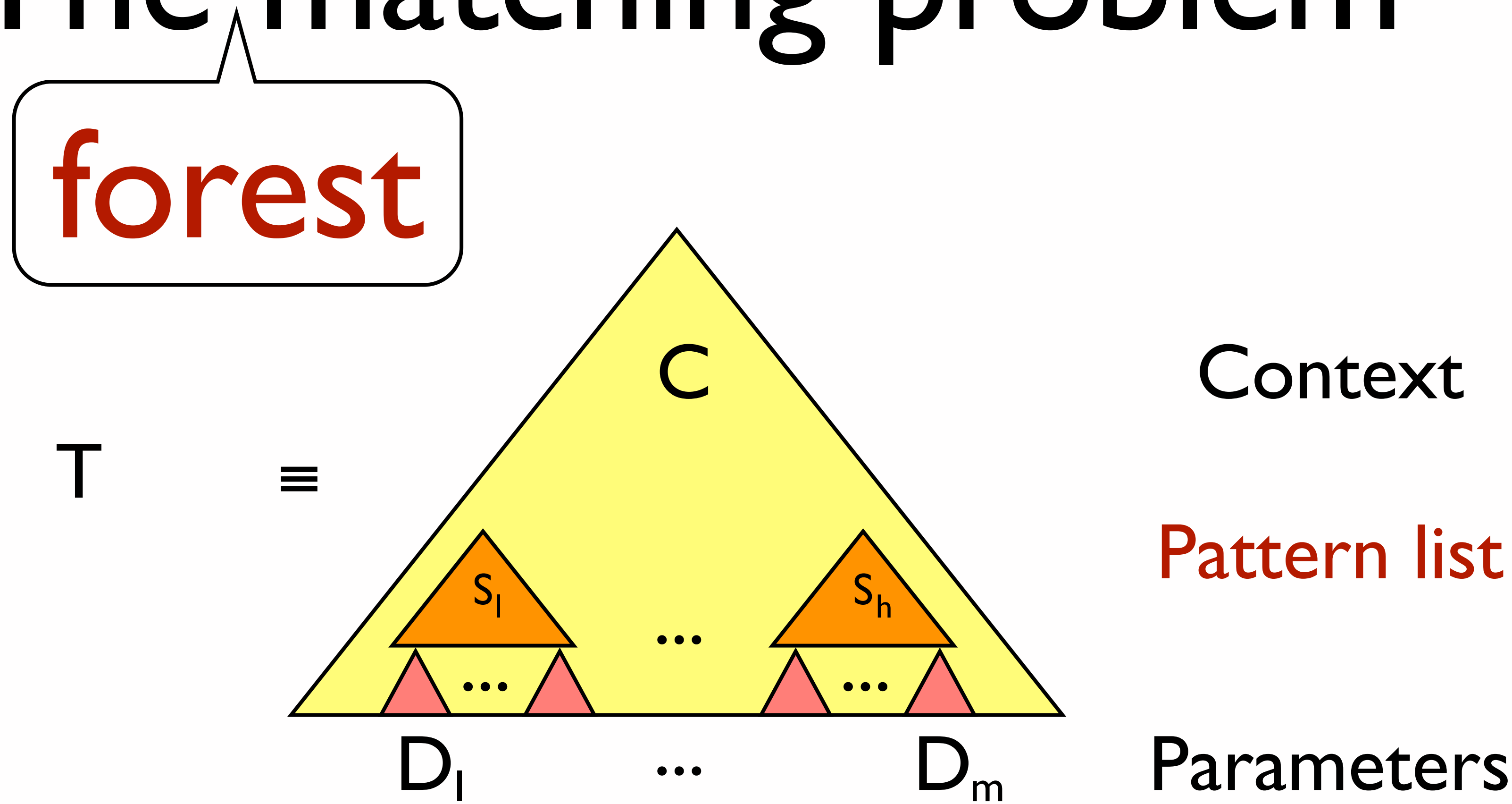
The matching problem

$T \equiv$



$$T \equiv (C\{S/z\})\{D_1,\ldots,D_m/x_1,\ldots,x_m\}$$

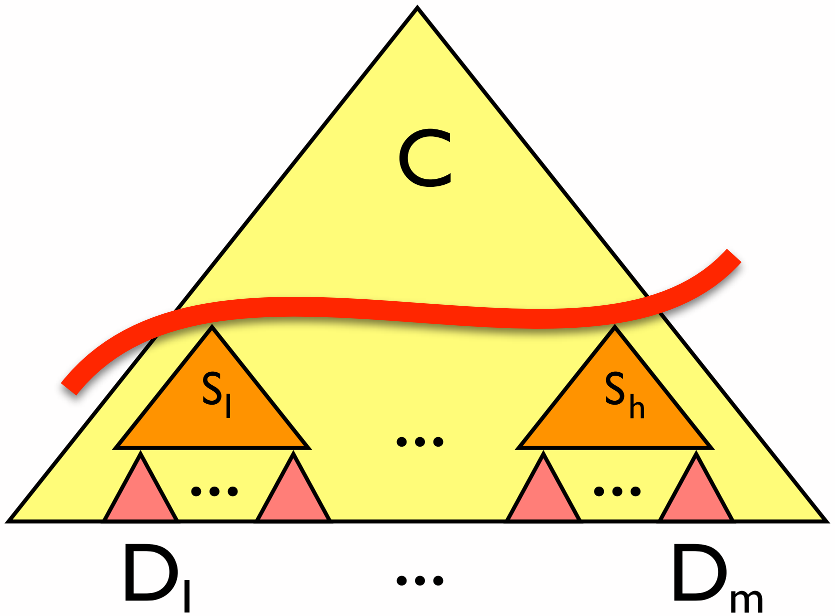
The matching problem



$$T \equiv (C\{S_1,\ldots,S_h/z_1,\ldots,z_h\})\{D_1,\ldots,D_m/x_1,\ldots,x_m\}$$

Forest matching is NP-complete

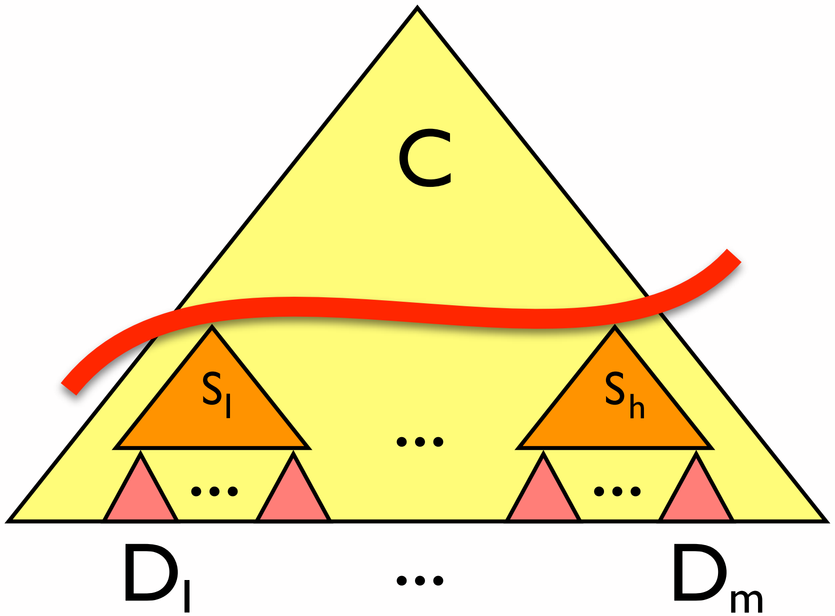
- *Tree matching problem* is in P (subtree isomorphim algorithm [Matula'78])
- *Forest matching problem* is NP-complete!



The pattern matchings must not overlap = ***antichain*** in T

Forest matching is NP-complete

- *Tree matching problem* is in P (subtree isomorphim algorithm [Matula'78])
- *Forest matching problem* is NP-complete!



≠ sub-forest isomorphism

The pattern matchings must not overlap

=

antichain in T

Rainbow antichain

Instance: a tree T colored on palette P of colors.
Problem: to find a P-colorful antichain in T.

Rainbow antichain

Instance: a tree T colored on palette P of colors.
Problem: to find a P-colorful antichain in T.

Proof sketch of NP-hardness

Rainbow antichain

Instance: a tree T colored on palette P of colors.
Problem: to find a P-colorful antichain in T.

Proof sketch of NP-hardness

reduction from 3-SAT

Rainbow antichain

Instance: a tree T colored on palette P of colors.
Problem: to find a P-colorful antichain in T.

reduction from 3-SAT

Proof sketch of NP-hardness

$$\overbrace{(\overline{x} \vee y \vee \overline{z})}^{C_1} \wedge \overbrace{(x \vee y \vee z)}^{C_2}$$

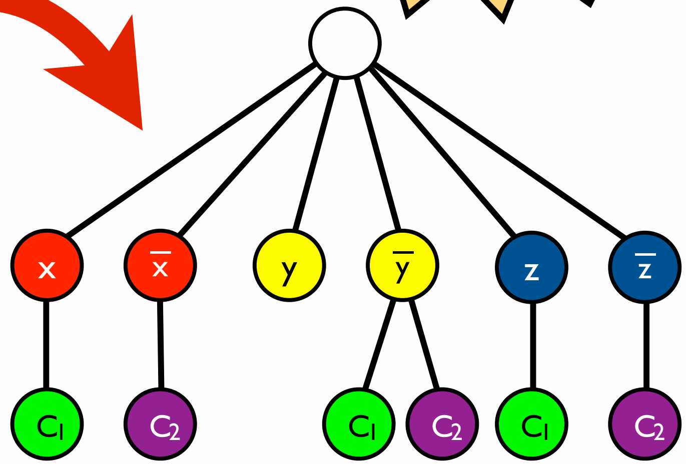
Rainbow antichain

Instance: a tree T colored on palette P of colors.
Problem: to find a P-colorful antichain in T.

Proof sketch of NP-hardness

reduction from 3-SAT

$$\overbrace{(\bar{x} \vee y \vee \bar{z})}^{C_1} \wedge \overbrace{(x \vee y \vee z)}^{C_2}$$



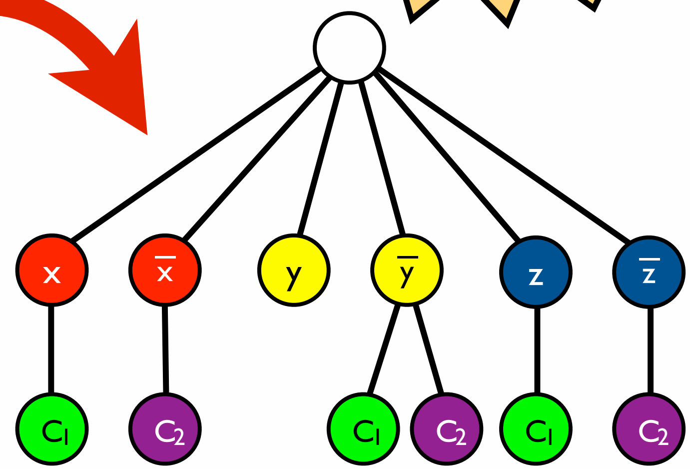
Rainbow antichain

Instance: a tree T colored on palette P of colors.
Problem: to find a P-colorful antichain in T.

Proof sketch of NP-hardness

reduction from 3-SAT

$$\overbrace{(\bar{x} \vee y \vee \bar{z})}^{C_1} \wedge \overbrace{(x \vee y \vee z)}^{C_2}$$



every truth assignment satisfying
the formula induces a rainbow antichain
...and *vice versa*

Fixed-parameter Tractability

How to cope with computational intractability?

approximation algorithms, average case analysis,
randomized algorithms, heuristics methods, etc...

Fixed-parameter Tractability

How to cope with computational intractability?

approximation algorithms, average case analysis,
randomized algorithms, heuristics methods, etc...

FPT's basic observation:

"for many hard problems, the seemingly inherent combinatorial explosion really can be restricted to a 'small part' of the input, the parameter"

[Downey-Fellows'99]

Parameterized algorithm for Rainbow antichain

- Rainbow antichain is the core problem behind the forest matching problem
- Rainbow antichain is solved in 2 steps:

1. Reduction to *kernel-size* of the input tree
2. Exhaustive search of a rainbow antichain

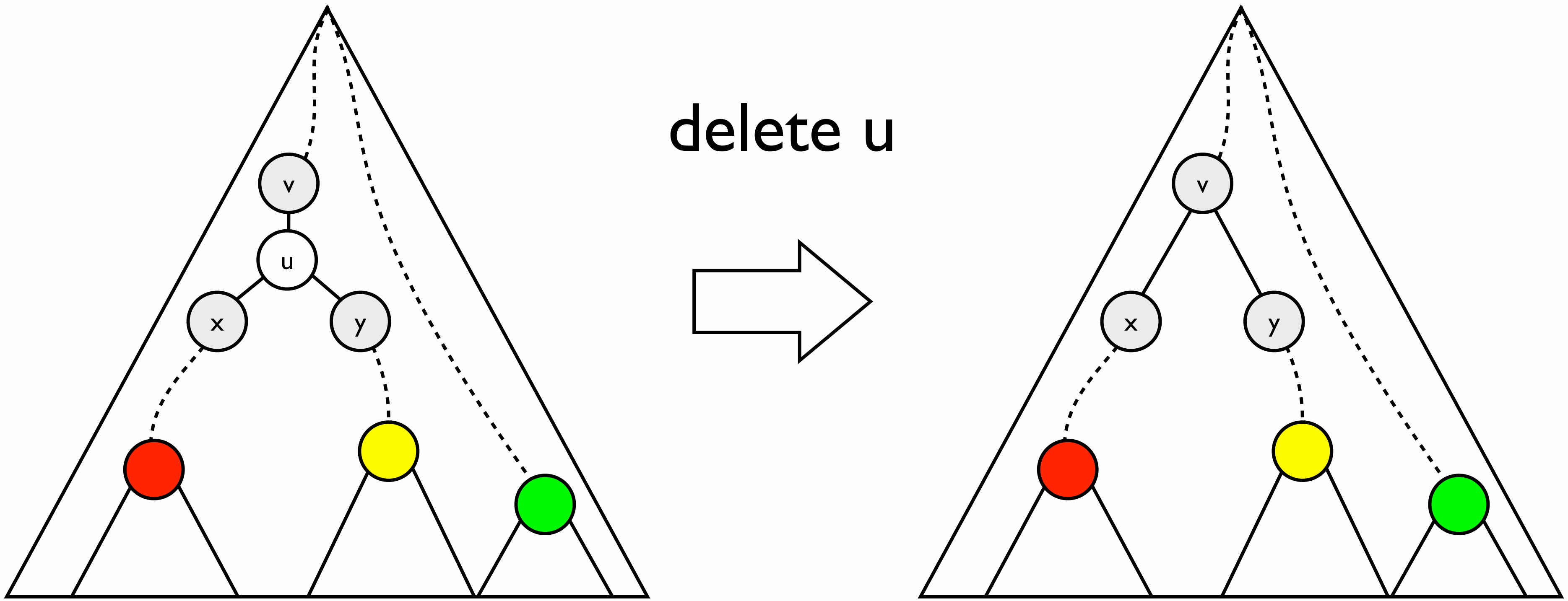
Parameterized algorithm for Rainbow antichain

parameter:
size of P

- Rainbow antichain is the core problem behind the forest matching problem
- Rainbow antichain is solved in 2 steps:

1. Reduction to *kernel-size* of the input tree
2. Exhaustive search of a rainbow antichain

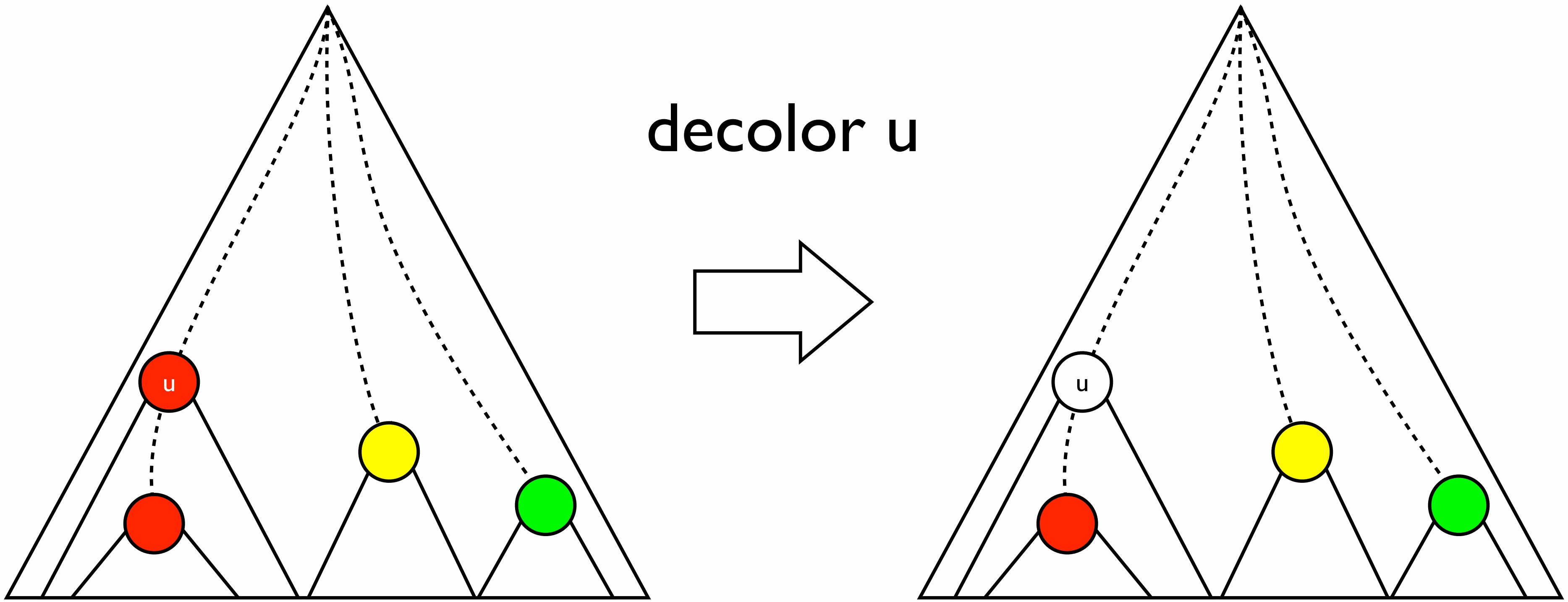
1. Reduction by decoloring



(*) we will assume the root cannot be deleted

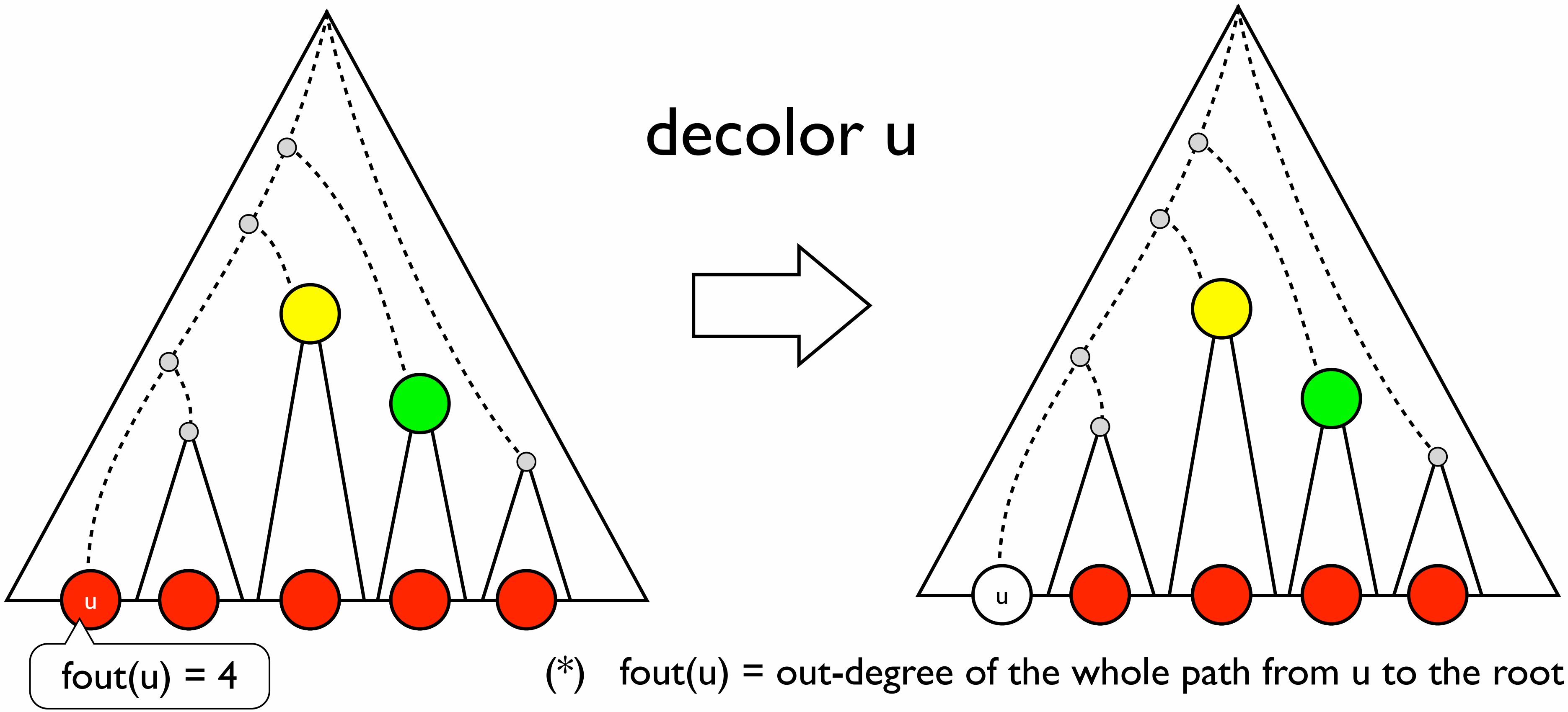
Deletion of uncolored nodes does not influence existence of rainbow antichain!

Decoloring (rule 1)



Ancestors with the same color can be decolored

Decoloring (rule 2)

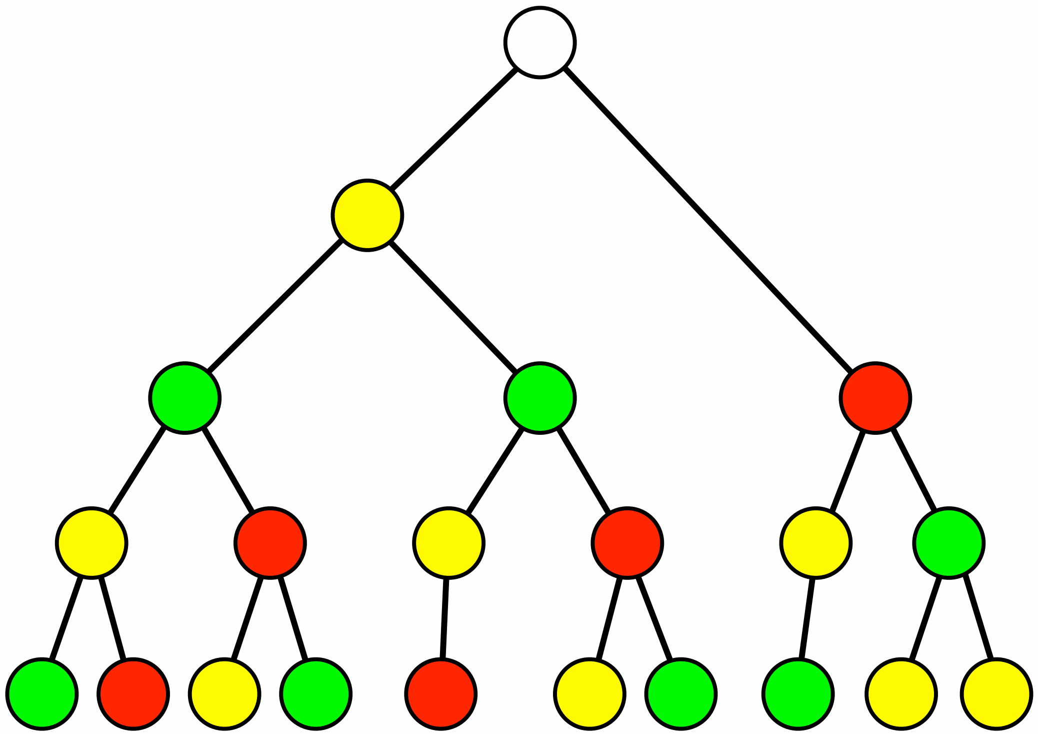


(*) fout(u) = out-degree of the whole path from u to the root

If the tree has all leaves of the same color,
then leaves with fan-out ≥ |P| can be decolored

How to apply the rules?

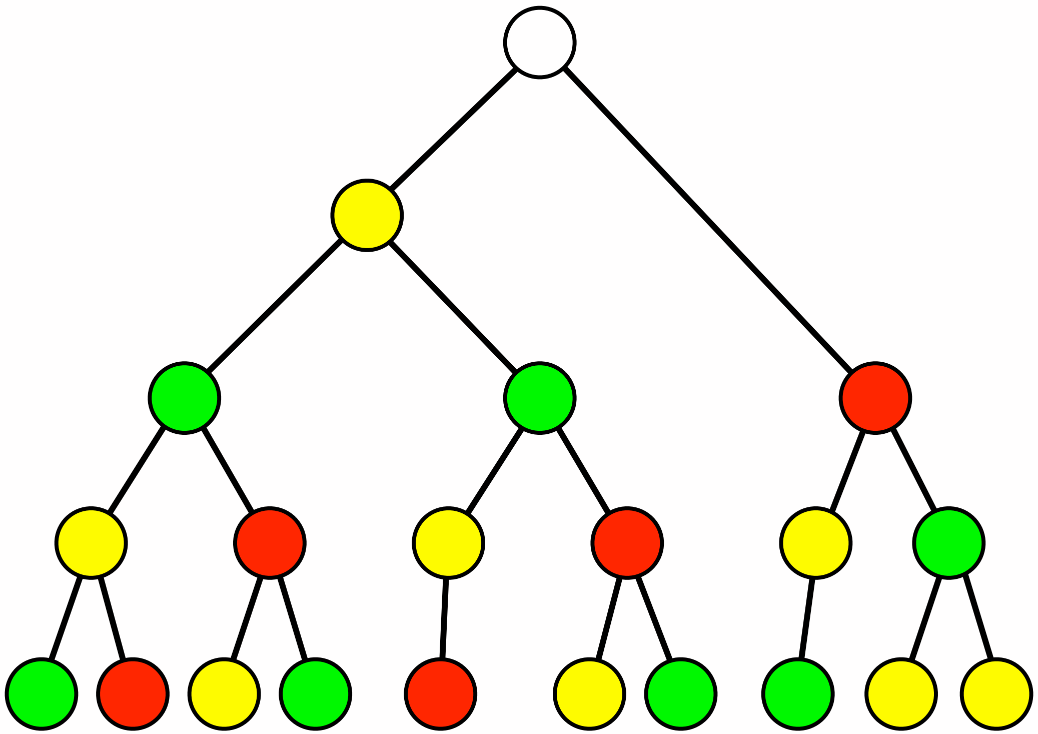
(an example of reduction to kernel size)



P = { red, yellow, green }

How to apply the rules?

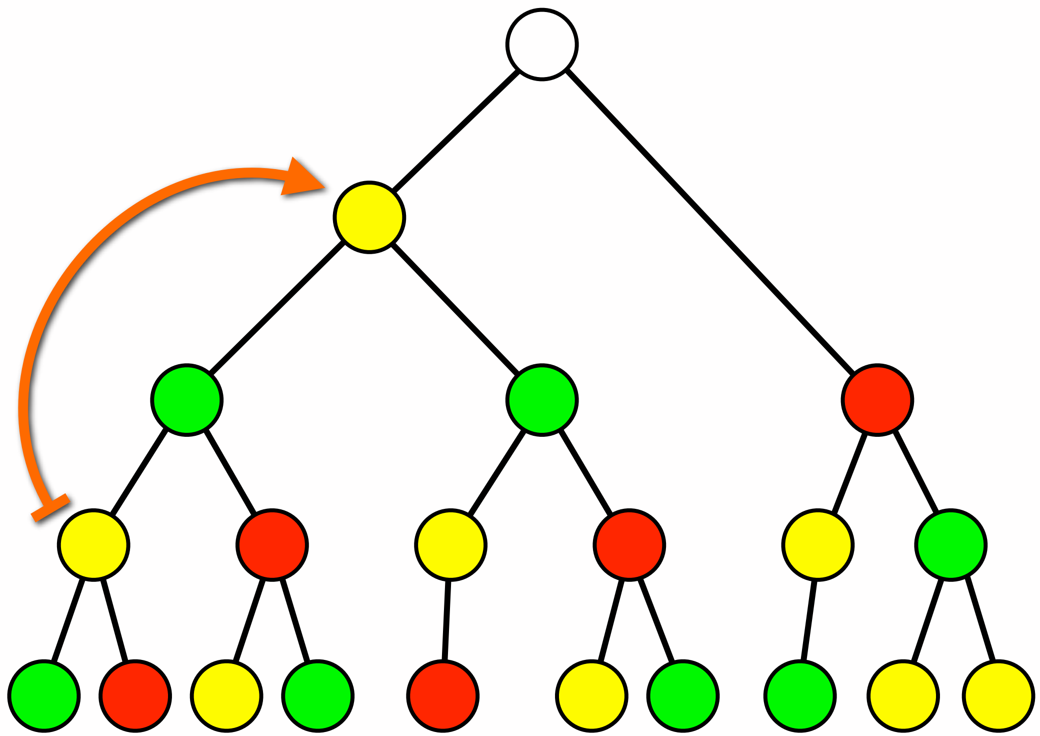
(an example of reduction to kernel size)



Rule 1: Decoloring of ancestors of the same color

How to apply the rules?

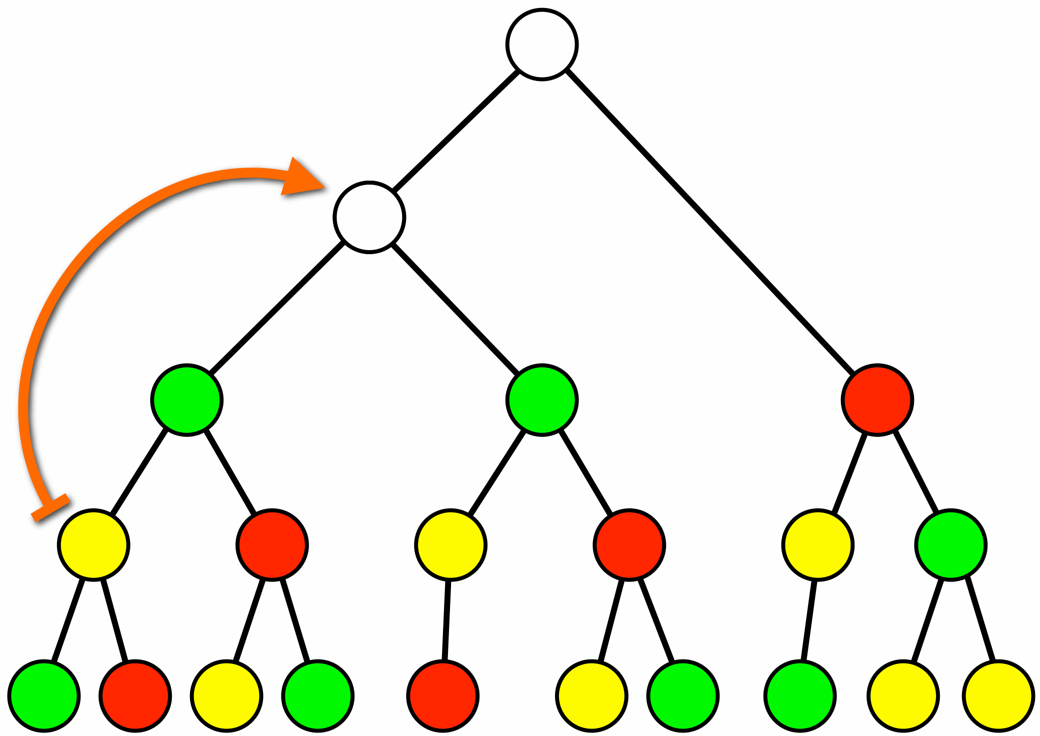
(an example of reduction to kernel size)



Rule 1: Decoloring of ancestors of the same color

How to apply the rules?

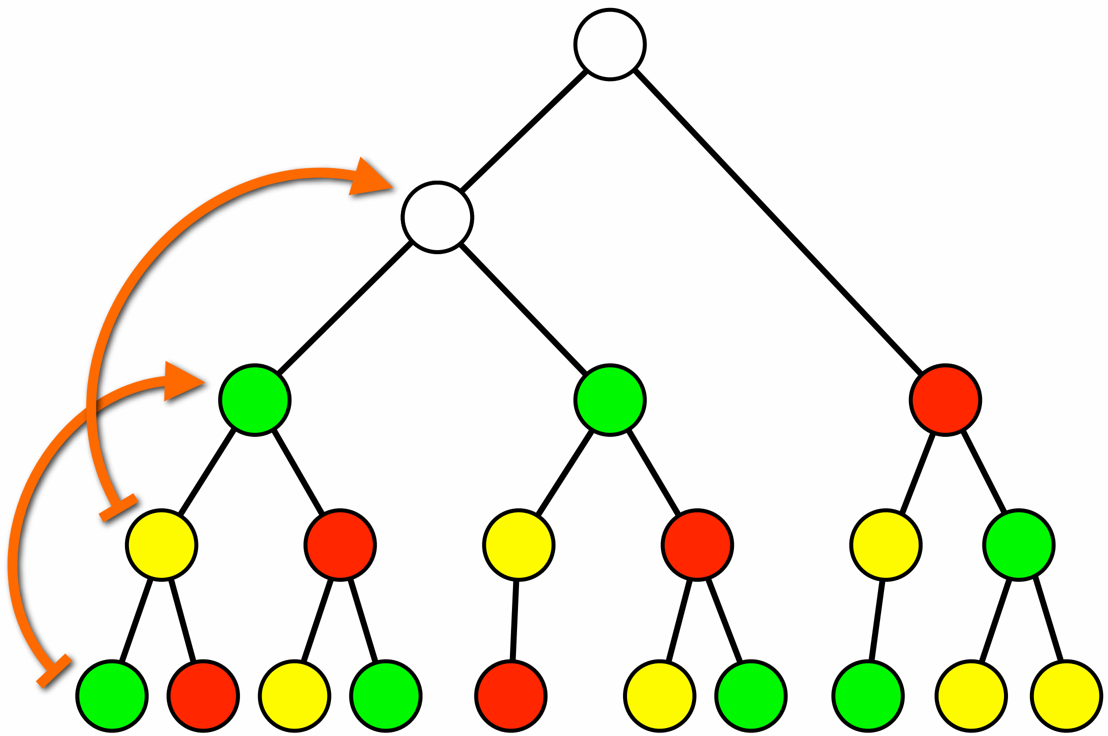
(an example of reduction to kernel size)



Rule 1: Decoloring of ancestors of the same color

How to apply the rules?

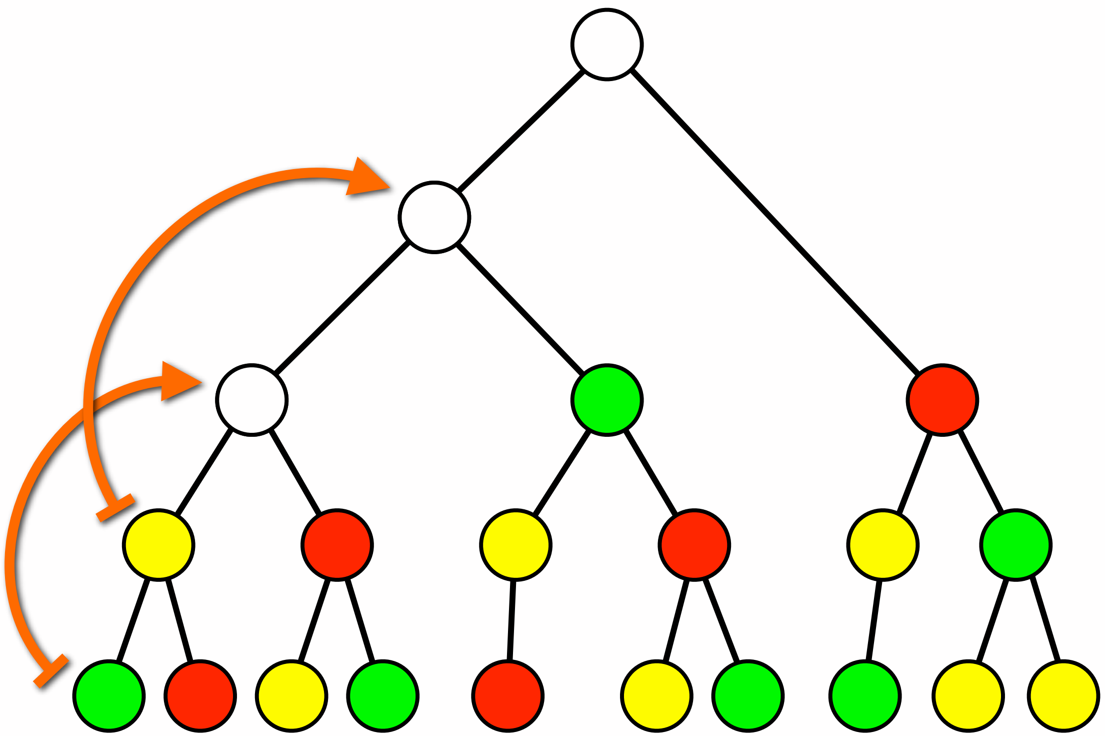
(an example of reduction to kernel size)



Rule 1: Decoloring of ancestors of the same color

How to apply the rules?

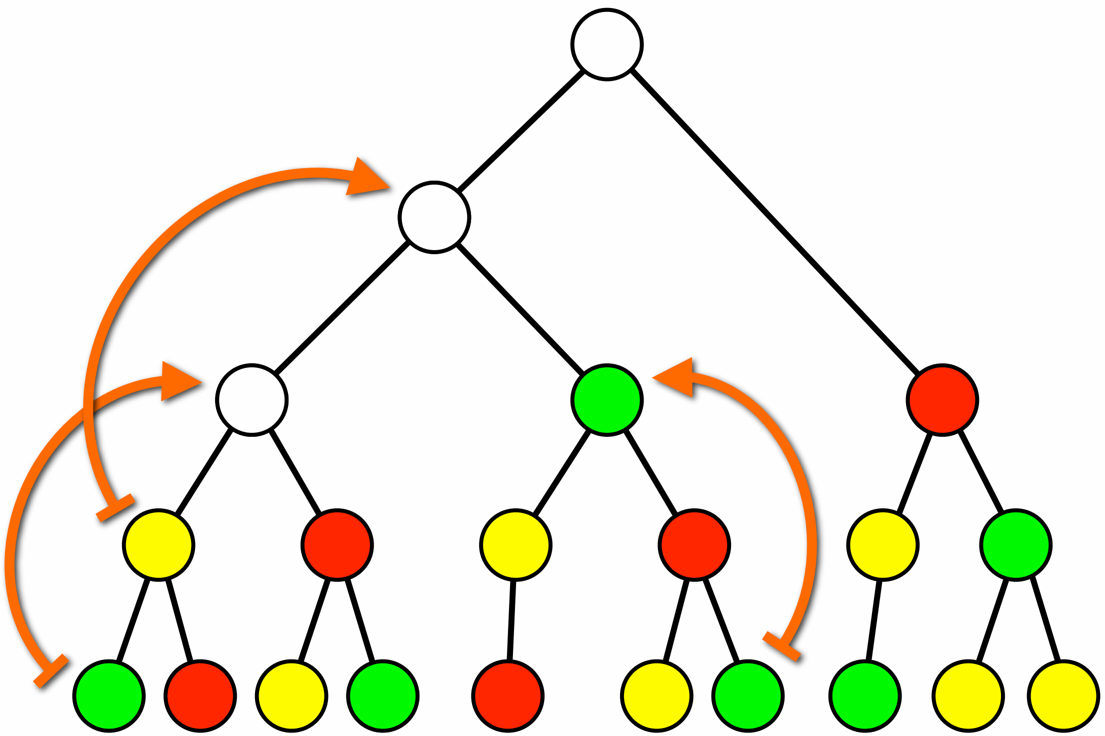
(an example of reduction to kernel size)



Rule 1: Decoloring of ancestors of the same color

How to apply the rules?

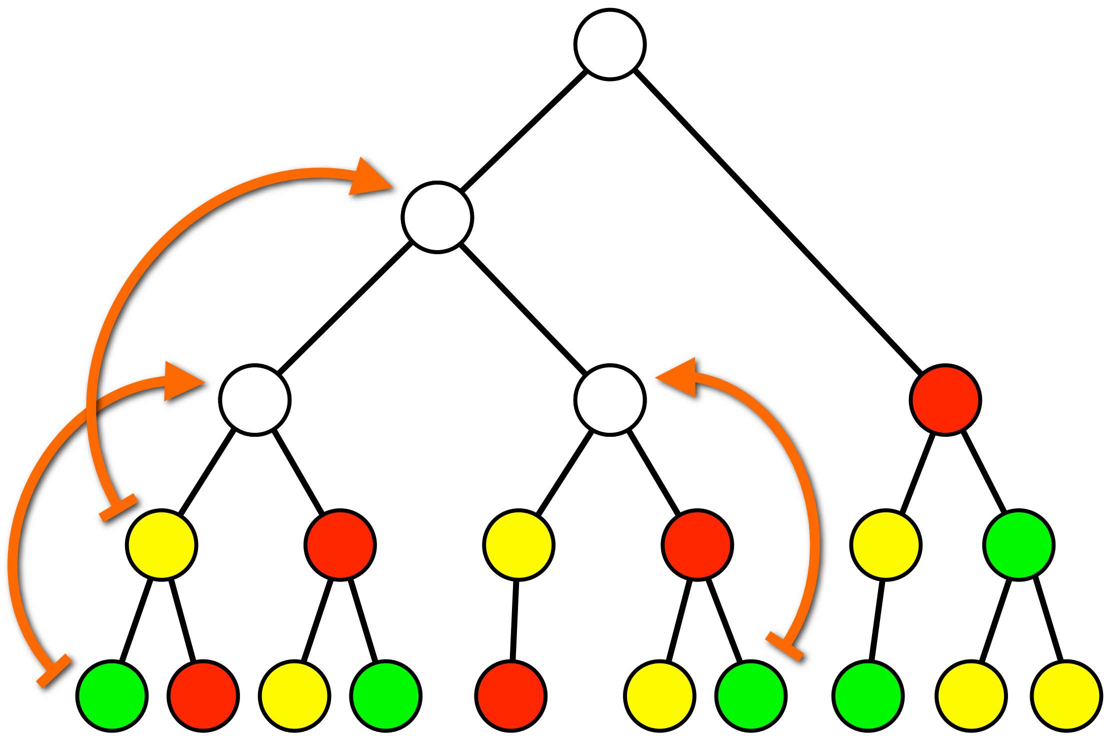
(an example of reduction to kernel size)



Rule 1: Decoloring of ancestors of the same color

How to apply the rules?

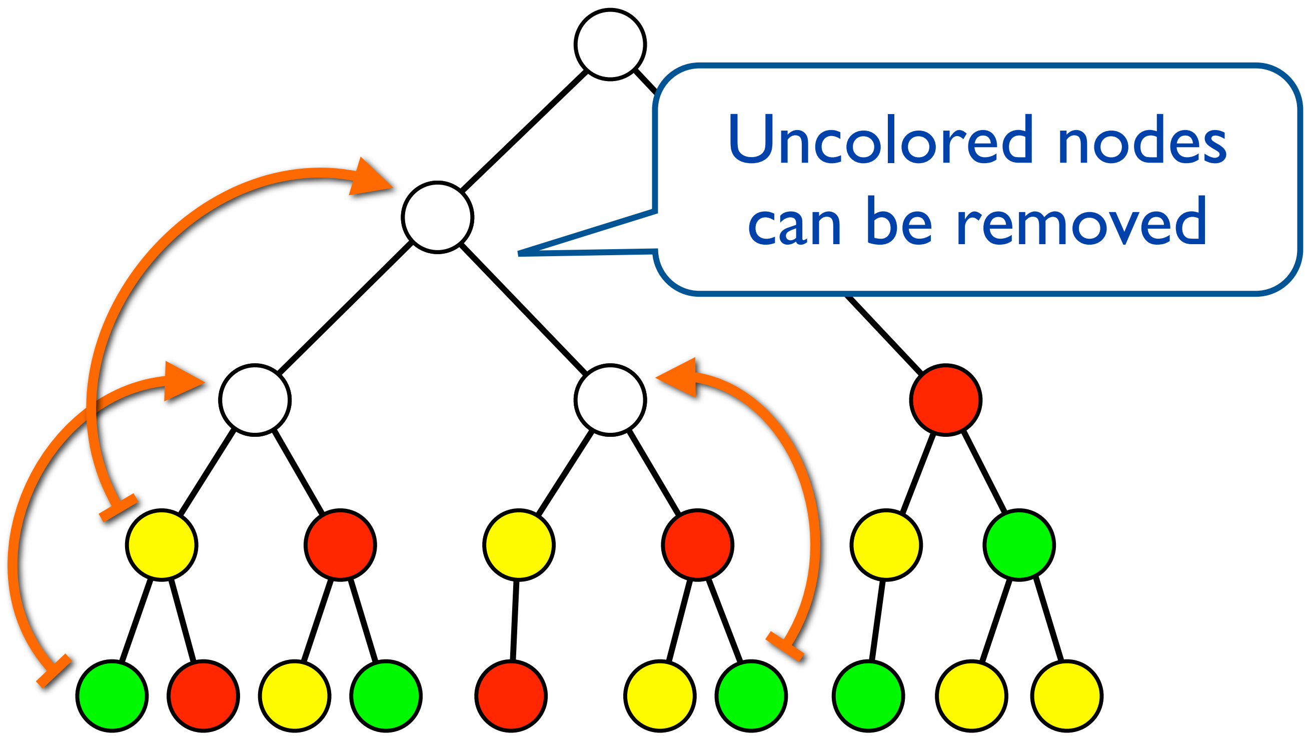
(an example of reduction to kernel size)



Rule 1: Decoloring of ancestors of the same color

How to apply the rules?

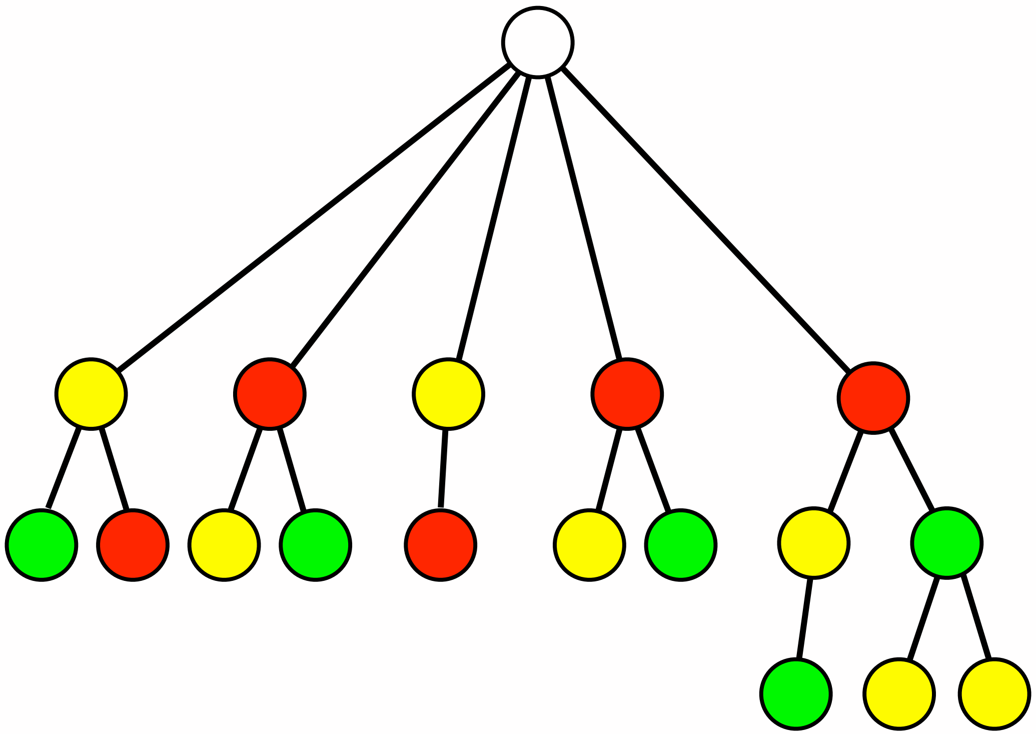
(an example of reduction to kernel size)



Rule 1: Decoloring of ancestors of the same color

How to apply the rules?

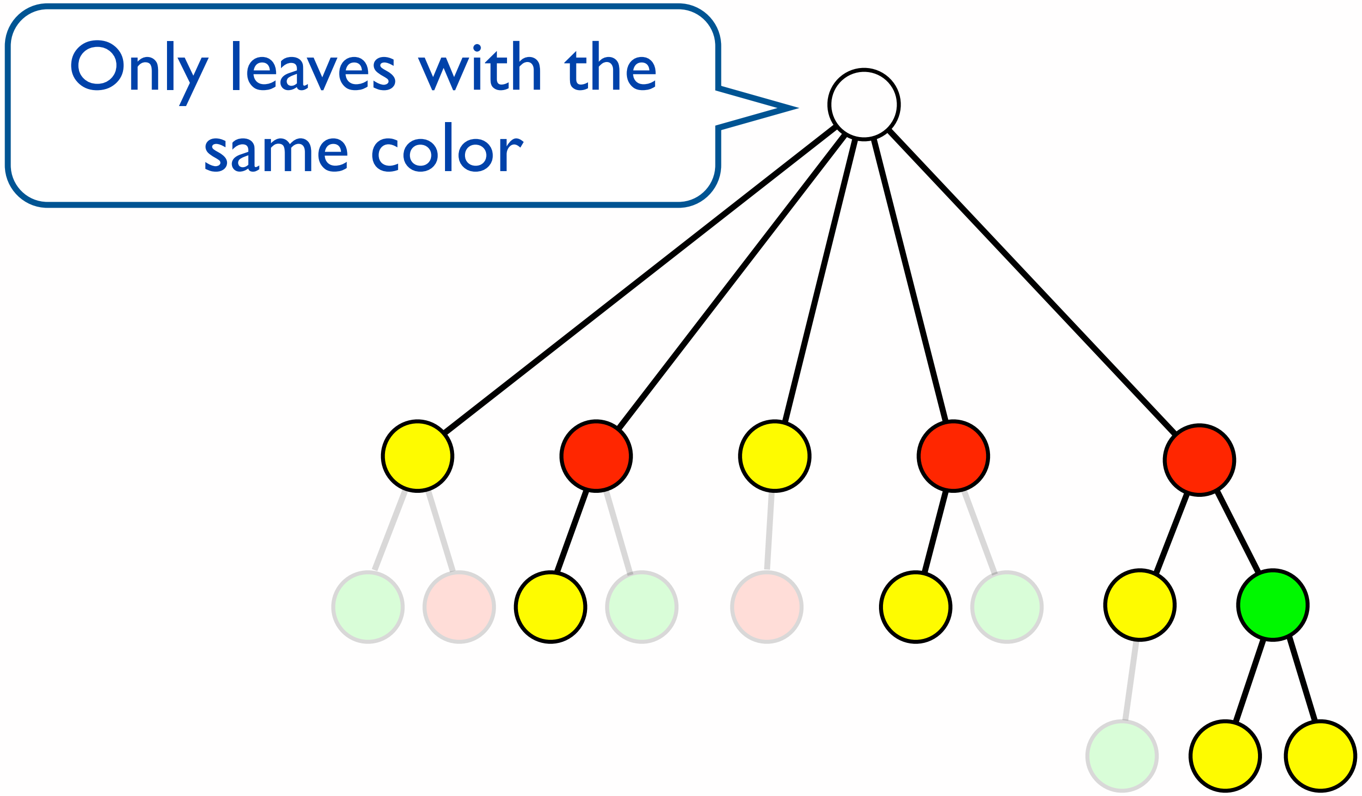
(an example of reduction to kernel size)



Rule 2: Decoloring of nodes with fan-out ≥ |P|

How to apply the rules?

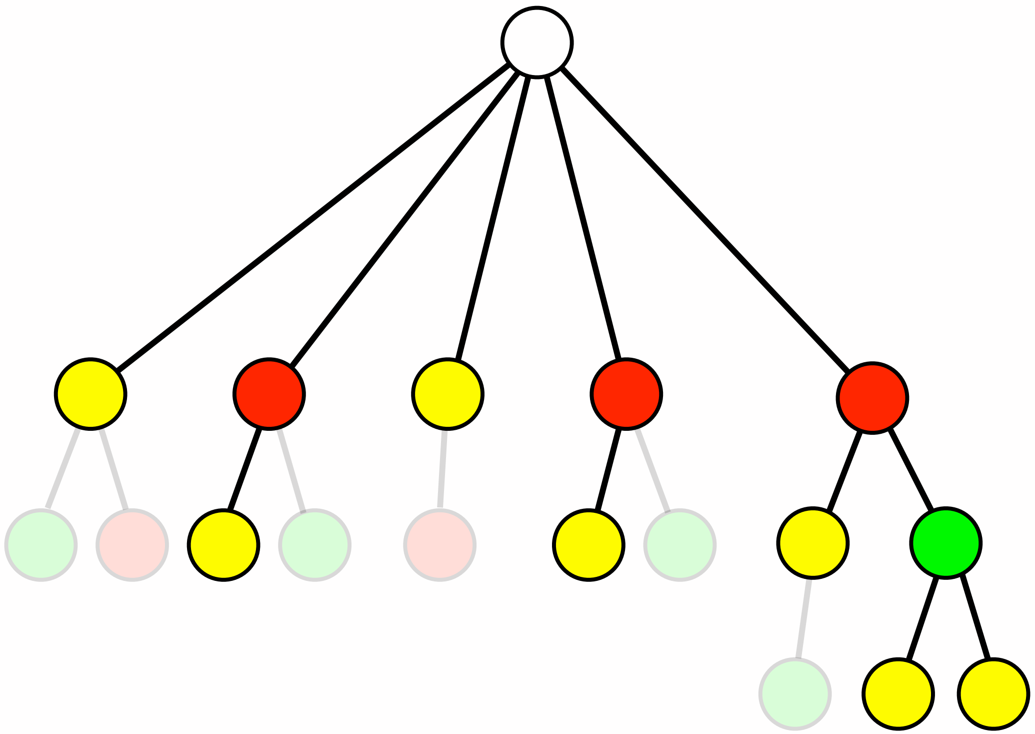
(an example of reduction to kernel size)



Rule 2: Decoloring of nodes with fan-out ≥ |P|

How to apply the rules?

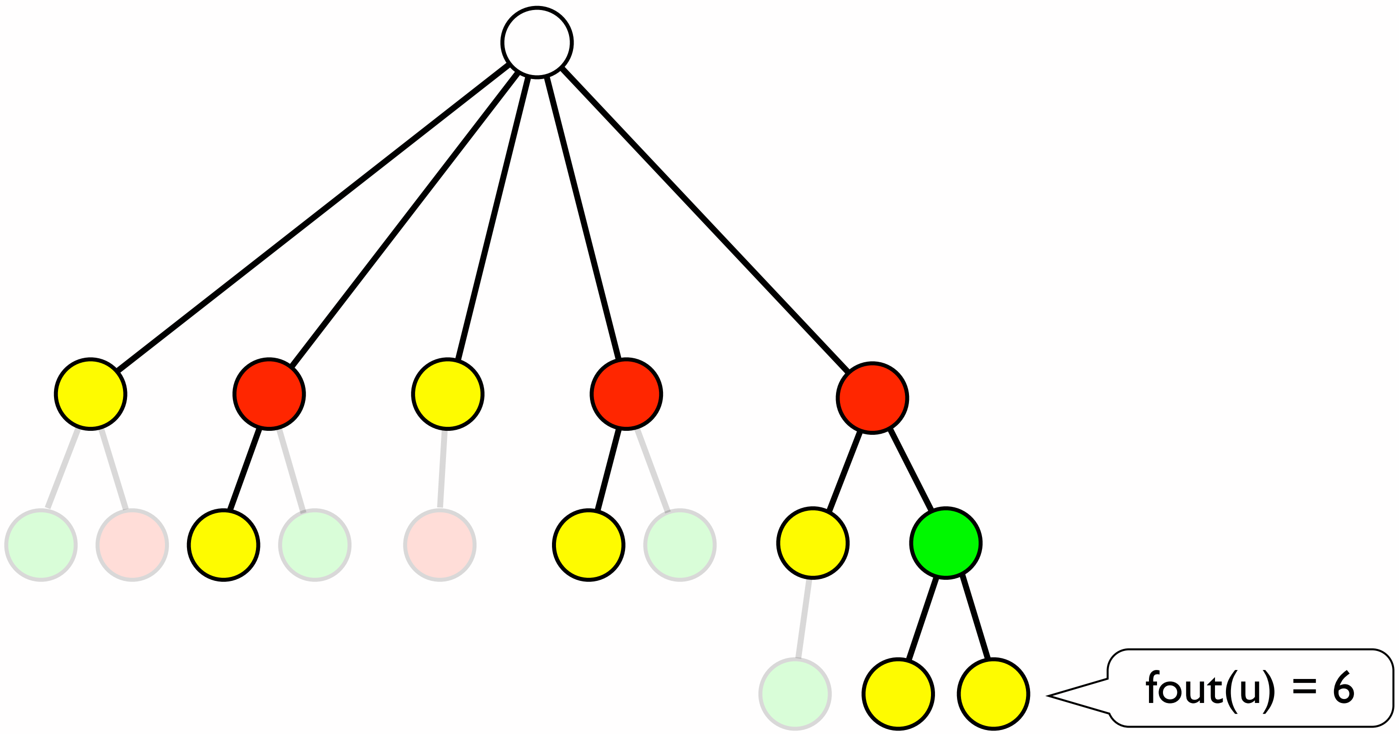
(an example of reduction to kernel size)



Rule 2: Decoloring of nodes with fan-out ≥ |P|

How to apply the rules?

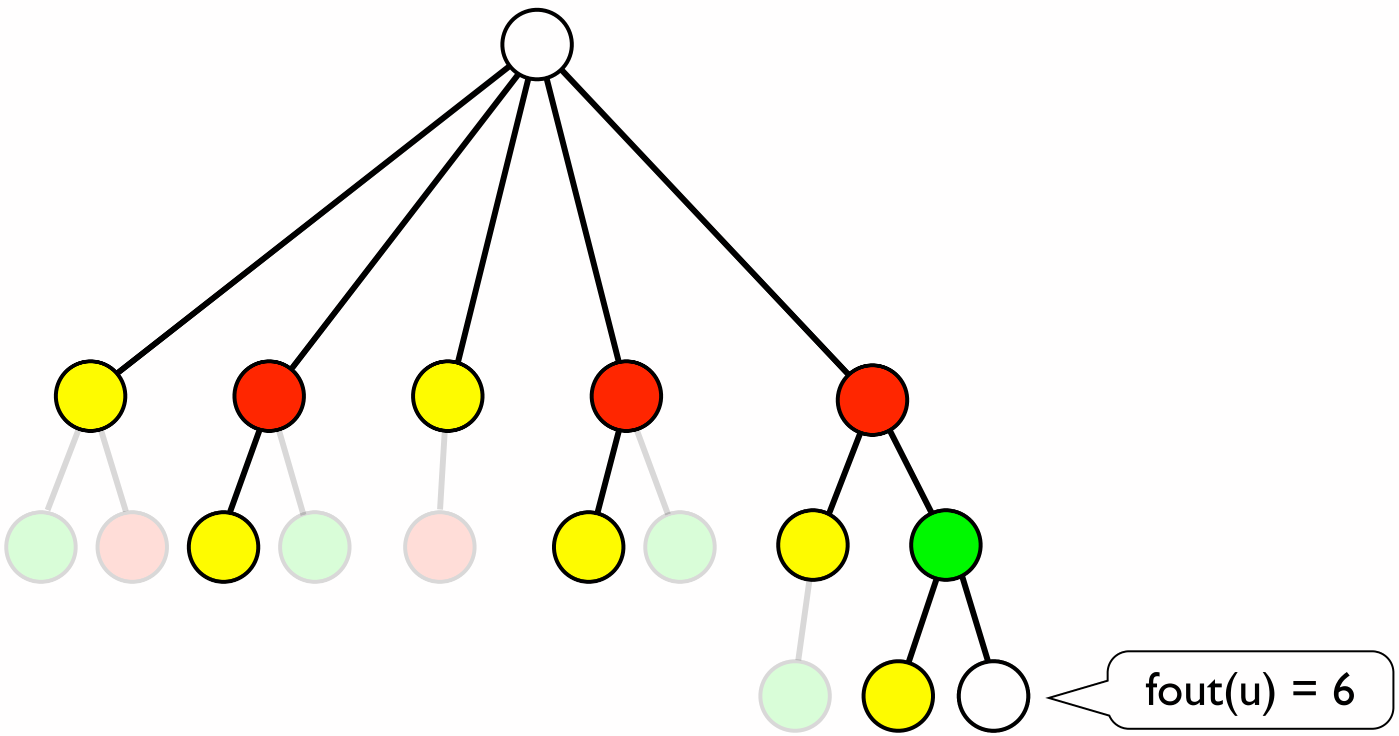
(an example of reduction to kernel size)



Rule 2: Decoloring of nodes with fan-out ≥ |P|

How to apply the rules?

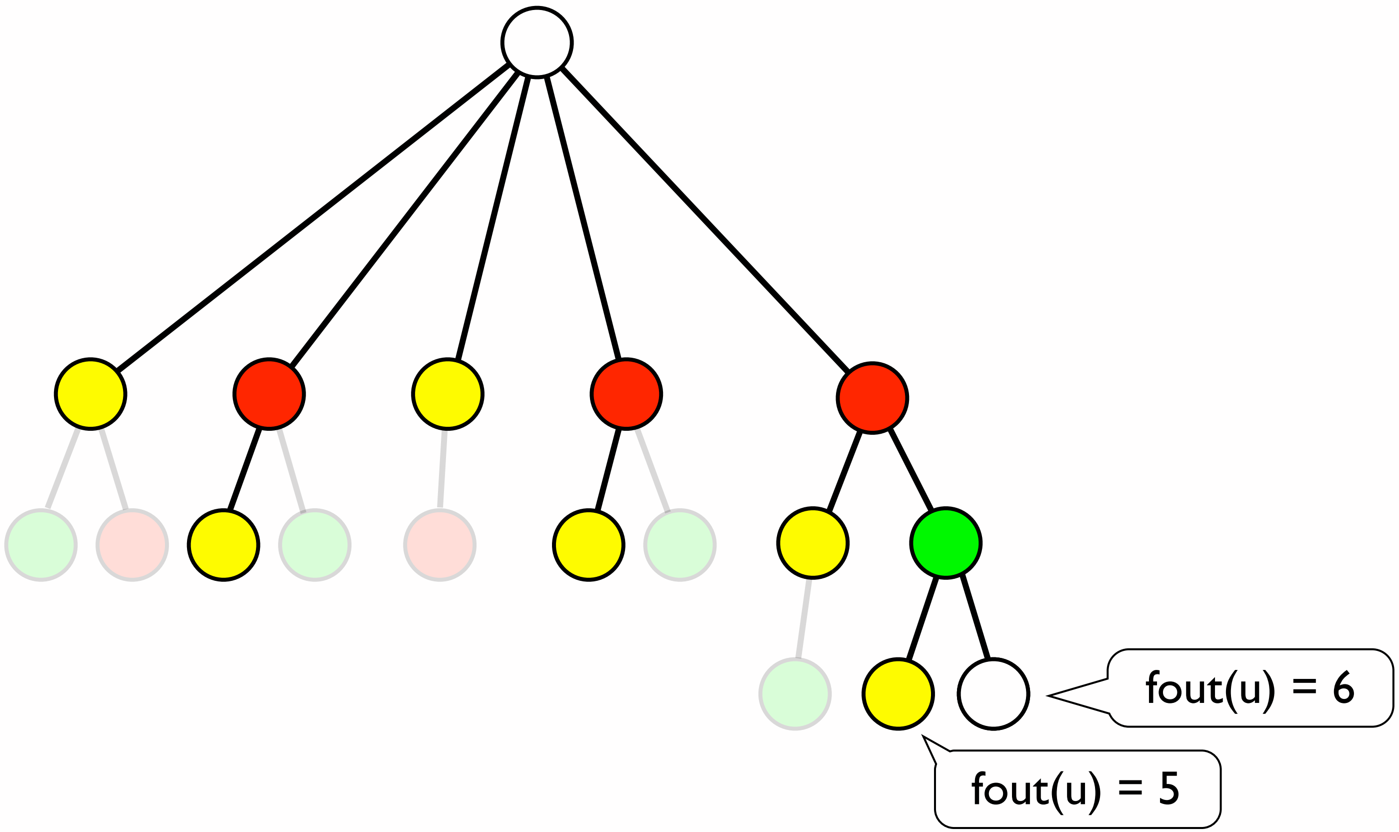
(an example of reduction to kernel size)



Rule 2: Decoloring of nodes with fan-out ≥ |P|

How to apply the rules?

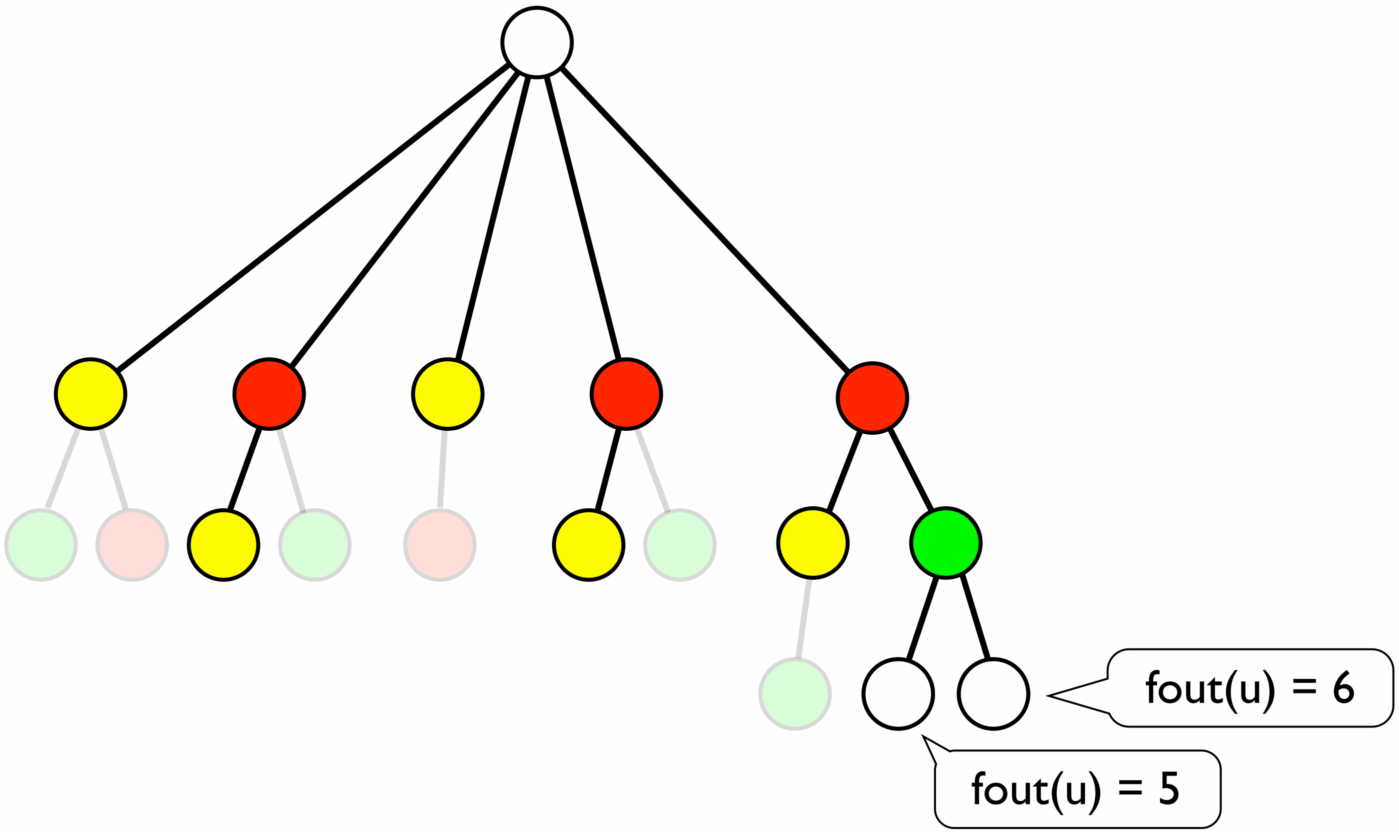
(an example of reduction to kernel size)



Rule 2: Decoloring of nodes with fan-out ≥ |P|

How to apply the rules?

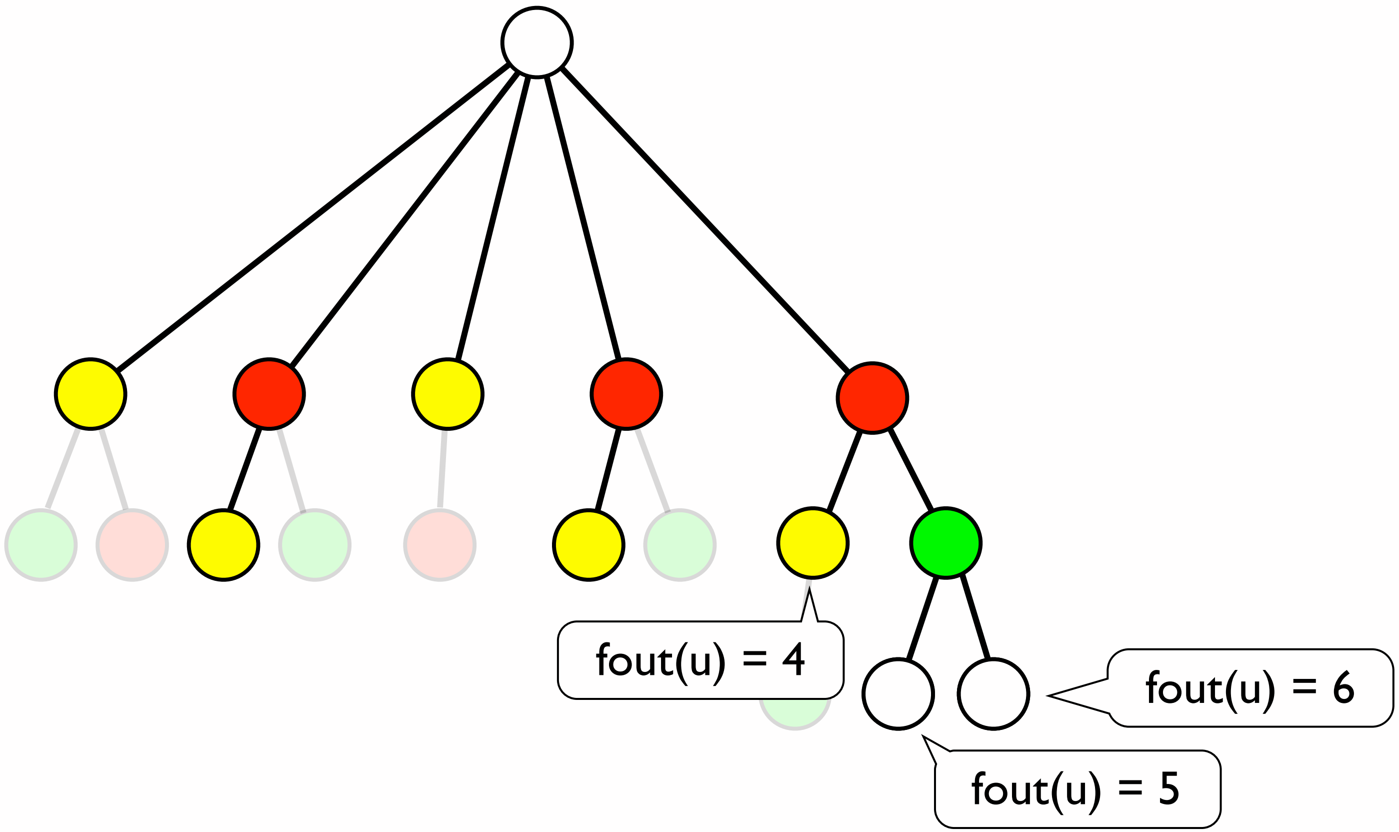
(an example of reduction to kernel size)



Rule 2: Decoloring of nodes with fan-out ≥ |P|

How to apply the rules?

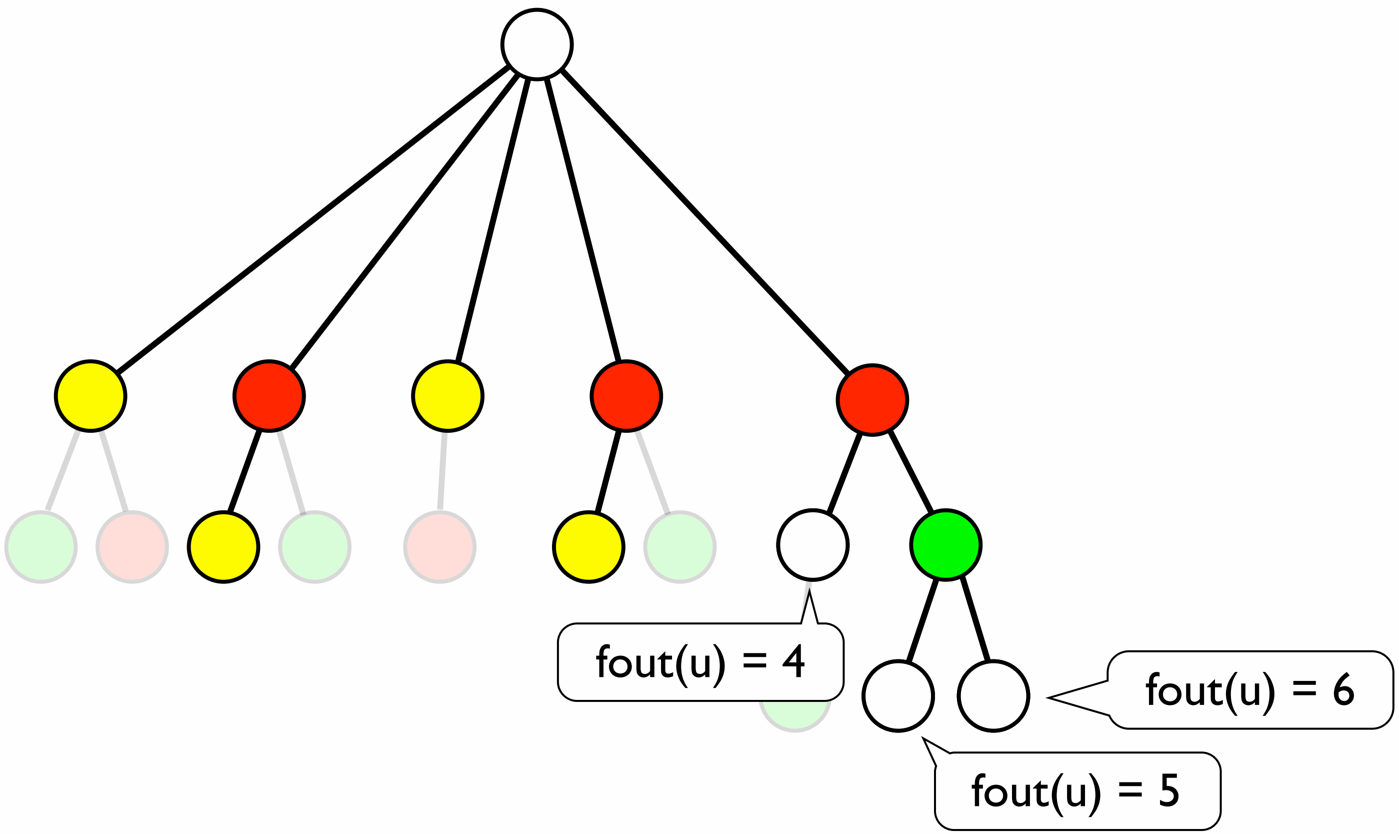
(an example of reduction to kernel size)



Rule 2: Decoloring of nodes with fan-out ≥ |P|

How to apply the rules?

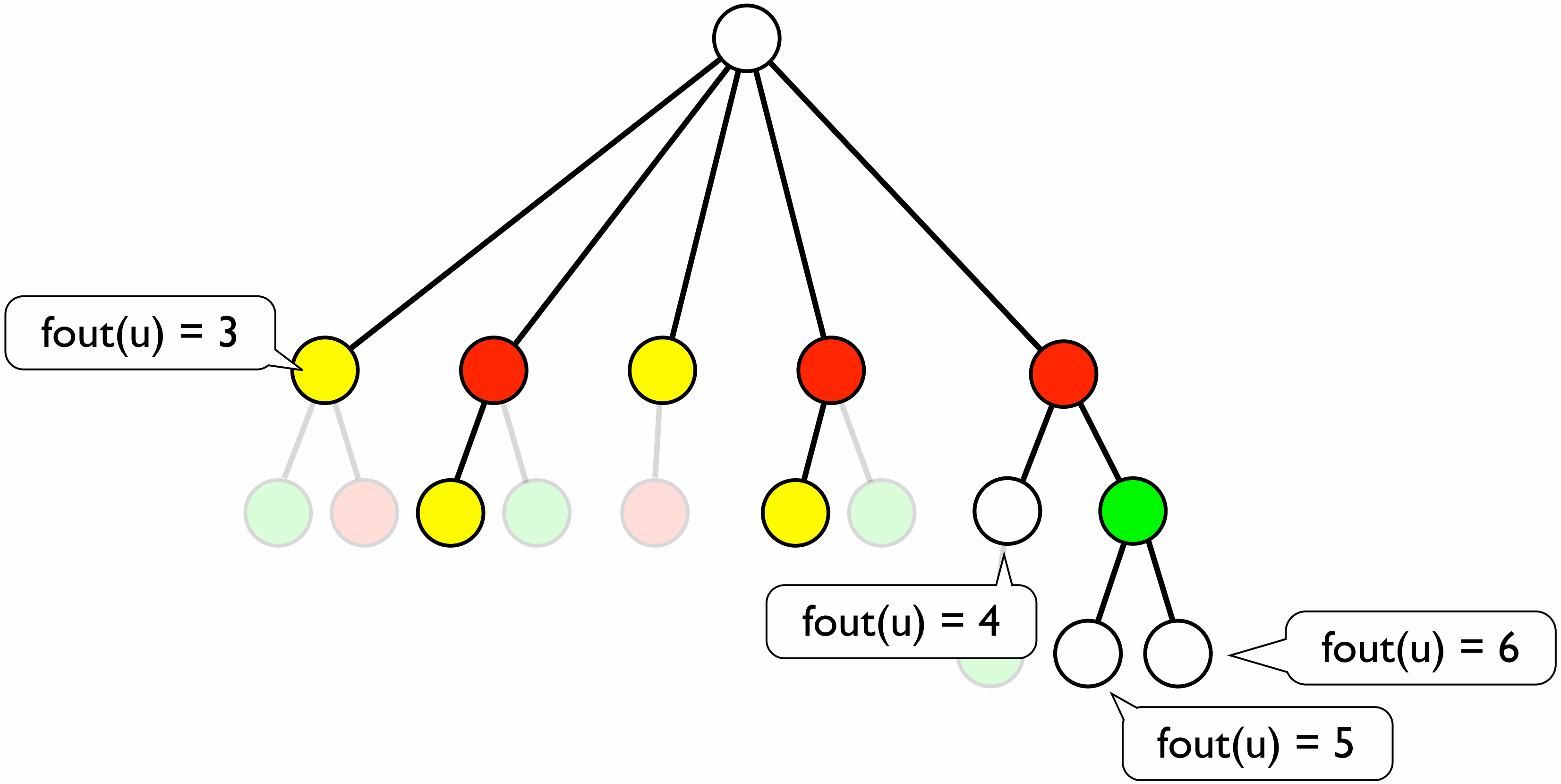
(an example of reduction to kernel size)



Rule 2: Decoloring of nodes with fan-out $\geq |P|$

How to apply the rules?

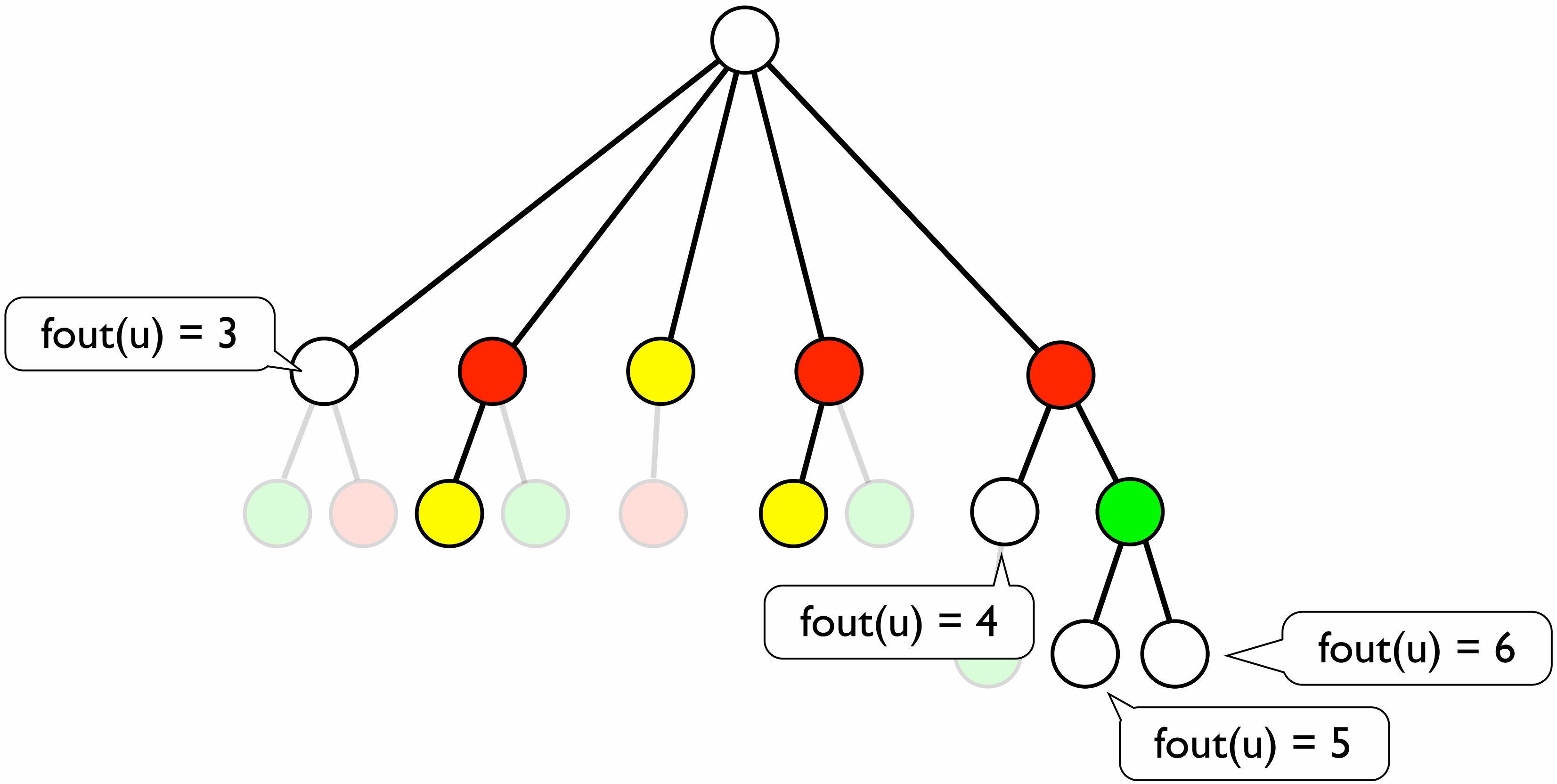
(an example of reduction to kernel size)



Rule 2: Decoloring of nodes with fan-out ≥ |P|

How to apply the rules?

(an example of reduction to kernel size)

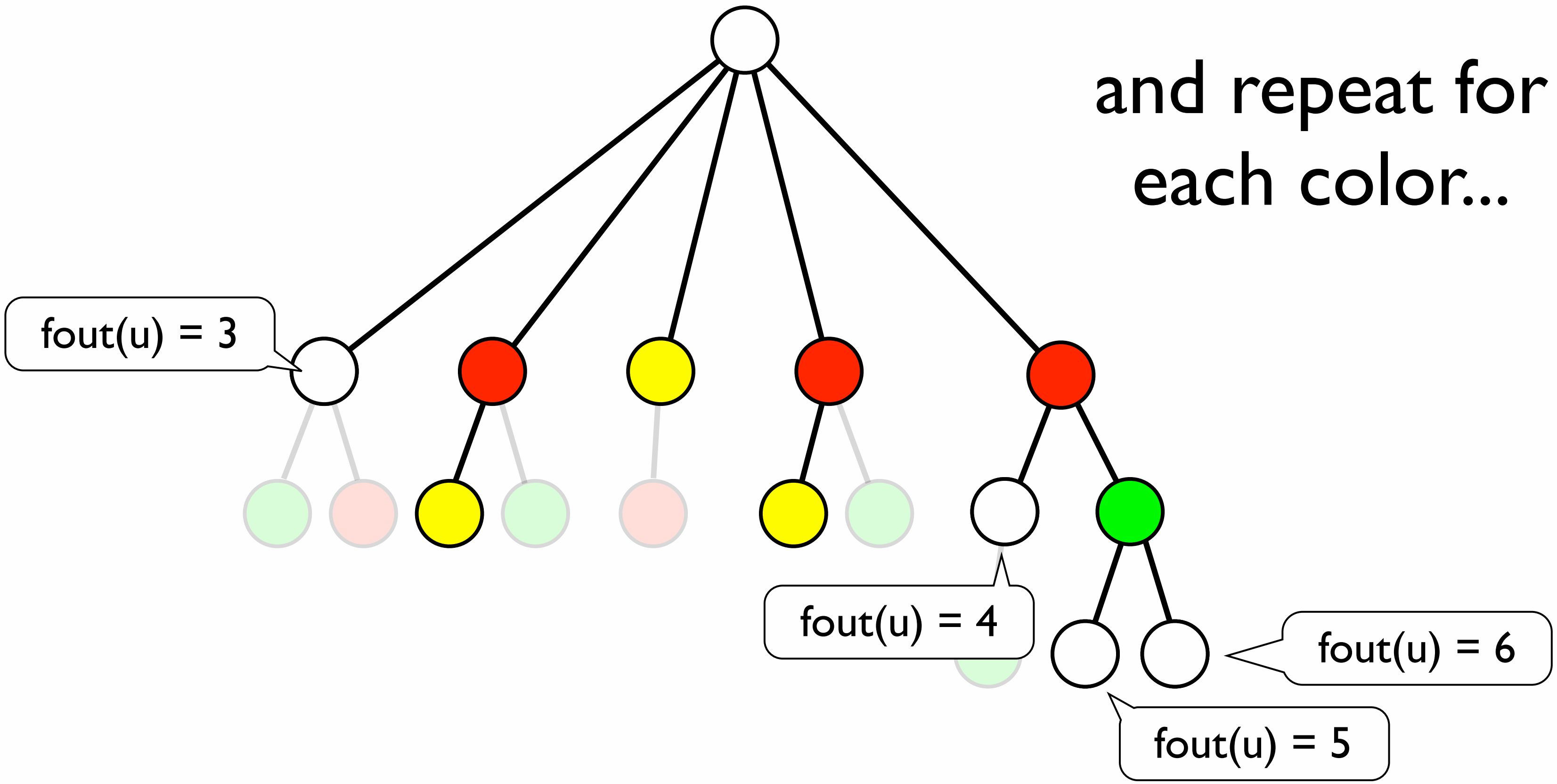


Rule 2: Decoloring of nodes with fan-out ≥ |P|

How to apply the rules?

(an example of reduction to kernel size)

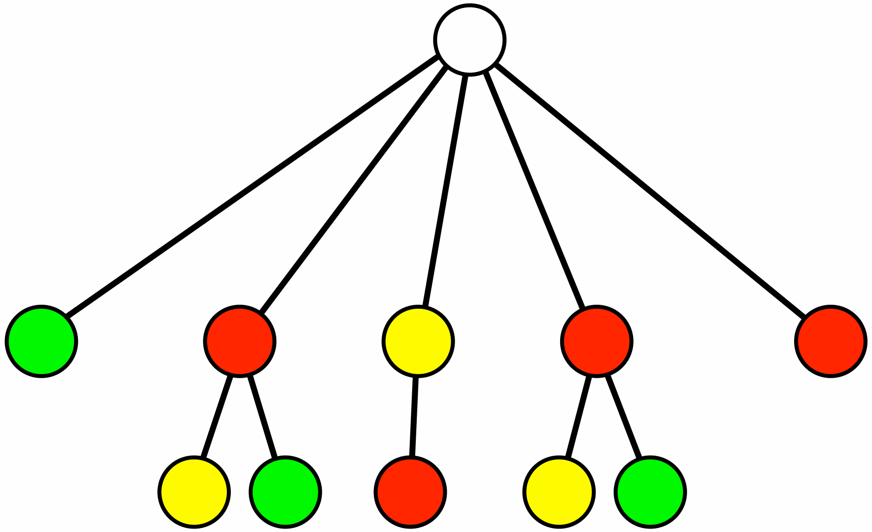
and repeat for each color...



Rule 2: Decoloring of nodes with fan-out ≥ |P|

How to apply the rules?

(an example of reduction to kernel size)



...to obtain the reduced tree

Reduction to Kernel-Size

- by repeating rule 1 ... $\text{height}(T) \leq |P|$
- by repeating rule 2 ... $|\text{c-color}(T)| \leq 2^{|P|}$

Reduction to Kernel-Size

No repetitions of colors
in a path from the root to a leaf

- by repeating rule 1 ... $\mathrm{height}(T) \leq |P|$
- by repeating rule 2 ... $|\text{c-color}(T)| \leq 2^{|P|}$

Reduction to Kernel-Size

No repetitions of colors
in a path from the root to a leaf

- by repeating rule 1 ... height(T) $\leq$ |P|
- by repeating rule 2 ... |c-color(T)| $\leq 2^{|P|}$

If max{ fout(n) | n node in T } $\leq$ m, then T has at most $2^{|P|}$ leaves

Reduction to Kernel-Size

No repetitions of colors
in a path from the root to a leaf

- by repeating rule 1 ... $height(T) \leq |P|$
- by repeating rule 2 ... $|c\text{-}color(T)| \leq 2^{|P|}$

If max{ fout(n) | n node in T } $\leq$ m, then T has at most $2^{|P|}$ leaves

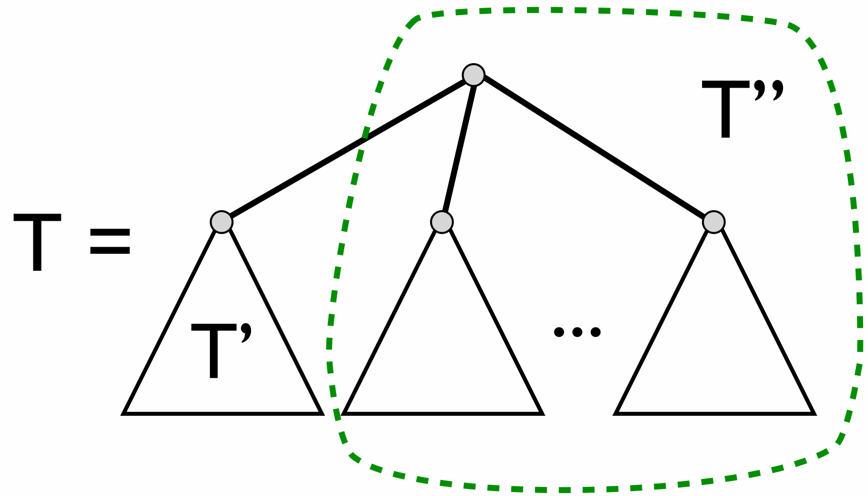
---

as a result $|nodes(T)| \leq |P|\, 2^{|P|}$

2. Searching for a rainbow

using the fast *subset convolution* algorithm by [Björklund et al. STOC'97]

$$A(T,X) = N(T,X) \vee \bigvee_{Y \subseteq X} \big(A(T',Y) \wedge A(T'',Y \setminus X)\big)$$

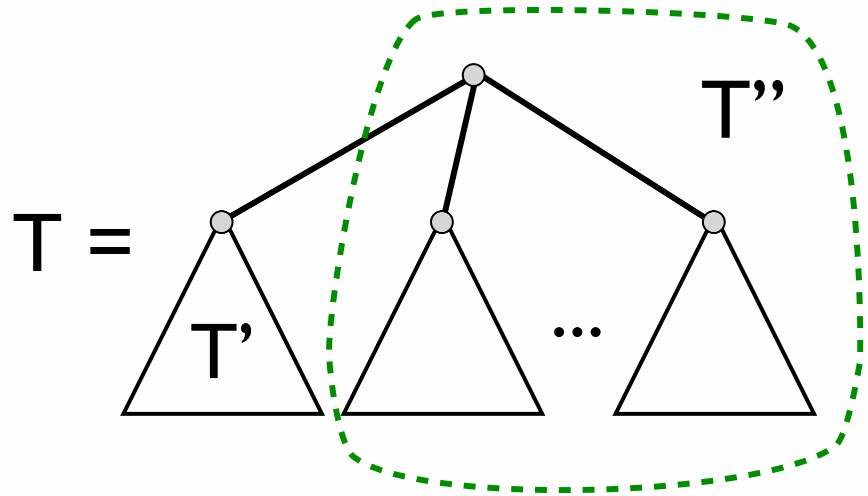


2. Searching for a rainbow

using the fast *subset convolution* algorithm by
[Björklund et al. STOC'97]

$$A(T, X) = N(T, X) \vee \bigvee_{Y \subseteq X} \Big(A(T', Y) \wedge A(T'', Y \setminus X) \Big)$$

true iff T contains all the colors in $X \subseteq P$

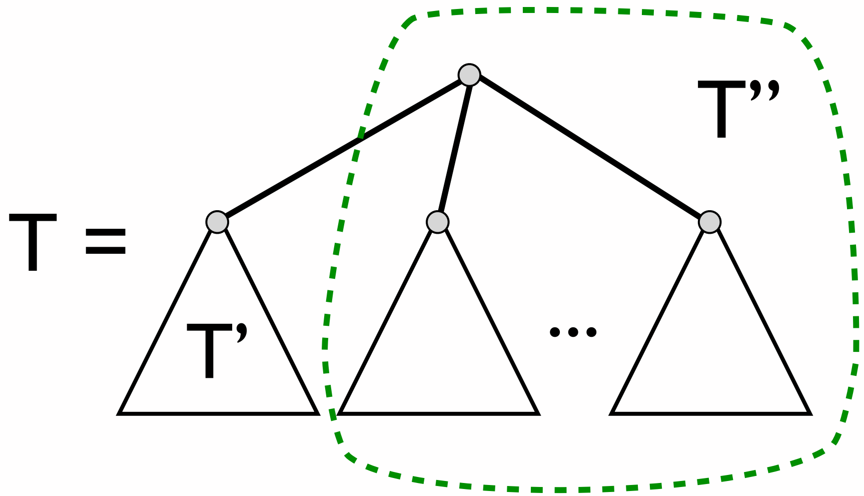


2. Searching for a rainbow

using the fast *subset convolution* algorithm by
[Björklund et al. STOC'97]

$$A(T, X) = N(T, X) \vee \bigvee_{Y \subseteq X} \big(A(T',Y) \wedge A(T'',Y \setminus X)\big)$$

true iff T contains all the colors in $X \subseteq P$



for each node: $O(|P|^2\, 2^{|P|})$

From Forest Matching to Rainbow Antichain

Target T

n[0] | m[x | n[0]] | k[n[y]] | m[0] | z

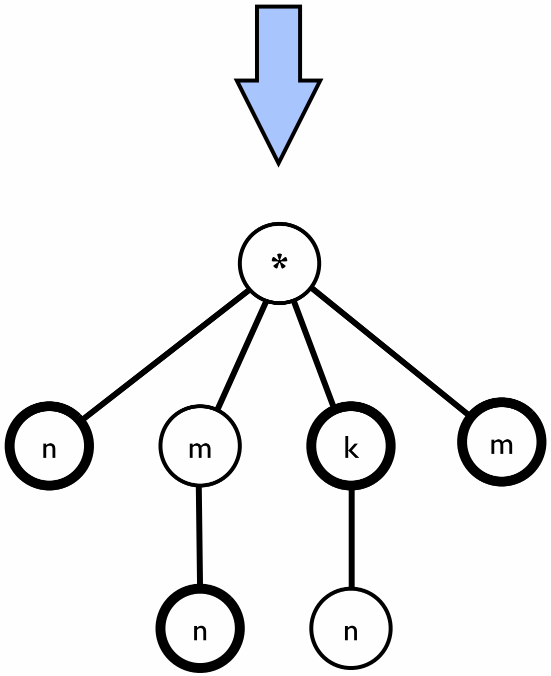
Forest pattern S

⟨m[x] | n[0] , m[0]⟩

From Forest Matching to Rainbow Antichain

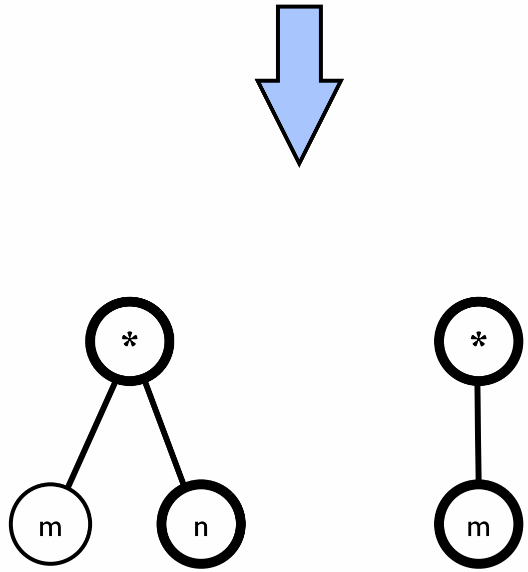
Target T

n[0] | m[x | n[0]] | k[n[y]] | m[0] | z



Forest pattern S

⟨m[x] | n[0] , m[0]⟩



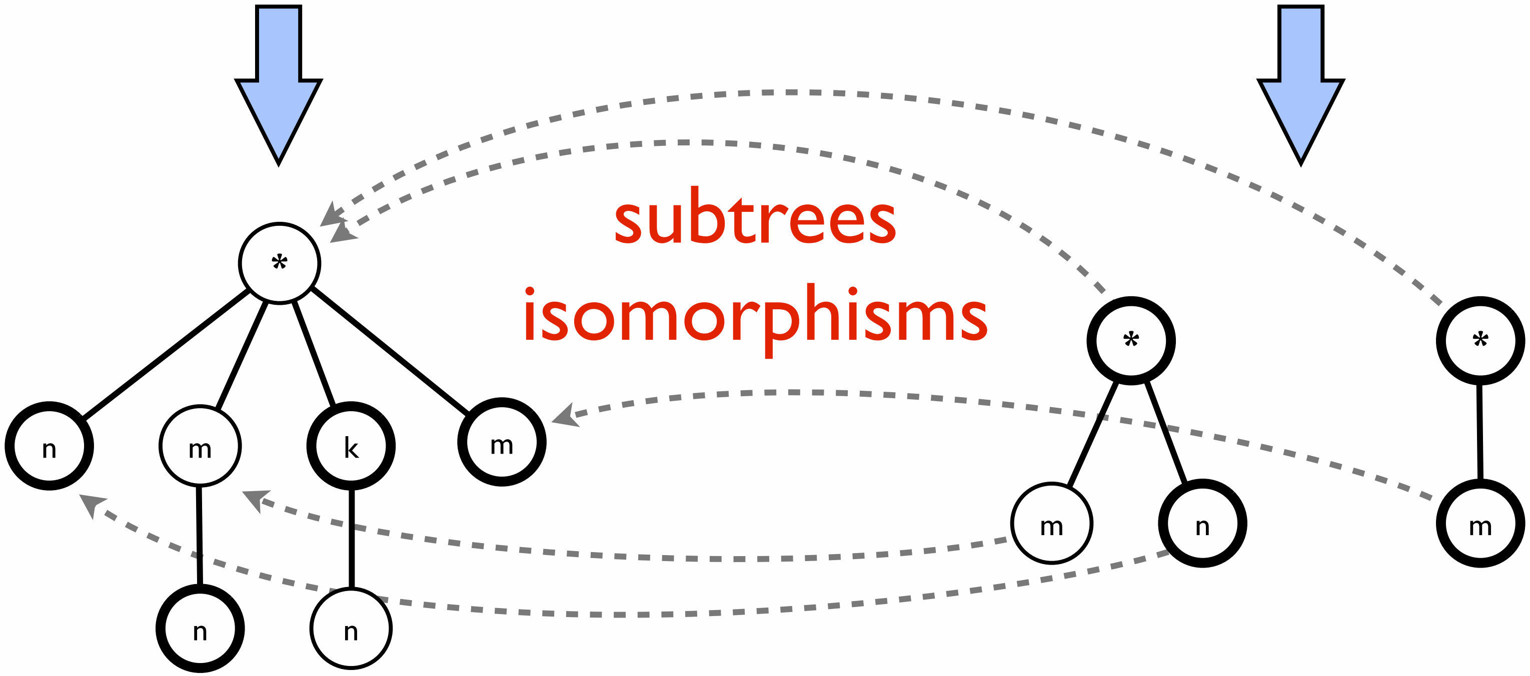
From Forest Matching to Rainbow Antichain

Target T

n[0] | m[x | n[0]] | k[n[y]] | m[0] | z

Forest pattern S

⟨m[x] | n[0] , m[0]⟩



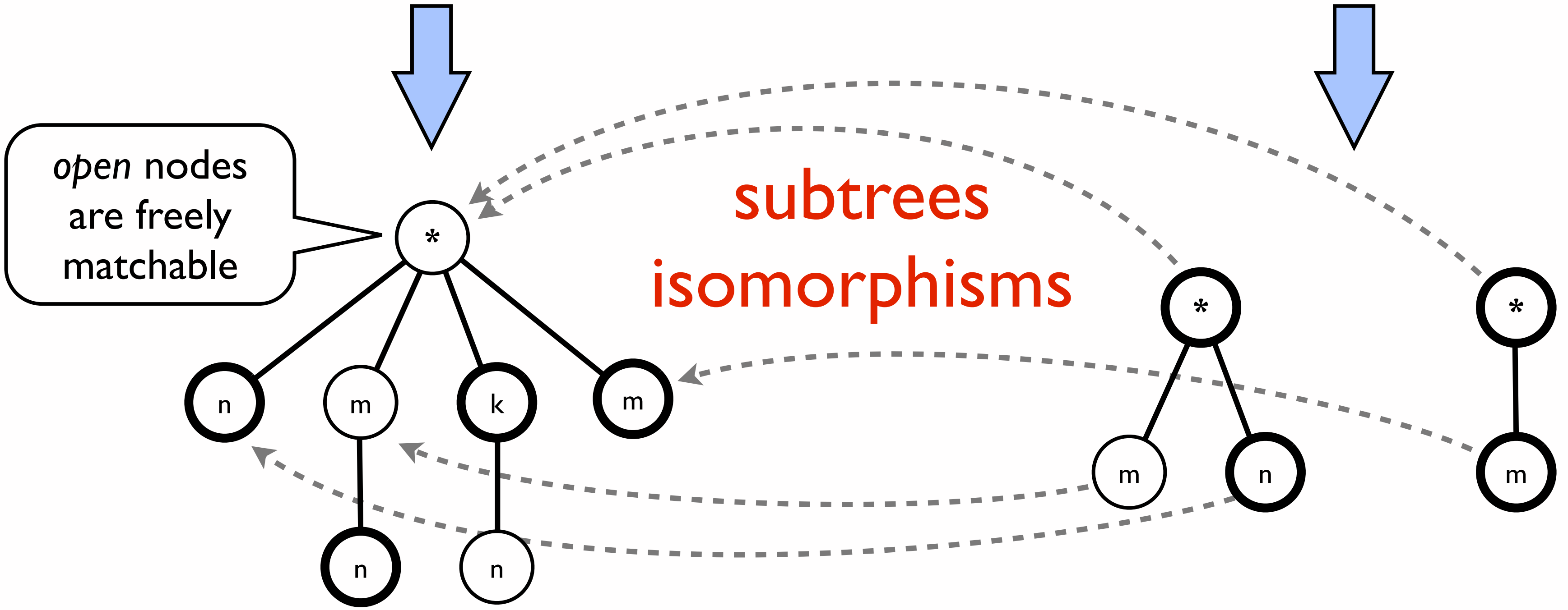
From Forest Matching to Rainbow Antichain

Target T

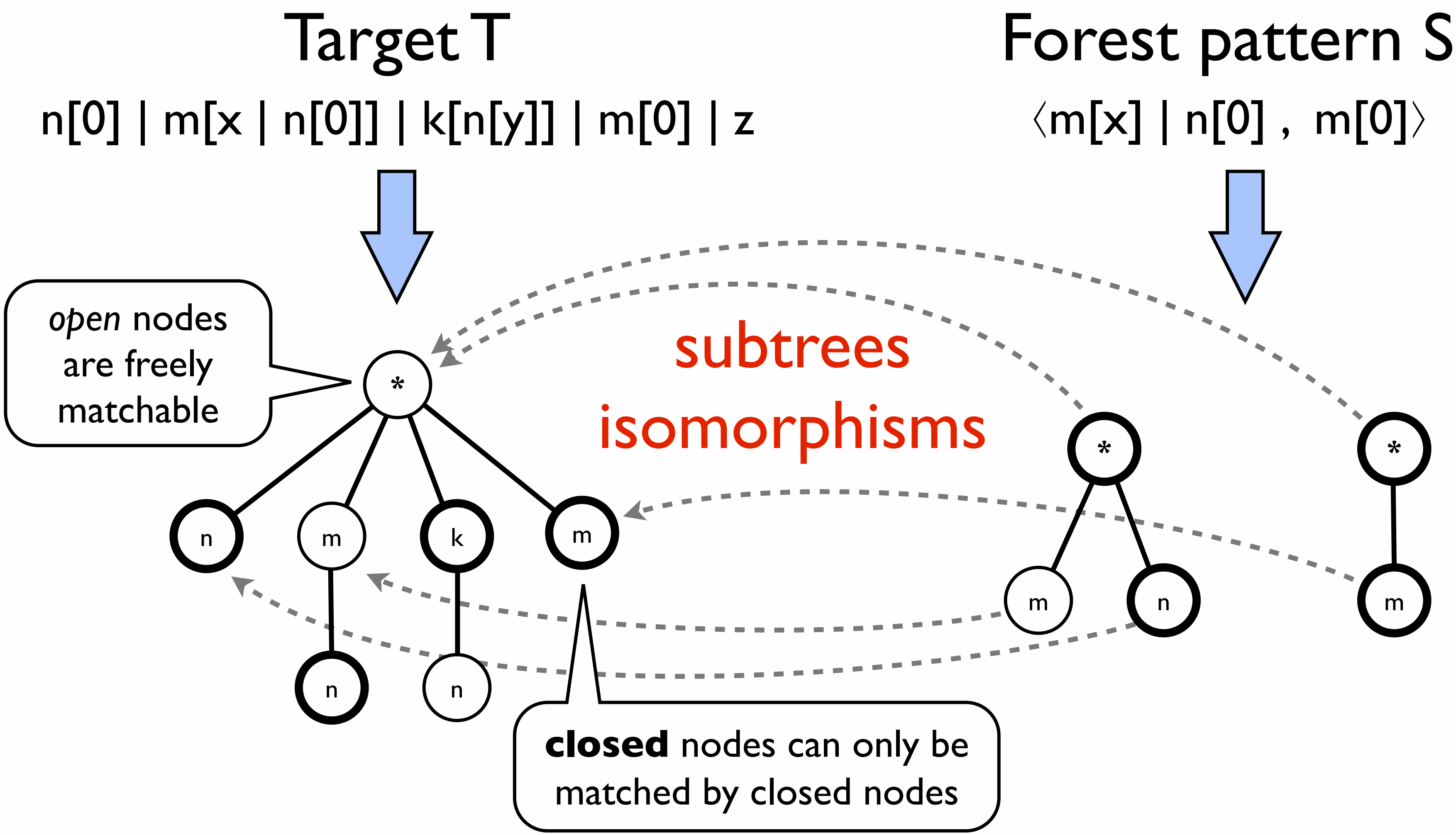
n[0] | m[x | n[0]] | k[n[y]] | m[0] | z

Forest pattern S

⟨m[x] | n[0] , m[0]⟩



From Forest Matching to Rainbow Antichain



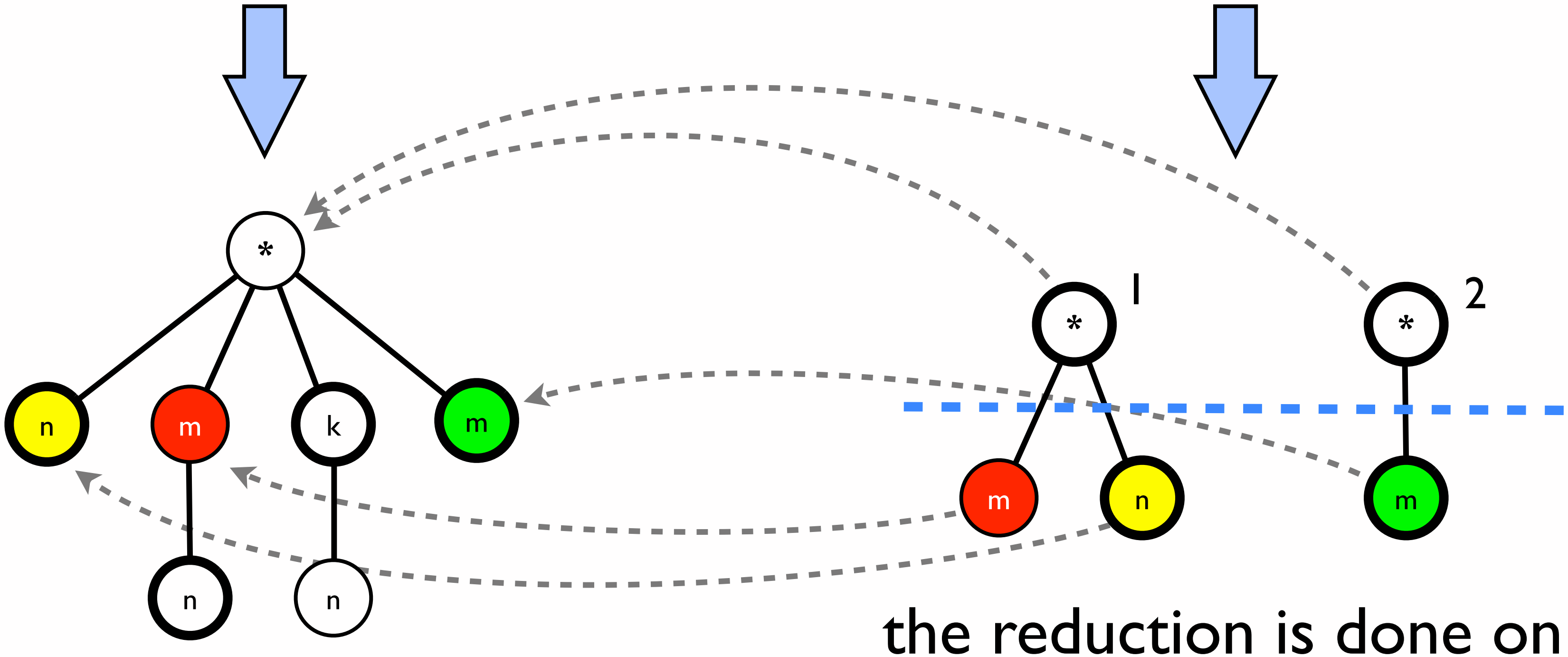
From Forest Matching to Rainbow Antichain

Target T

n[0] | m[x | n[0]] | k[n[y]] | m[0] | z

Forest pattern S

⟨m[x] | n[0] , m[0]⟩



the reduction is done on a 2-level palette

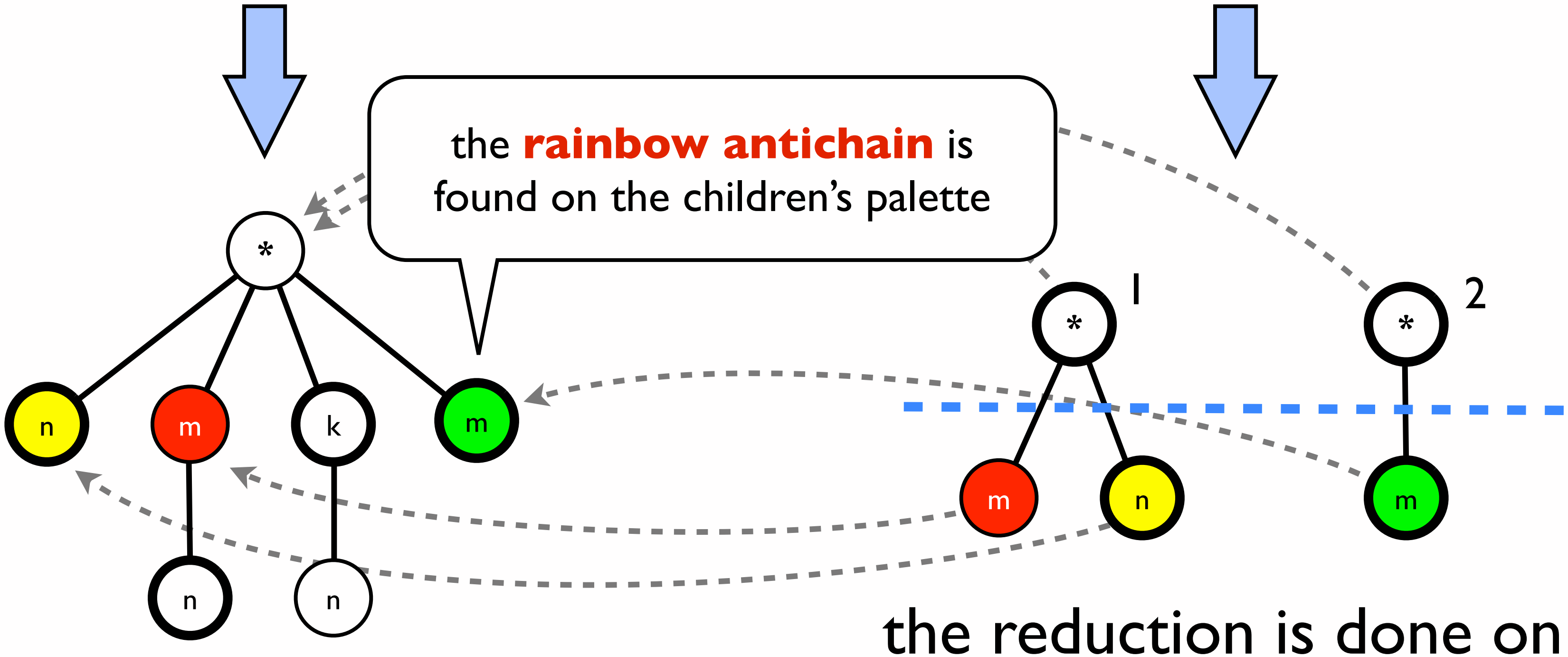
From Forest Matching to Rainbow Antichain

Target T

n[0] | m[x | n[0]] | k[n[y]] | m[0] | z

Forest pattern S

⟨m[x] | n[0] , m[0]⟩



the reduction is done on a 2-level palette

Forest matching is Fixed Parameter Tractable

Parameters

h = "# trees in pattern S"
k = "max # of 1st-level children in S"

- subtree isomorphisms: $O(|S|\ |T|^{3/2})$
- reduction to kernel size: $O(|T|)$
- exhaustive search of antichains: $O(h^3\ k\ 2^{2h})$

Forest matching is Fixed Parameter Tractable

Parameters

h = "# trees in pattern S"

k = "max # of 1st-level children in S"

- subtree isomorphisms: $O(|S|\ |T|^{3/2})$
- reduction to kernel size: $O(|T|)$
- exhaustive search of antichains: $O(h^3\ k\ 2^{2h})$

Total: $O(h^3\ k\ 2^{2h}) + O(|S|\ |T|^{3/2})$

Conclusions

- WRSs yield simpler and smaller semantics
- We have shown that matching for WRSs is feasible:
  - exponential in redex width (constant and small)
  - but polynomial in the size of agent
- Side result: *rainbow antichain problem*
- Applications: abstract machines (distributed π, Ambients, CaSPiS, etc.) and *bigraphic reactive systems*
- Future work: quantitative variants (probabilistic, etc.)

Thanks for your attention