The restriction method in circuit and proof complexity

Paul Beame

University of Washington

PCMI 2000 Friday July 28

Circuit lower bound for parity

- Theorem [Hastad] The n-bit parity function $\mathbf{x_1}$Å$\mathbf{x_2}$Å... Å$\mathbf{x_n}$ cannot be computed by unbounded fan-in circuits in size $\mathbf{S}$ and depth d unless $$\mathbf{S} \geq \mathbf{2^{cn^{1/d}}}$$

- Corollary: Polynomial-size circuits for parity require W(**log n/loglog n**) depth
  - **Parity** Ï **AC^0**

- Original proof used **restriction** argument

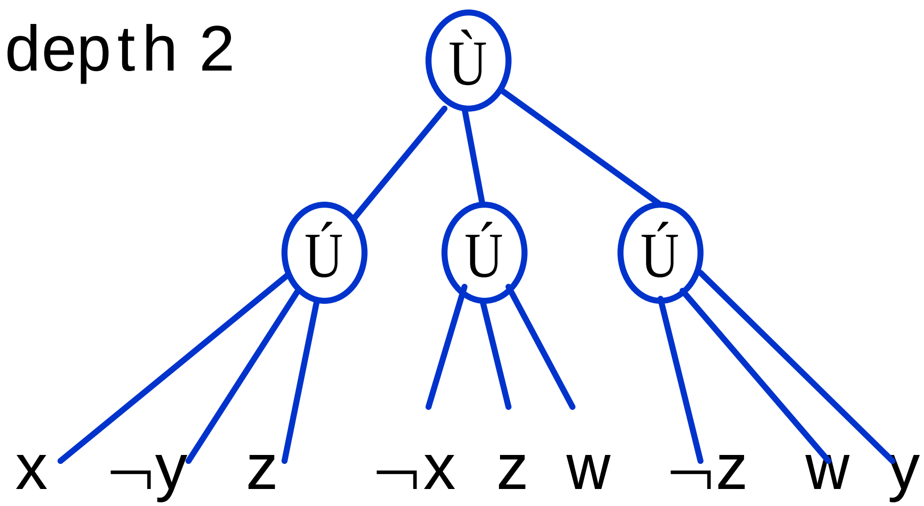
Restrictions

- **Defn:** Given a set $\mathbf{X}$ of Boolean variables, a restriction ρ is a partial assignment of values to the variables of $\mathbf{X}$
  - formally $\rho : \mathbf{X} \to \{\mathbf{0},\mathbf{1},*\}$ where $\rho(\mathbf{x_i}) = *$ indicates that the variable $\mathbf{x_i}$ is not assigned a value

- If F is a function, formula, or circuit, write $F|_\rho$ for the result of substituting $\rho(x_i)$ for each x_i s.t. $\rho(x_i) \neq *$

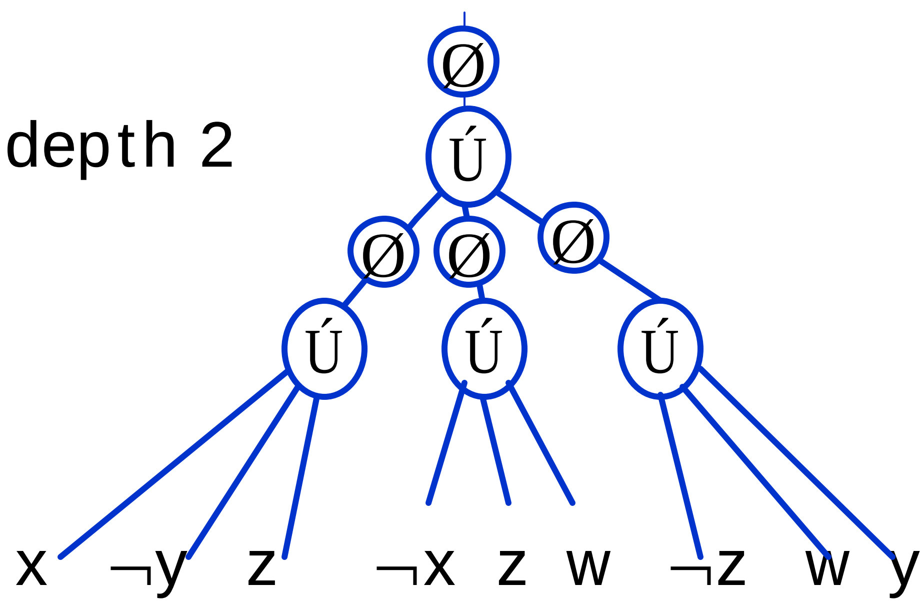
Unbounded fan-in circuits

- Restrict to connectives Ú, Ø
  - results for other connective is easily defined
- Defn: The **depth** of a formula **F** (circuit C) is max # of Ú on any path from an input to an output
- e.g. CNF/DNF have depth 2



Unbounded fan-in circuits

- Restrict to connectives Ú, Ø
  - results for other connective is easily defined
- Defn: The **depth** of a formula F (circuit C) is max # of Ú on any path from an input to an output
- e.g. CNF/DNF have depth 2



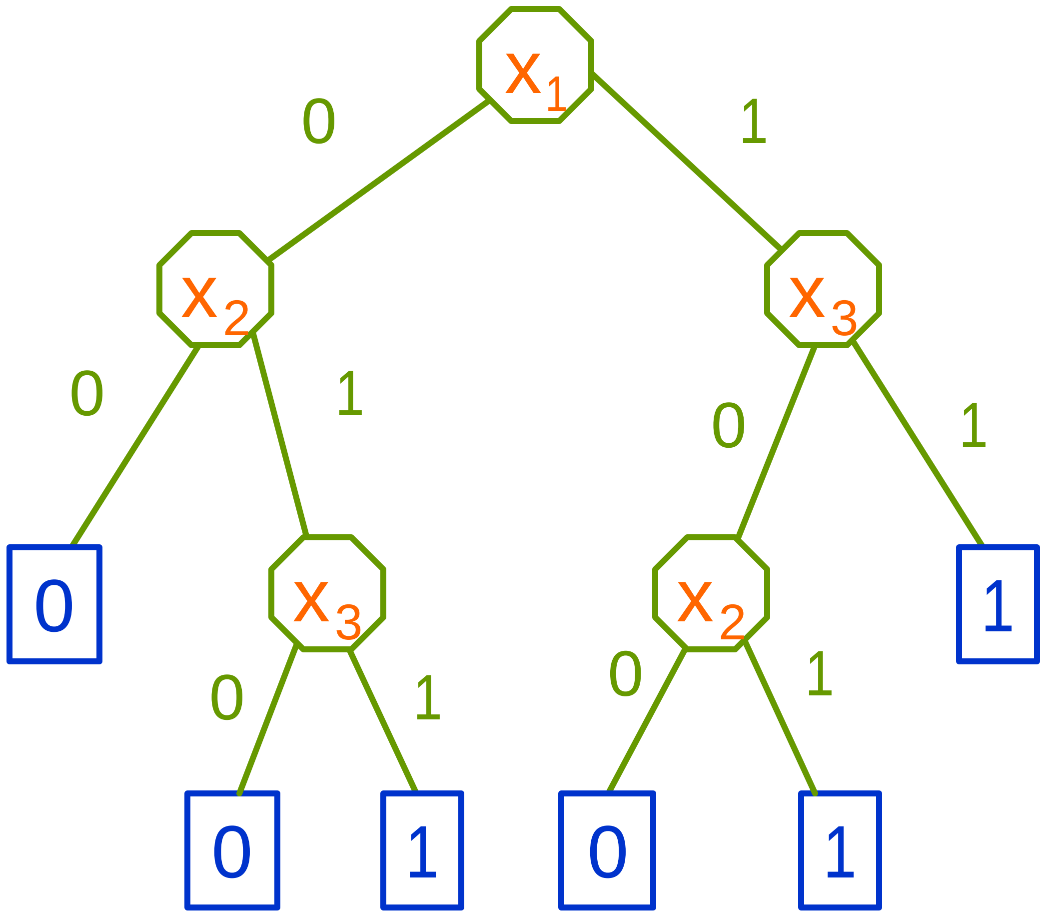
Why restrictions might be useful for circuit lower bounds

- Restrictions simplify functions, circuits, formulas
  - Given $\mathbf{F=(\bigvee_i x_i \wedge \bigvee_j \neg x_j)}$
    - assigning a single $\rho(\mathbf{x_i})=\mathbf{1}$ or a $\rho(\mathbf{x_j})=\mathbf{0}$ makes $\mathbf{F}|_\rho$ a constant; i.e. wiping out **F** but only setting one variable
  - Simplification is substantially more than # of variables assigned
- Basic idea: To prove that small circuit **C** cannot compute function **f**, choose a restriction ρ such that
  - $\mathbf{f}|_\rho$ is still complicated but
  - $\mathbf{C}|_\rho$ is extremely simple so that it obviously cannot compute $\mathbf{f}|_\rho$

Boolean decision trees

- **Defn:** A Boolean decision tree **T** is a binary rooted tree s.t.
  - each internal node is labelled by some $\mathbf{x_i}$
  - leaf nodes are labelled 0 or 1
  - edges out of each internal node are labelled 0 or 1
  - no two nodes on a path have the same variable label

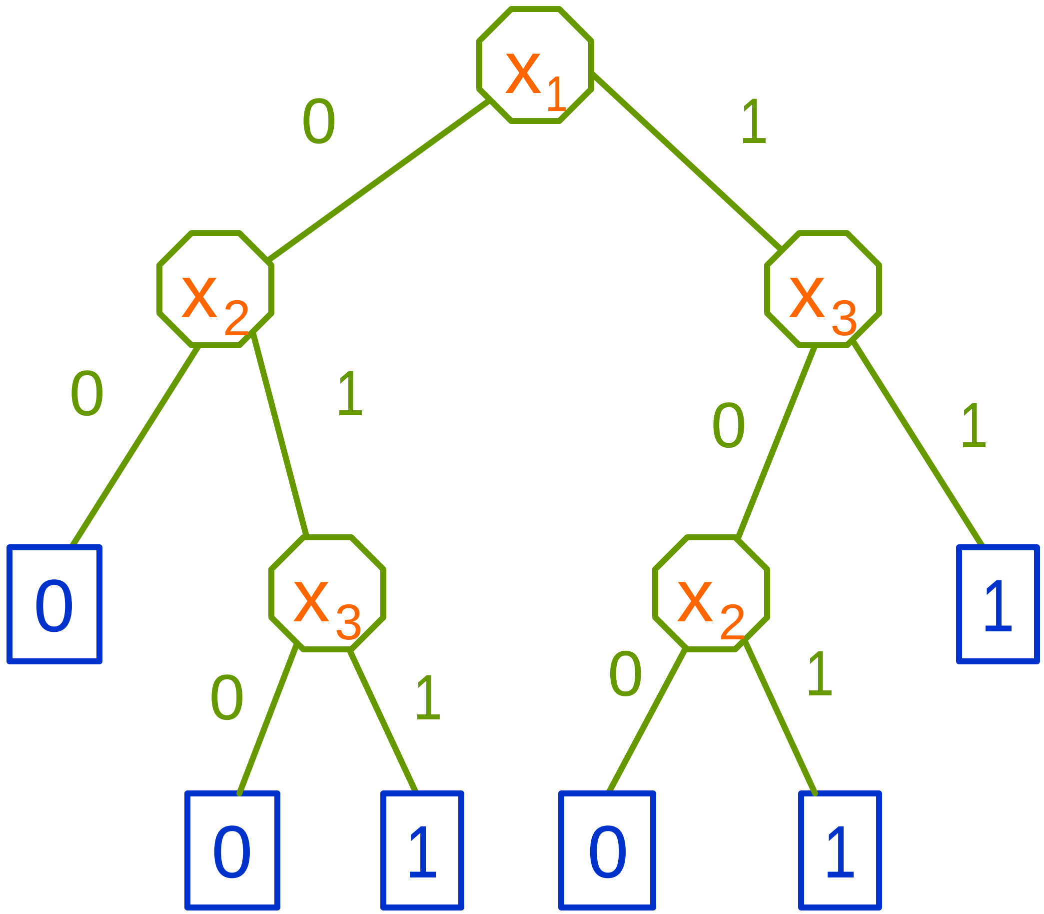
A Boolean Decision Tree



Paths in decision trees

- Every root-leaf path (branch) corresponds to a restriction r of the input variables
  - For $b \in \{0,1\}$, $x_i \leftarrow b$ is in r iff on that branch the out-edge labelled b is taken from node labelled x_i
- The tree T computes f iff for every branch B of T
  - the restriction r corresponding to branch B has the property that $f|_r$ equals the leaf label of B

Tree for $f(x)=\text{‘}x_1+x_2+x_3\geq 2\text{’}$



Property of Decision Trees

- Decision trees $\Rightarrow$DNF:Every function computed by a decision tree of height t can be represented
  - in CNF with clause size at most t
    - clauses correspond to branches with leaf label 0
  - in DNF with term size at most t
    - terms correspond to branches with leaf label 1
- DNF$\Rightarrow$decision tree
  - Canonical conversion

DNF$\Rightarrow$decision tree

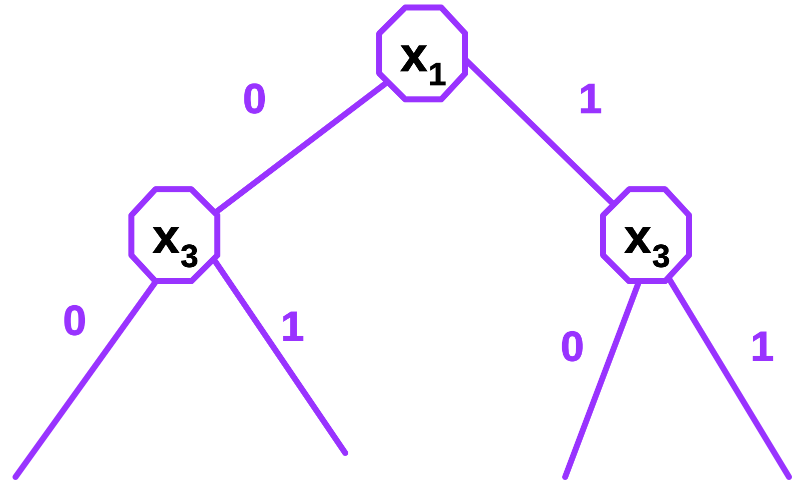
$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$

DNF$\Rightarrow$decision tree

$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$

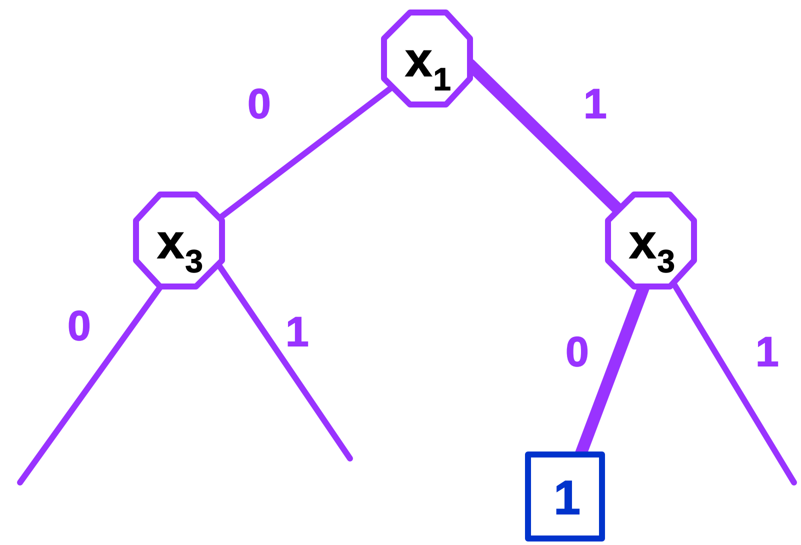
DNF⇒decision tree

$$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$$



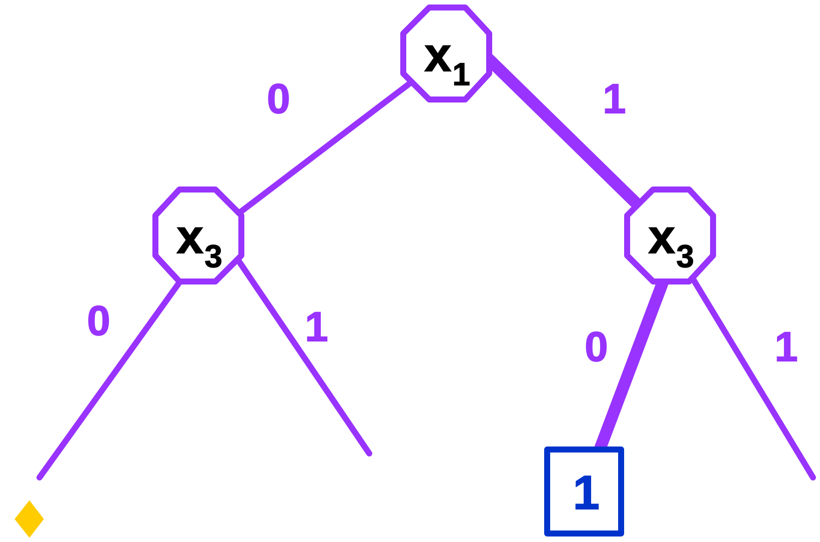
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$



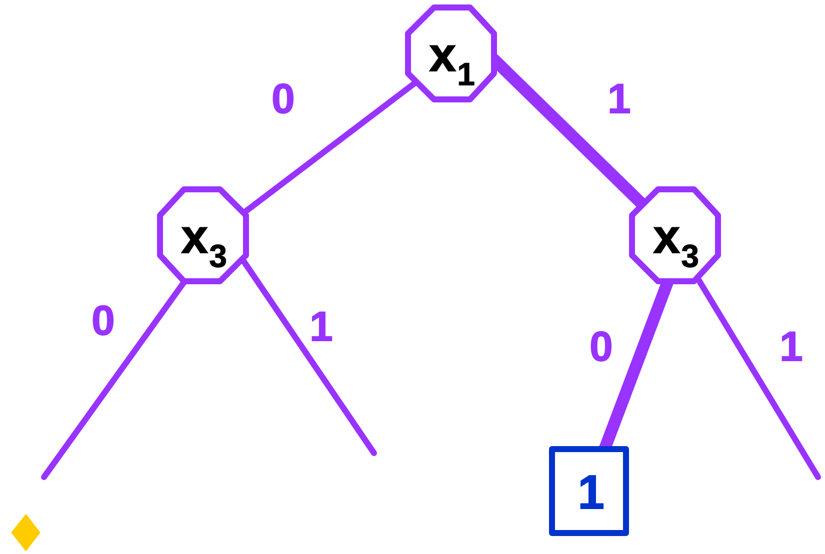
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$



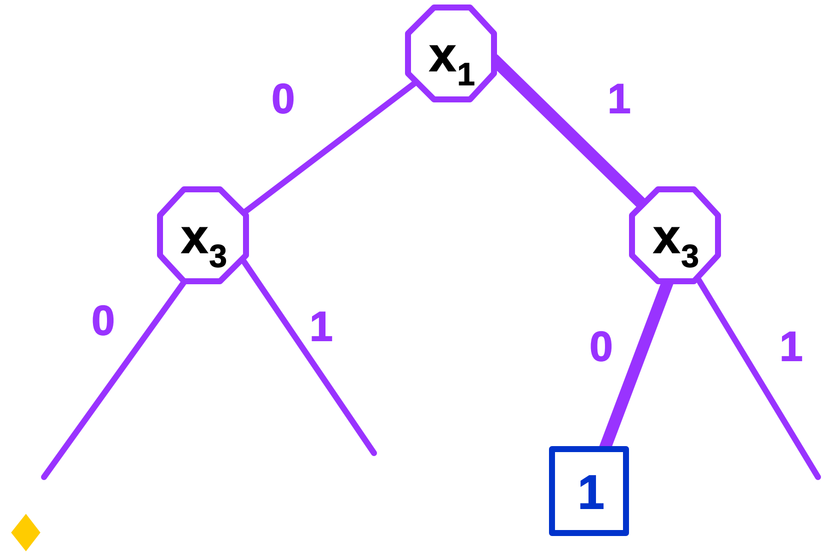
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$



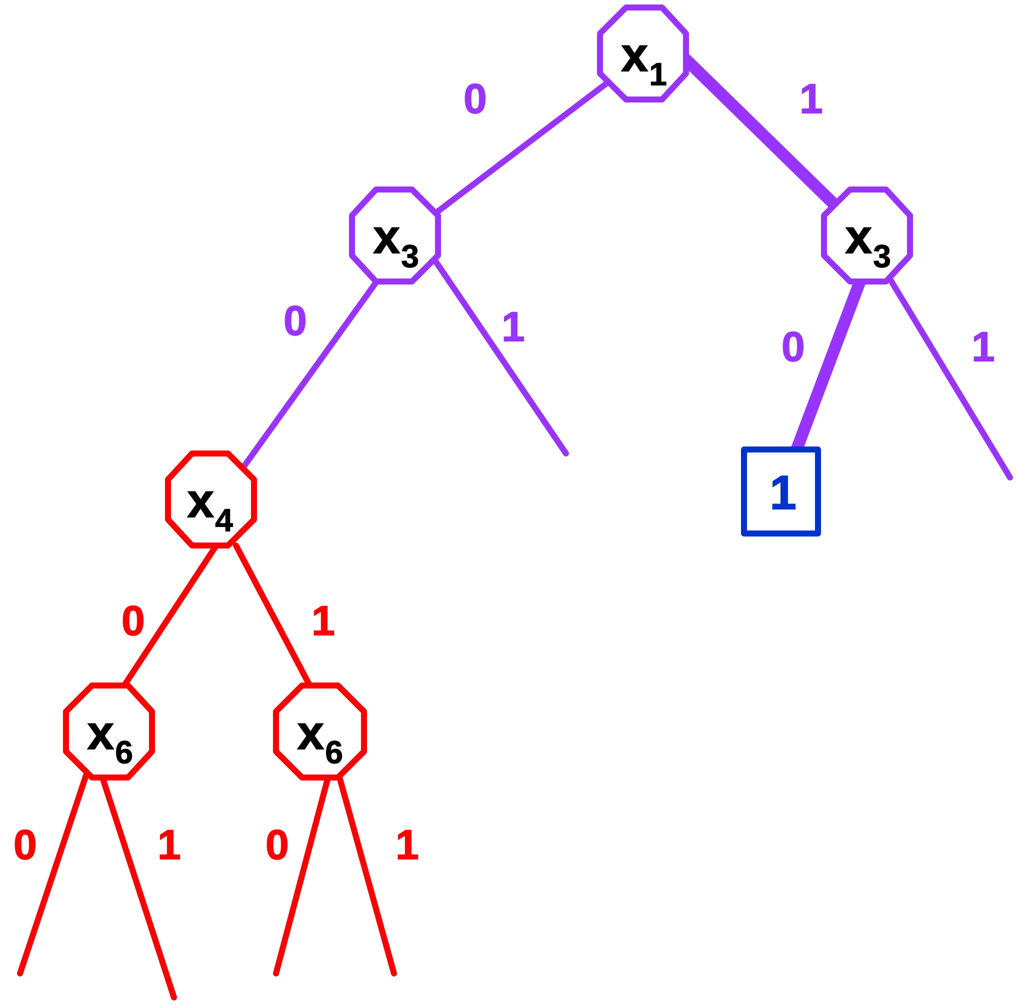
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$



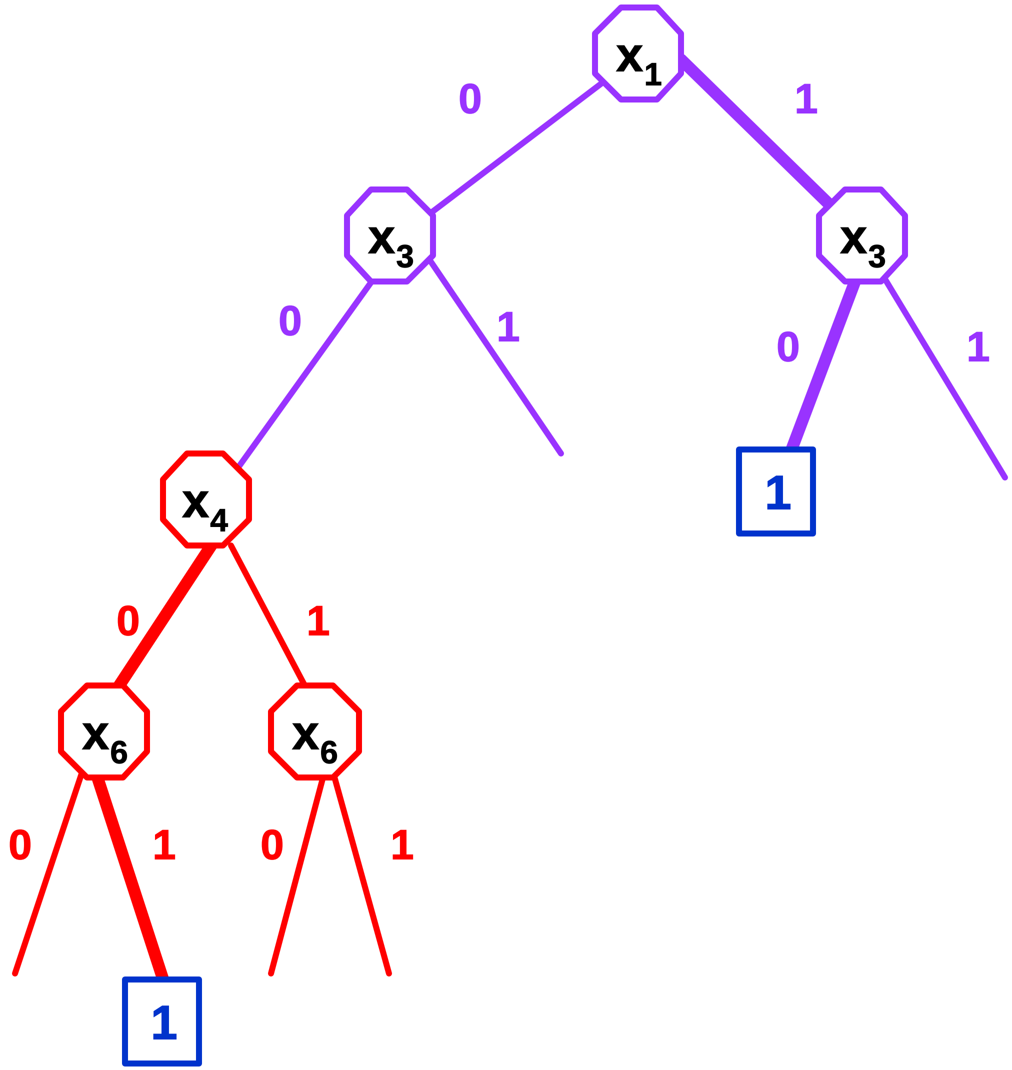
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3 x_4 \vee \overline{x_4} x_6$



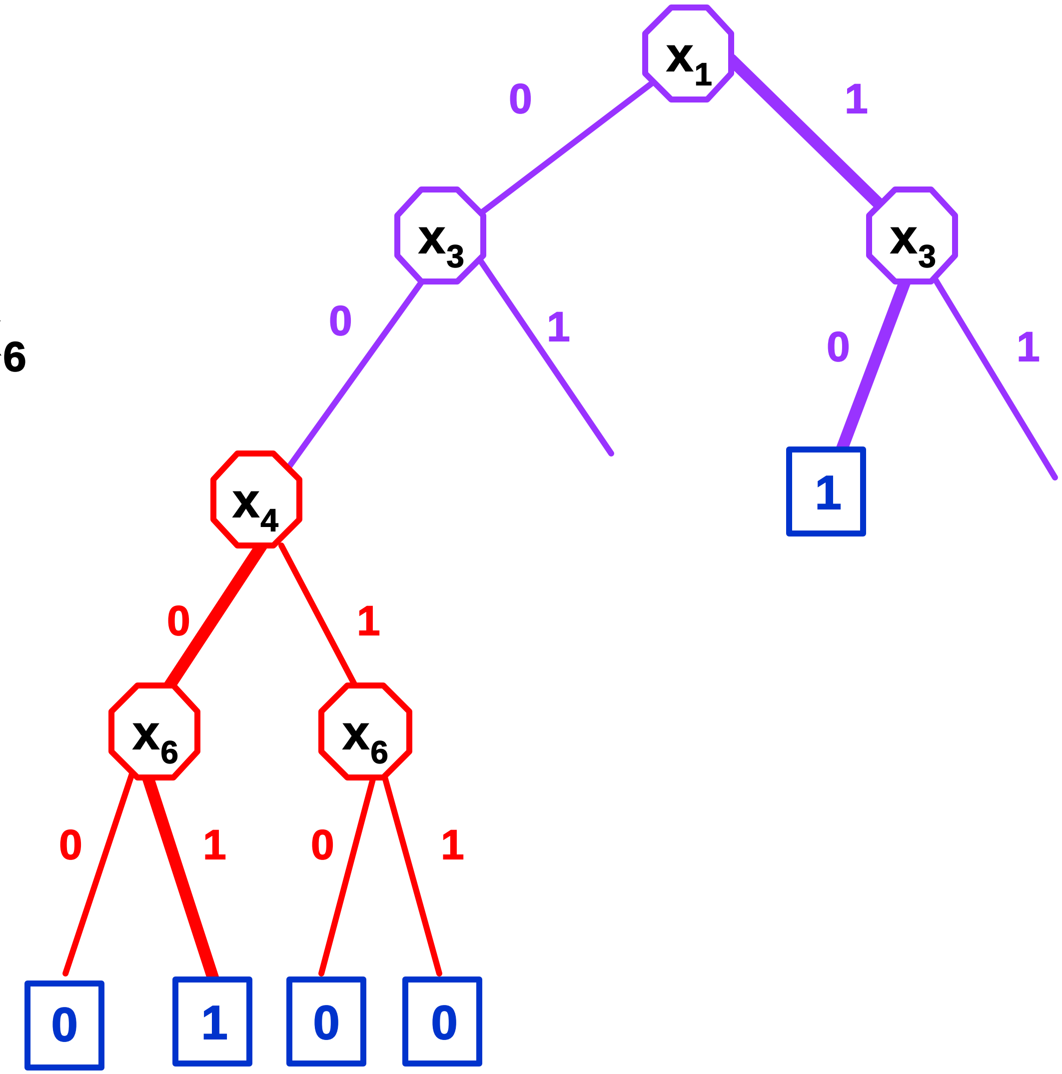
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3 x_4 \vee \overline{x_4} x_6$



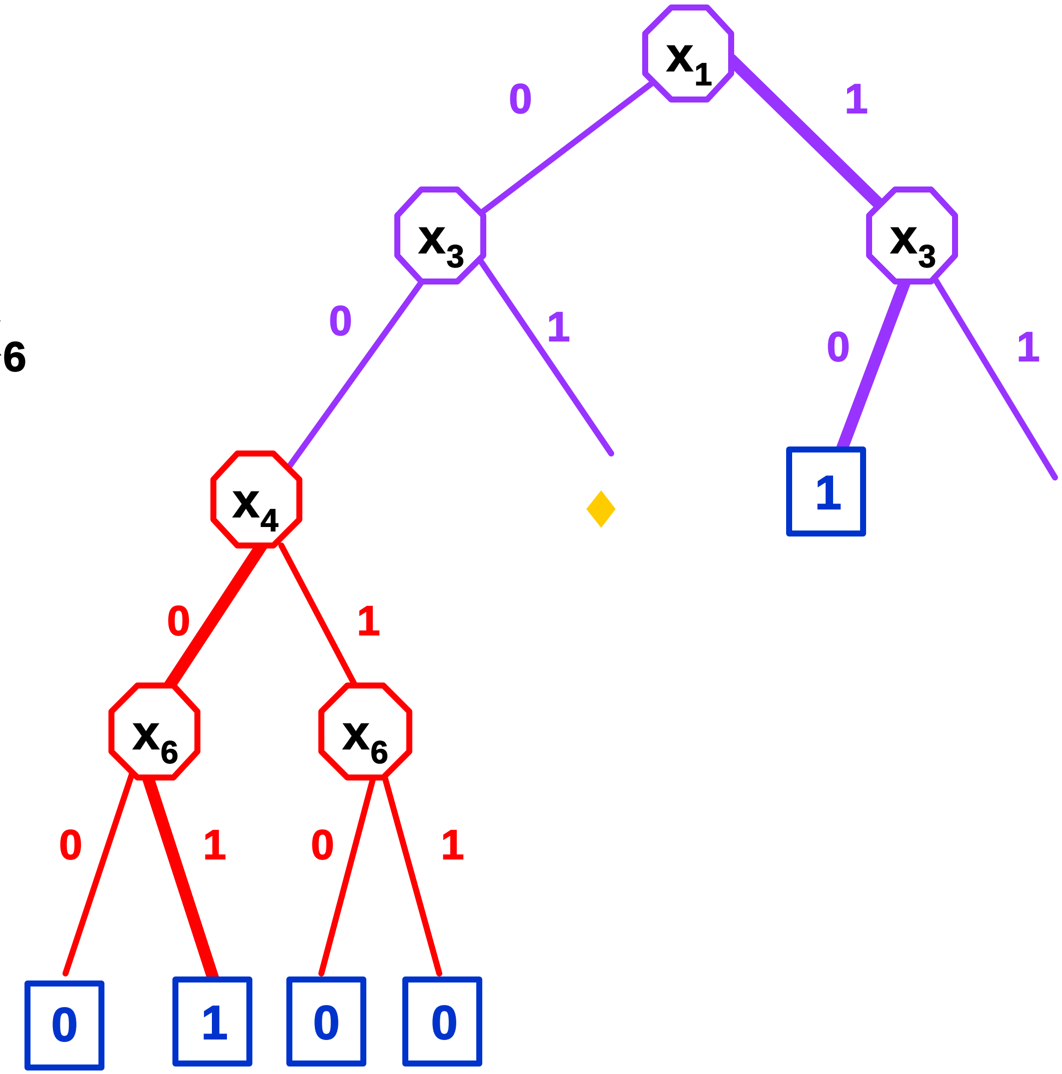
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$



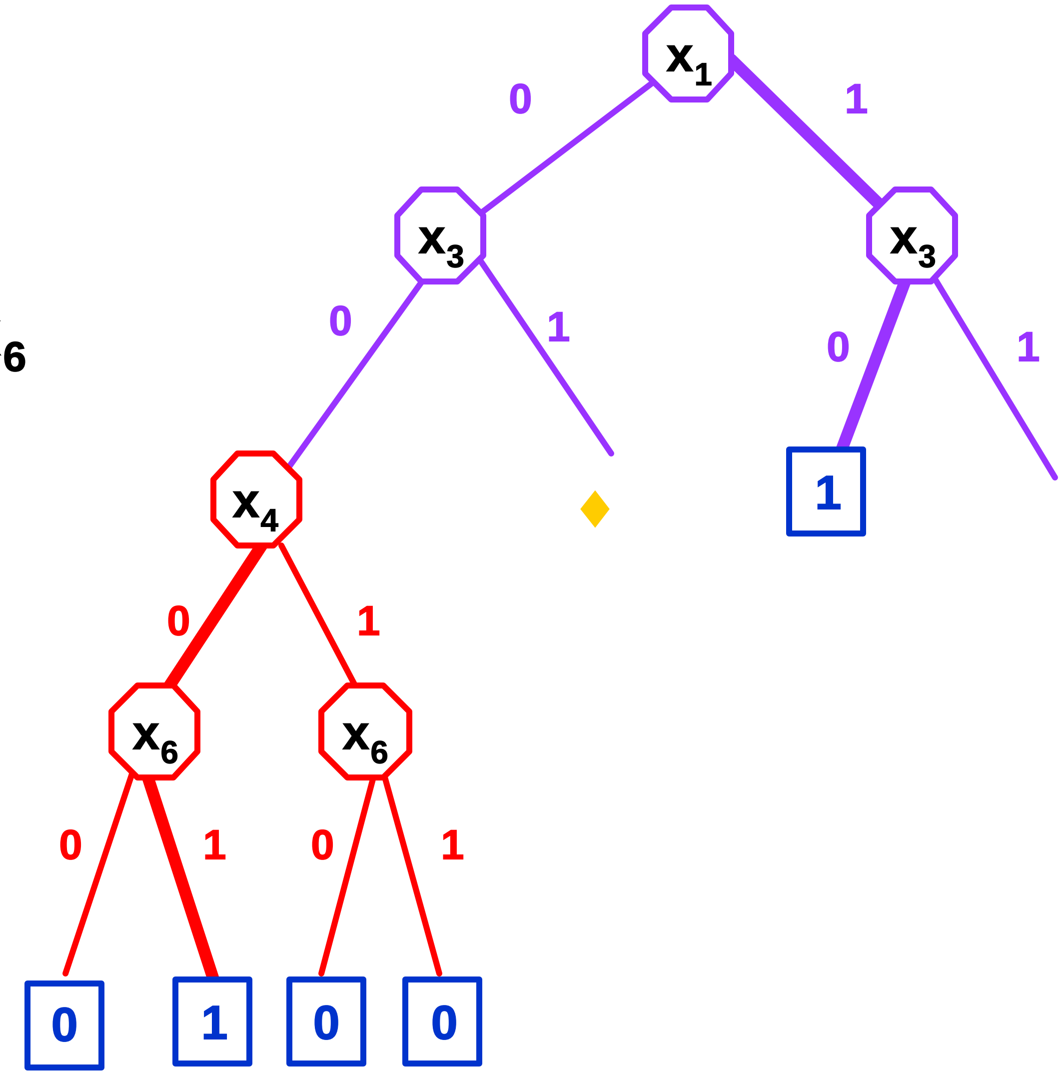
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$



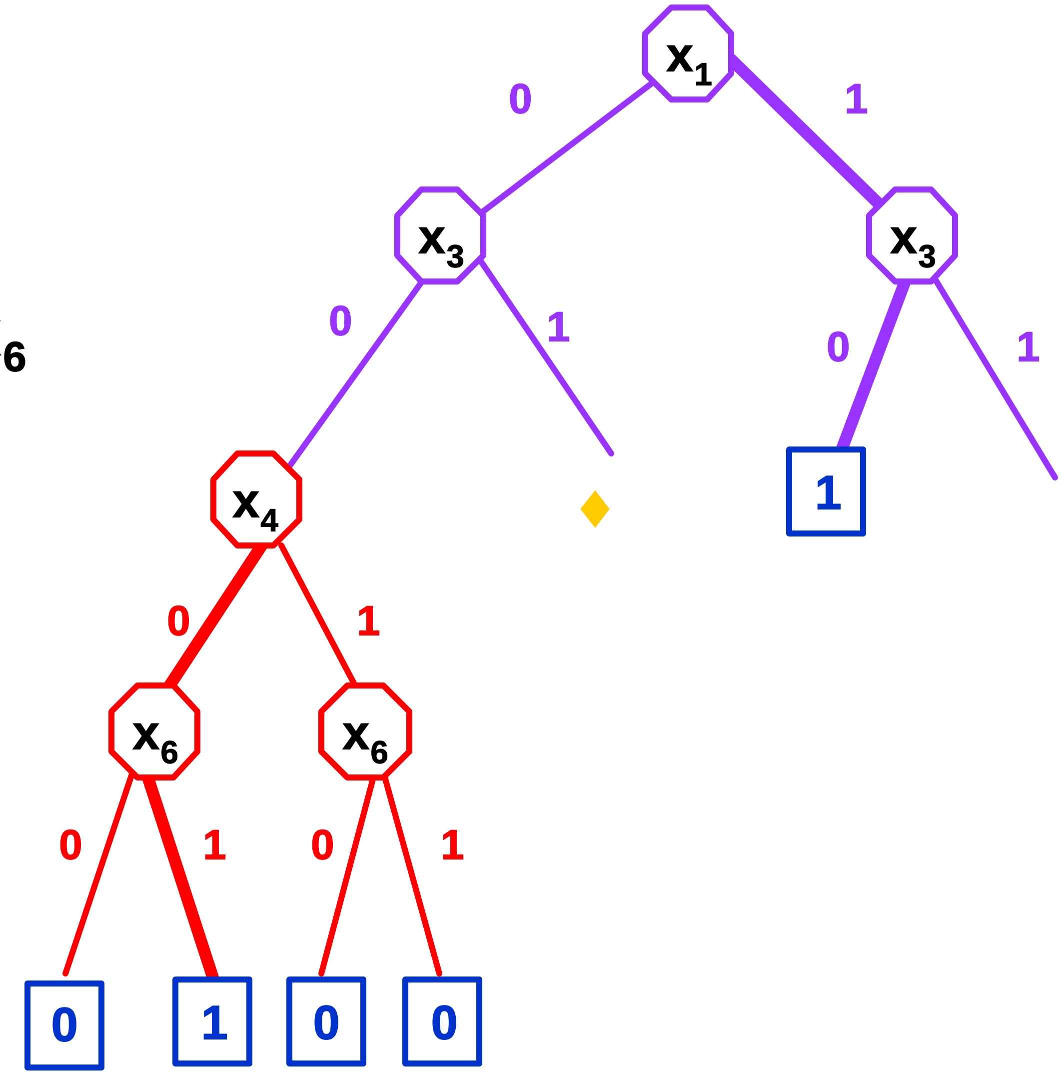
DNF⇒decision tree

$$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$$



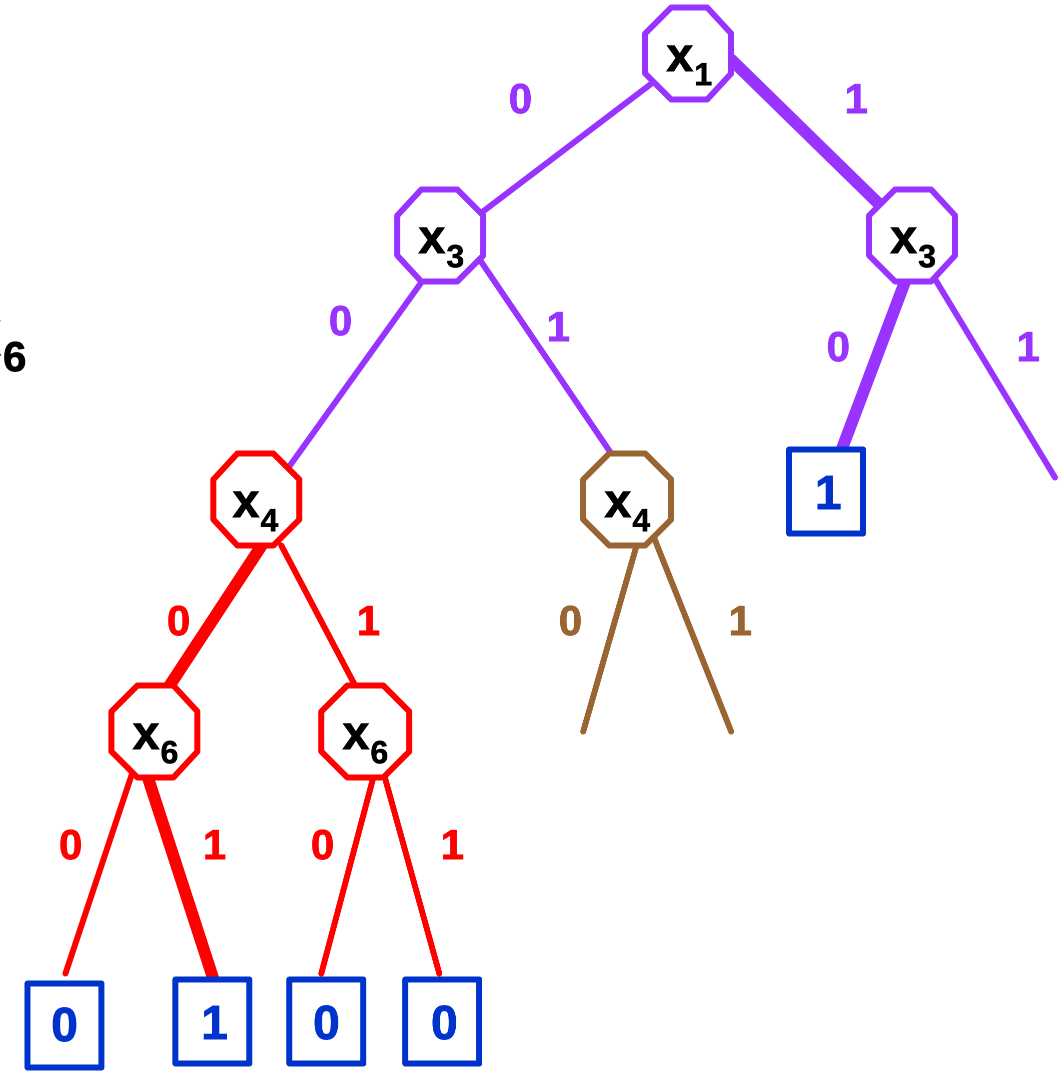
DNF⇒decision tree

$F = x_1\overline{x_3} \lor x_3x_4 \lor \overline{x_4}x_6$



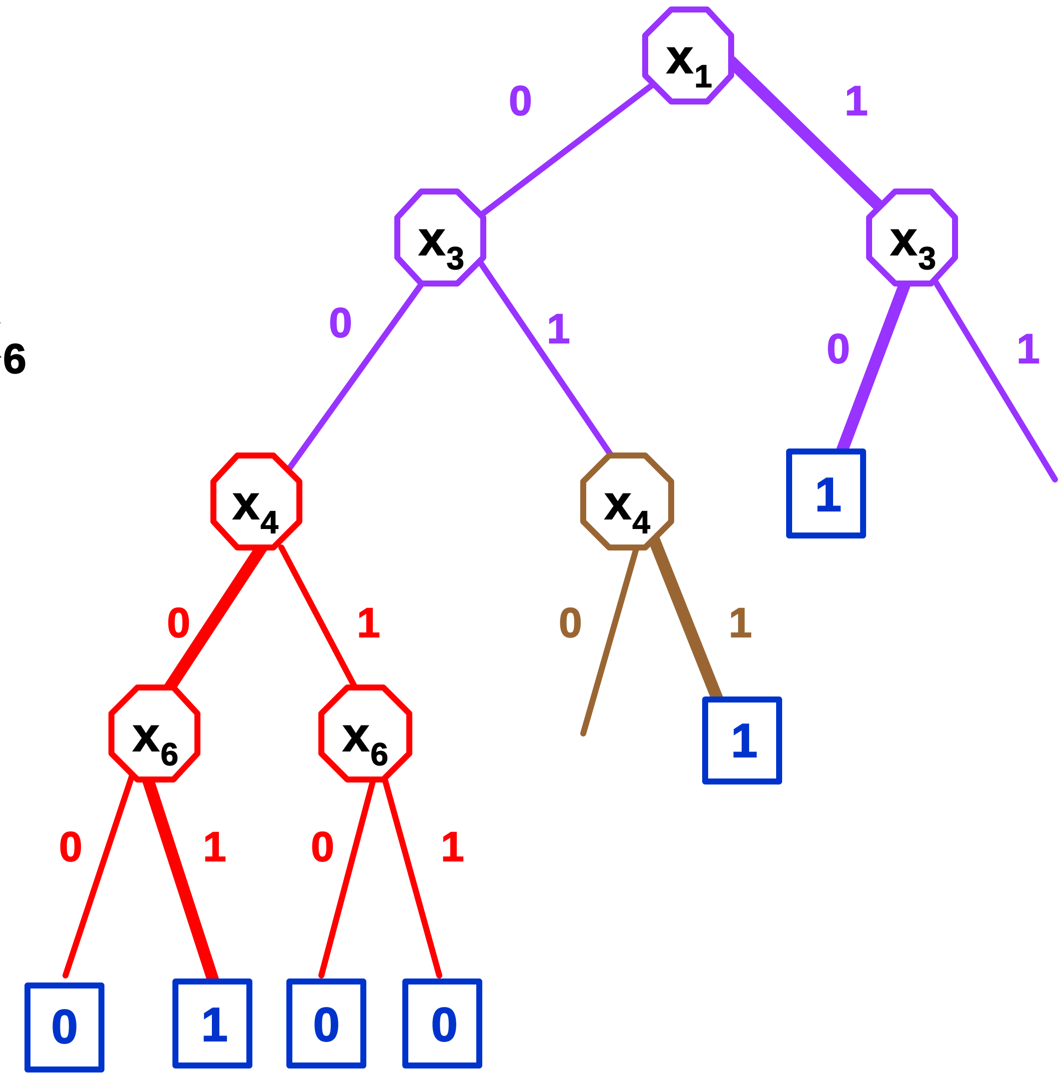
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3 x_4 \vee \overline{x_4} x_6$



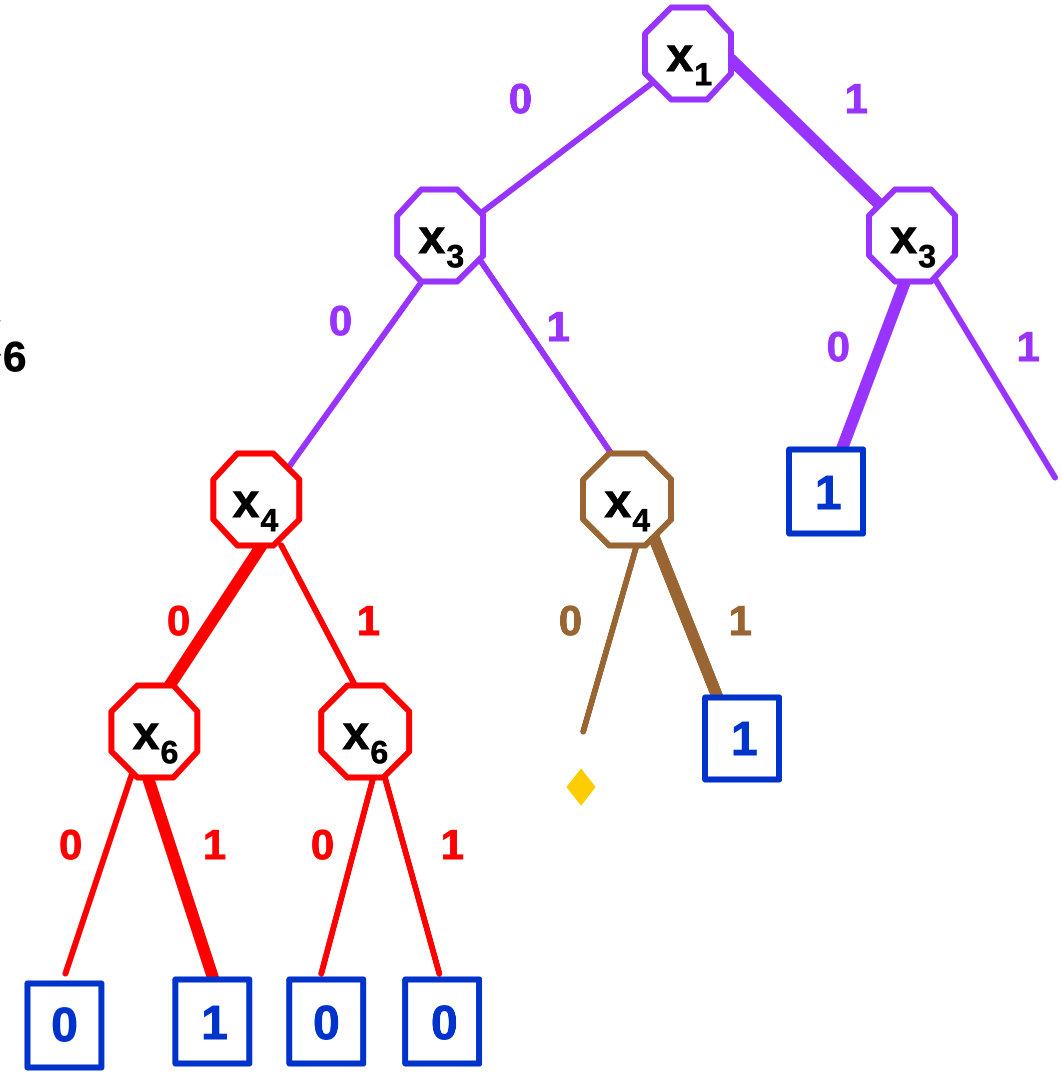
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$



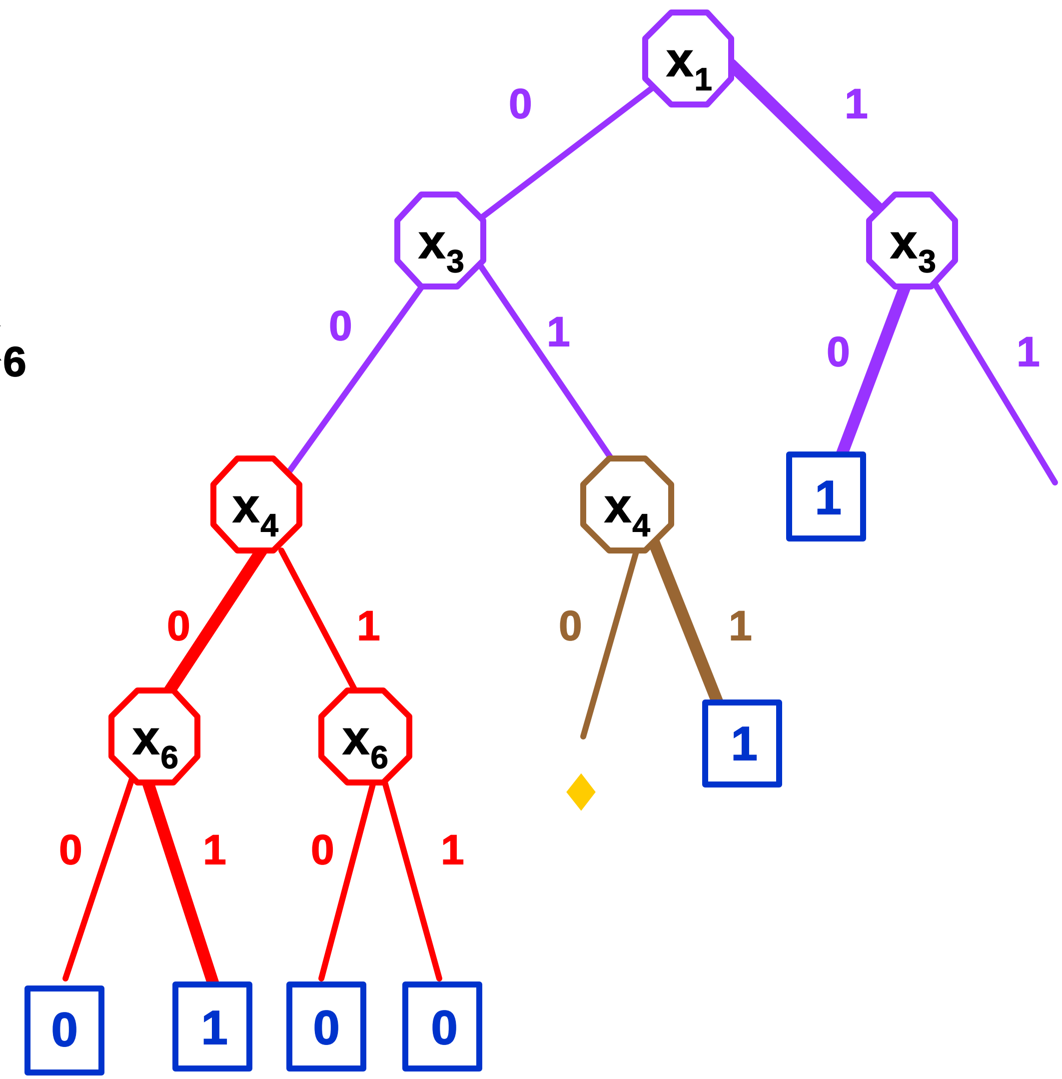
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3 x_4 \vee \overline{x_4} x_6$



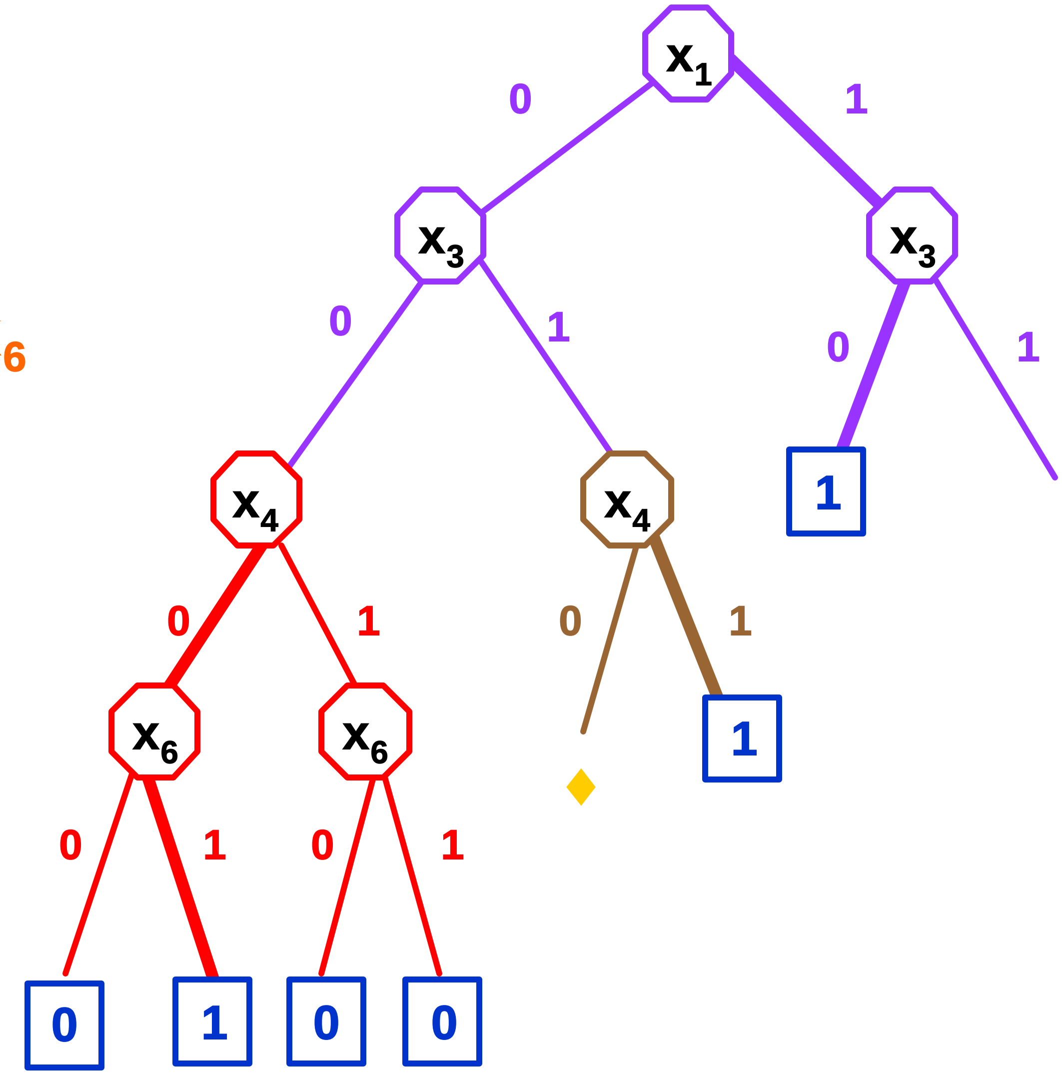
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3 x_4 \vee \overline{x_4} x_6$



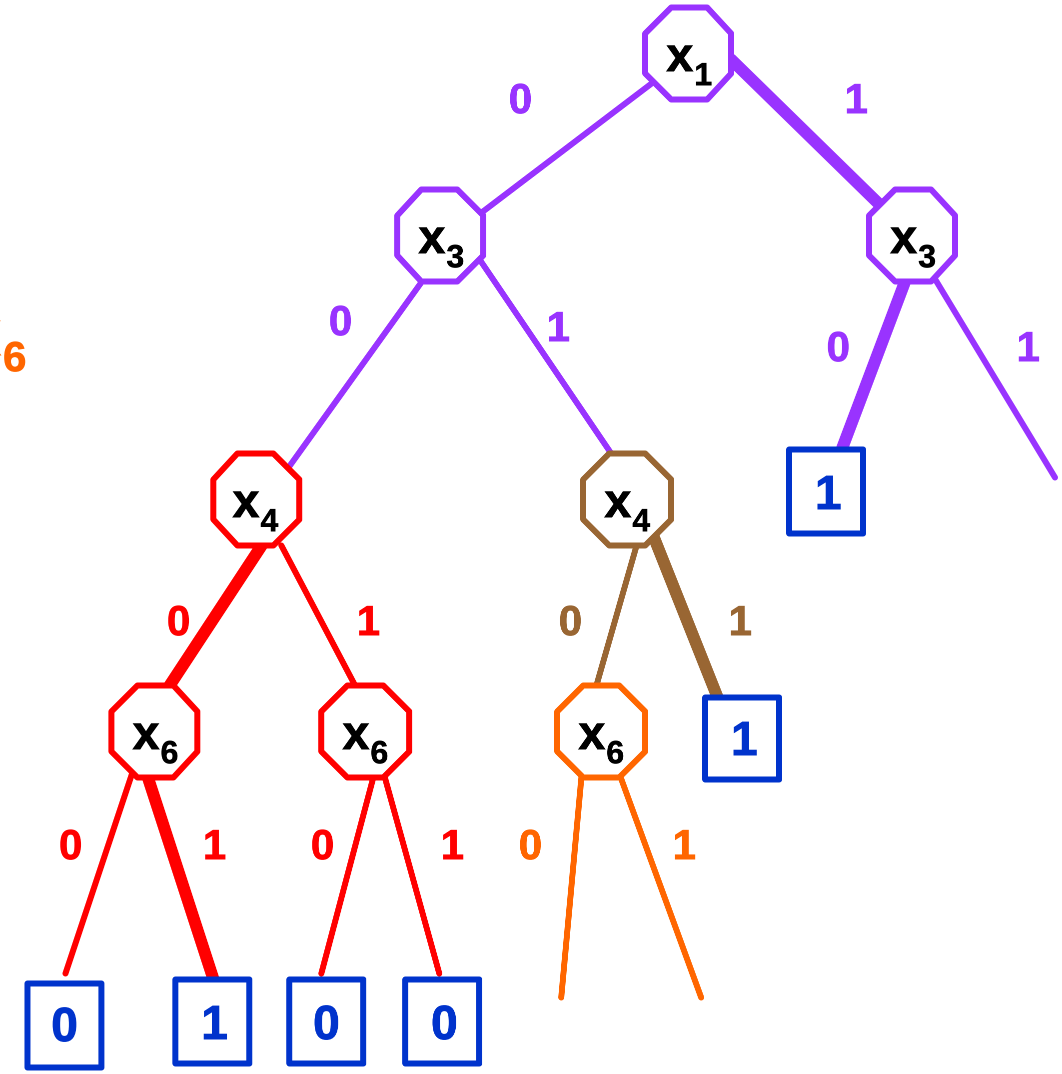
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3 x_4 \vee \overline{x_4} x_6$



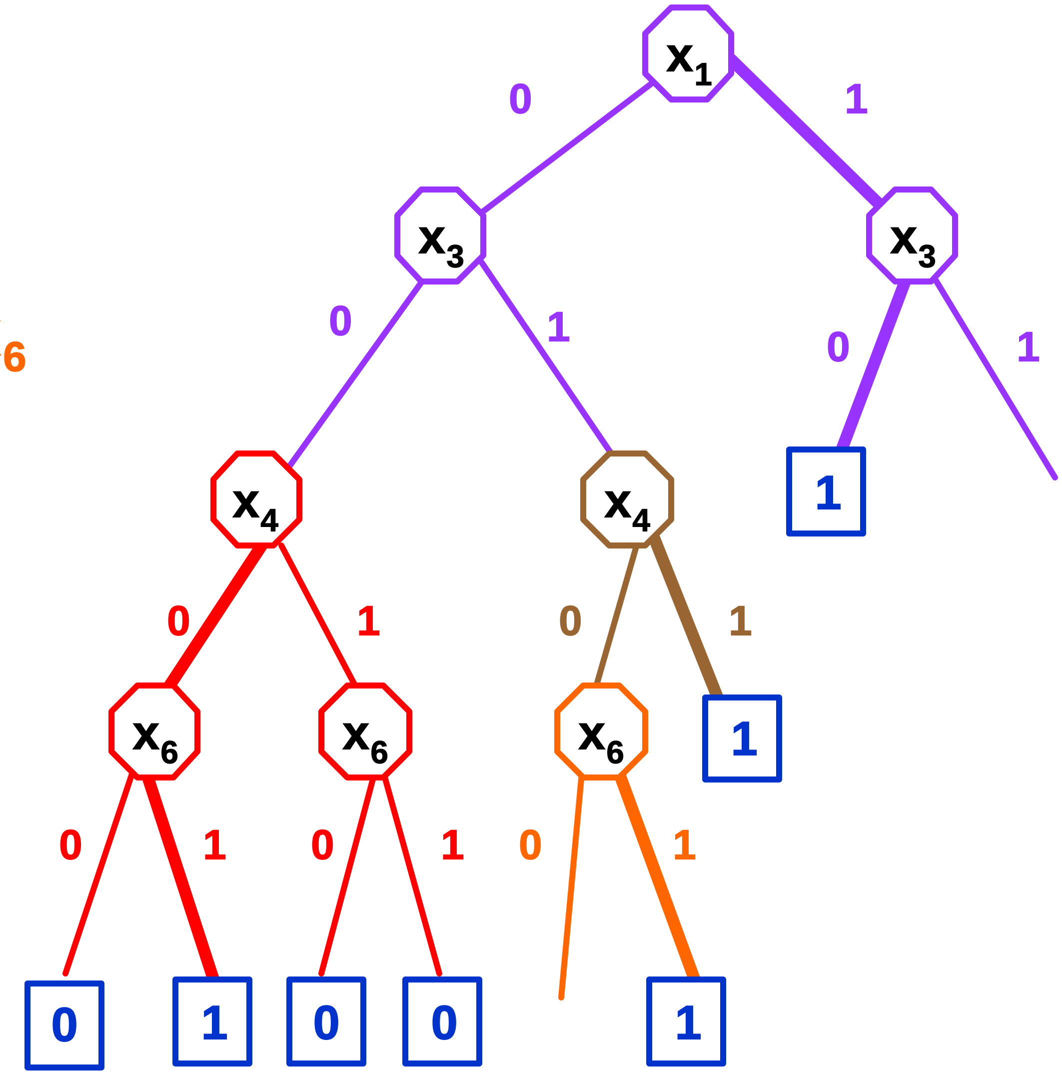
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3 x_4 \vee \overline{x_4} x_6$



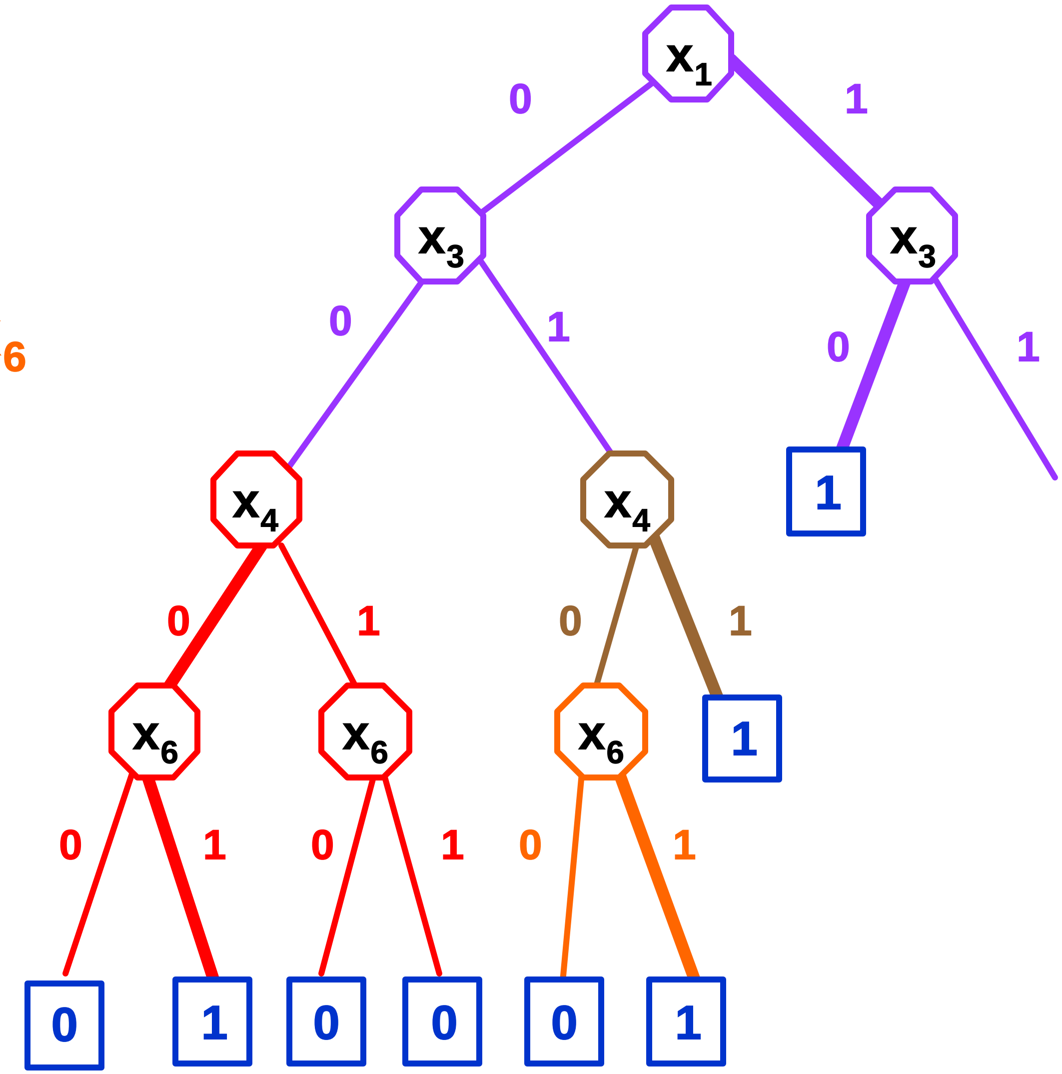
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3 x_4 \vee \overline{x_4} x_6$



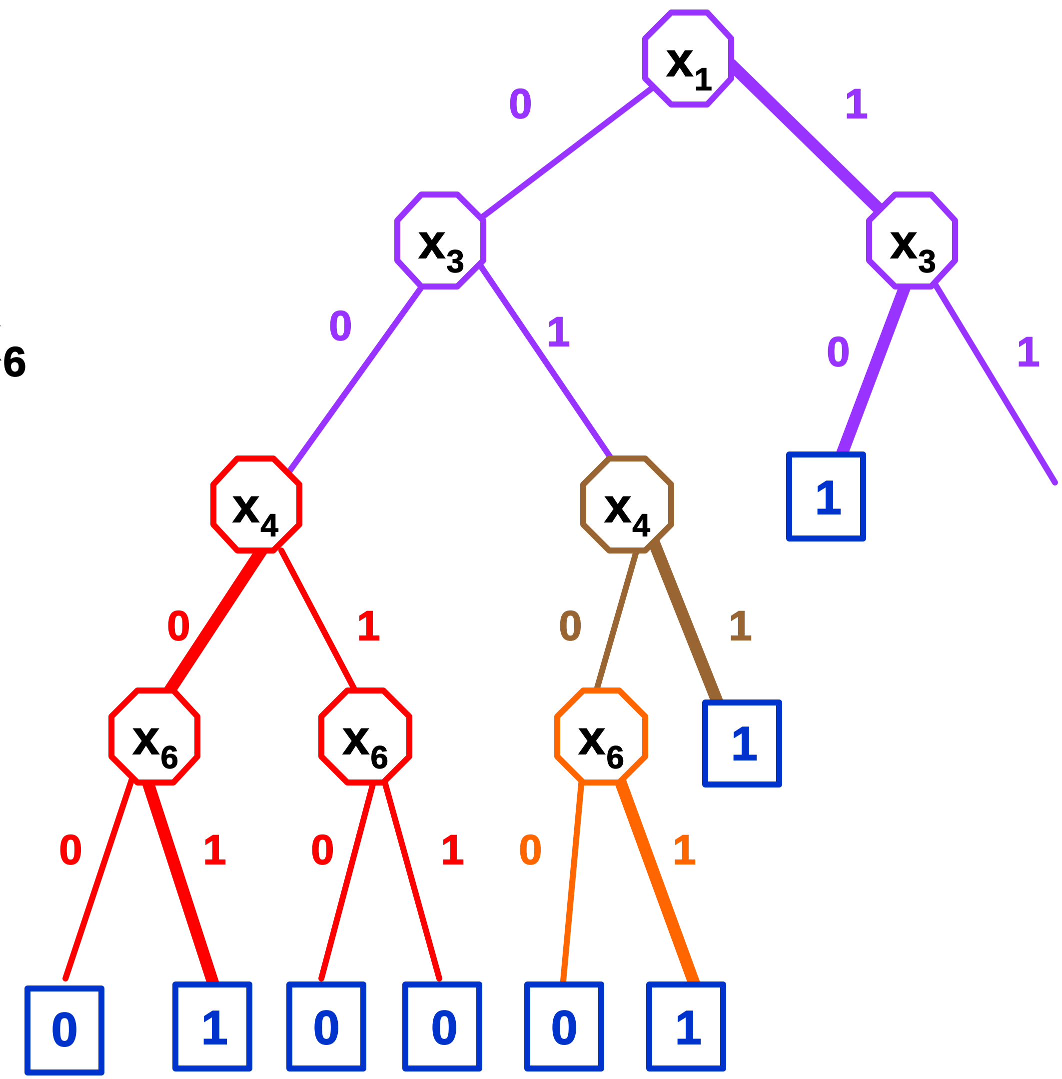
DNF⇒decision tree

$$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$$



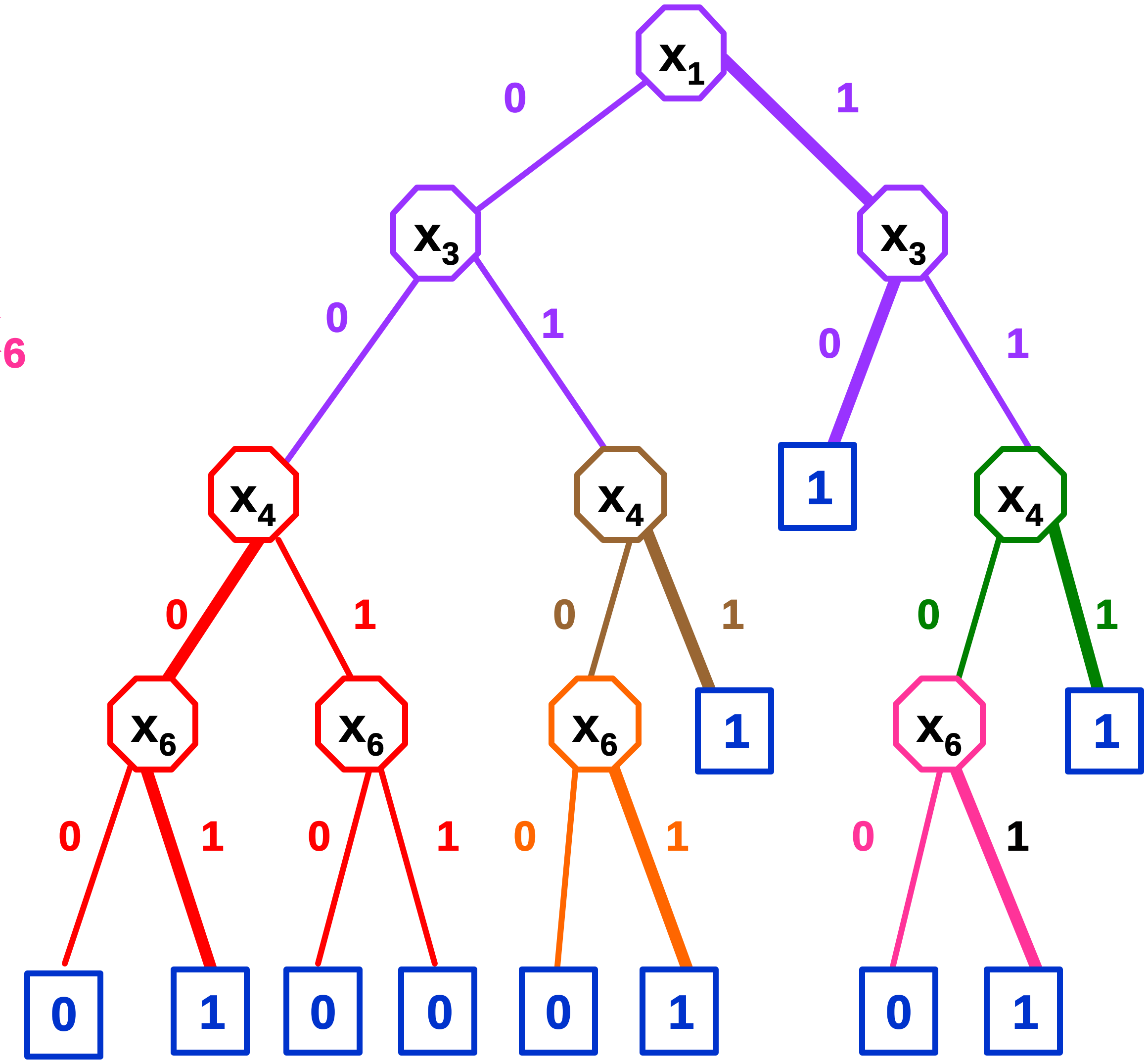
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$



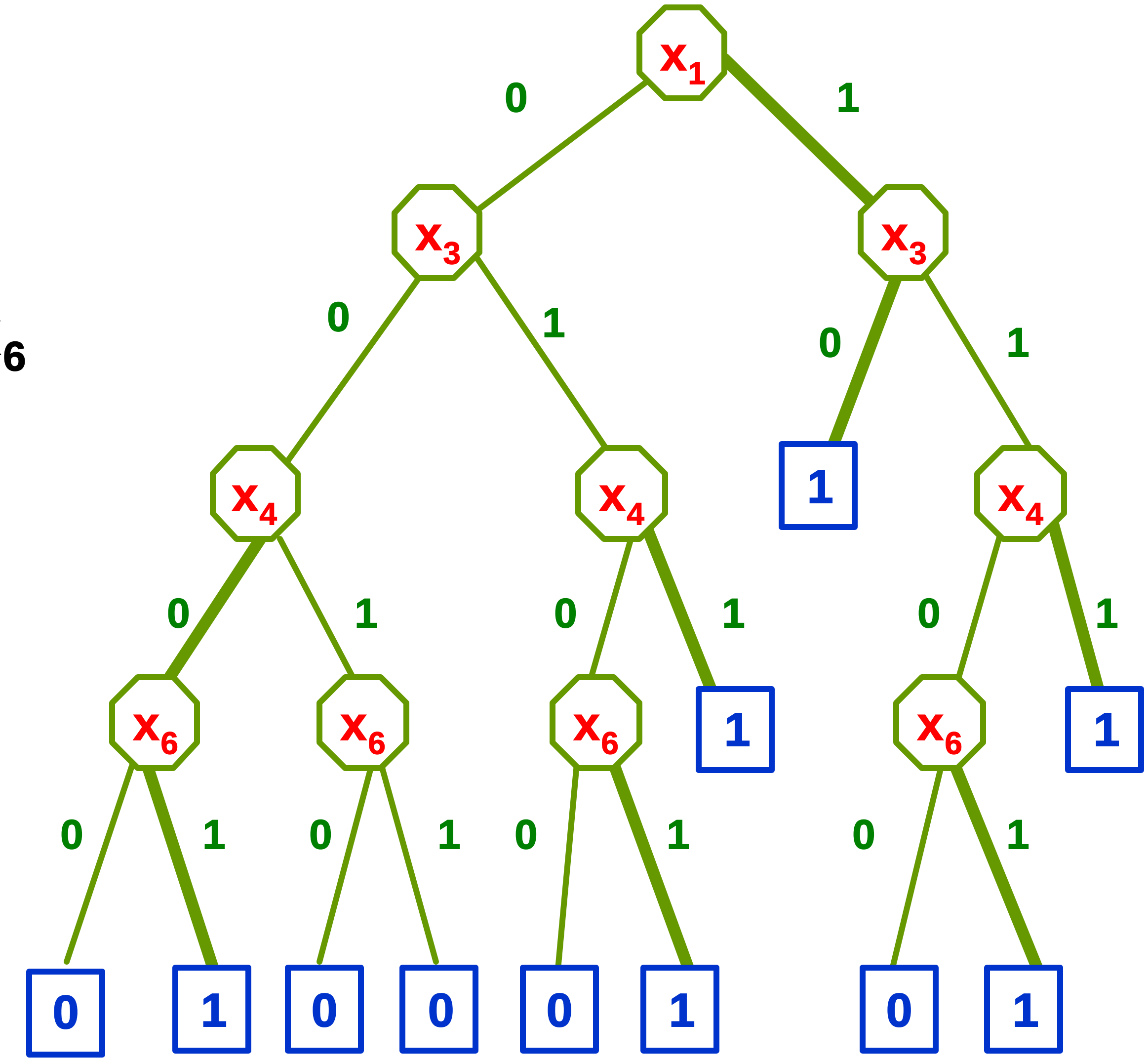
DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3 x_4 \vee \overline{x_4} x_6$



DNF⇒decision tree

$F = x_1\overline{x_3} \vee x_3x_4 \vee \overline{x_4}x_6$



Parity properties

- For any restriction r, $\text{Parity}|_r$ is either parity or its negation on the variables that are still not assigned values

- Parity or its negation requires a decision tree of height n

  - Compare with $x_1 \vee \ldots \vee x_n$
    - any decision tree also requires height n
    - but most restrictions of it are constant and so only require height 0

Restriction for constant-depth circuits

- An **(S,d)**-circuit will be an unbounded fan-in circuit of size $\le$ **S** and depth $\le$ **d**

- To show that no **(S,d)**-circuit **C** computes function **f**, find a set **$R_{S,d}(f)$** of restrictions s.t.
  - For any **(S,d)-**circuit **C**, there is a $\rho \in$ **$R_{S,d}(f)$** s.t. we can associate a **short*** Boolean decision tree **$T(g)$** to each gate **g** of **C**, s.t. **$T(g)$** computes **$g|_\rho$**

  - For any $\rho \in$ **$R_{S,d}(f)$**, **$f|_\rho$** is **not** computed by any short* decision tree

  *relative to the number of variables unset by ρ

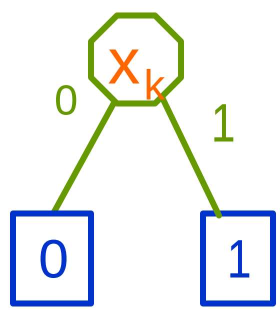
Restriction for constant-depth circuits

- An **(S,d)**-circuit will be an unbounded fan-in circuit of size $\leq$ **S** and depth $\leq$ **d**
- To show that no **(S,d)**-circuit **C** computes function **f**, find a set **$R_{S,d}(f)$** of restrictions s.t.
  - For any **(S,d)-**circuit **C**, there is a $\rho \in$ **$R_{S,d}(f)$** s.t. we can associate a **short*** Boolean decision tree **$T(g)$** to each gate **g** of **C**, s.t. **$T(g)$** computes **$g|_\rho$**
  - For any $\rho \in$ **$R_{S,d}(f)$**, **$f|_\rho$** is **not** computed by any short* decision tree

  *in case of parity this just means < number of variables

How to find restrictions for Parity circuits

- Start at the inputs of the circuit and work upwards a layer at a time,
  - maintaining a current restriction ρ_i and a tree $\mathbf{T_i(g)}$ for each gate $\mathbf{g}$ in the first $\mathbf{i}$ layers s.t. $\mathbf{T_i(g)}$ computes $\mathbf{g}|_{\rho_{\mathbf{i}}}$
  - For layer $\mathbf{0}$, gates are input variables, ρ_0 is empty and decision trees have height 1



How to find restrictions for Parity circuits

- Working up the layers of the circuit
  - If **h=Øg** then let **T_i(h)** be **T_i(g)** with its leaf labels toggled between **0** and **1**.

  - If **h=(g_1Ú ... Úg_t)** then the function **h|**$_{r_i}$ may require tall decision trees even if all **T_i(g_j)** are short
    - so we look for a further small restriction π to the inputs in the hopes of simplifying h|$_{r_i}$ so that the tree will be short

    - We'd like to choose **one** p that simultaneously does this for **all** the unbounded fan-in V's in this layer (up to S of them!)

What will we do once we have π

- Once we have such a restriction π - a tall order it seems
  - Set $r_{i+1} = r_i\, p$
  - Short $\mathbf{T_{i+1}(h)}$ for **h** in this layer exist by our assumed properties of p
  - For all gates **g** below this layer, set $\mathbf{T_{i+1}(g)=T_i(g)}|_p$
  - continue upward...

  - We end by letting $r=r_d$ and we will have chosen the various p so that the trees will be shorter than the number of inputs that r leaves unset
    - circuit cannot compute parity

Finding π

- Probabilistic method
  - Show that a randomly chosen small p fails to shorten the decision tree for any single V-gate h in this layer with probability < 1/S
  - There are at most **S** V-gates in this layer, so Pr[$an V-gate in this layer not shortened by p] < 1
  - ...so there must exist a small p that does the job
    - choose it

Hastad's Switching Lemma

- Let $\mathbf{R_{k,n}}$ be the set of all restrictions to variables $\mathbf{x_1,\ldots,x_n}$ that leave precisely $\mathbf{k}$ variables unset

- **Lemma:** Given a DNF formula $\mathbf{F}$ in variables $\mathbf{x_1,\ldots,x_n}$ with terms of size at most $\mathbf{t}$, for ρ chosen uniformly at random from $\mathbf{R_{k,n}}$, if $\mathbf{n>12tk}$ then

  $$\Pr[\text{canonical decision tree for } \mathbf{F}|_\rho \text{ has height} \geq \mathbf{t}] < \mathbf{2^{-t}}.$$

Final analysis

- Maintain trees of height $\mathbf{t=\log_2 S}$
- Number of variables decreases by a factor of
  $\mathbf{13t=13\log_2 S}$ per layer

- Height will be less than # of variables if
  $\log_2 S < n/(13\log_2 S)^d$ i.e. $\log_2 S < n^{1/(d+1)}/13$
  - can't compute parity if this holds

- Can save one power of $\log_2 S$ by being careful

Restriction method in proof complexity

- **Theorem** [Ajtai,PBI,KPW]: **ontoPHP**$^{n+1 \circledR n}$ requires exponential size **AC^0-Frege** proofs

- **Theorem** [Ajtai,BP] **Count**$^{2n+1|2}$ requires exponential size **AC^0-Frege** proofs even given **PHP**$^{m+1 \circledR m}$ as extra axiom schemas

- **Theorem** [BIKPP] **Count**$^{pn+1|p}$ requires exponential size proofs even given **Count**$^{qm+1|q}$ as axiom schemas

Restrictions in Proof Complexity

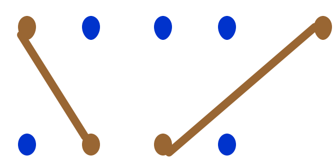
- In circuit complexity,
  - for each gate **g** we defined decision trees **T(g)** that precisely compute each **g**$|_r$ in the circuit
- Obvious analogue in proof complexity, e.g. in proof of a tautology
  - do the same
- But this can't work
  - every formula in the proof computes the constant function 1 since it is a tautology!

What we do instead

- Come up with a different notion of decision trees that **approximates** each formula so that
  - bigger proof needed for a tautology implies worse approximation of it
  - decision trees are well-behaved under restrictions
  - approximation is particularly bad for the goal formula **F** you want to prove
    - Any short approximating decision tree for
      - **F** looks like **false**
      - an axiom looks like **true**
      - any formula with a short proof looks like **true**
  - like circuit case define decision trees for **each subformula** in the proof and tailor decision trees & restrictions to **F**

Restrictions for $PHP^{n+1 \to n}$

- Don't want restrictions to force $PHP^{n+1 \to n}$ to true so...
- Restrictions p are partial matchings as before

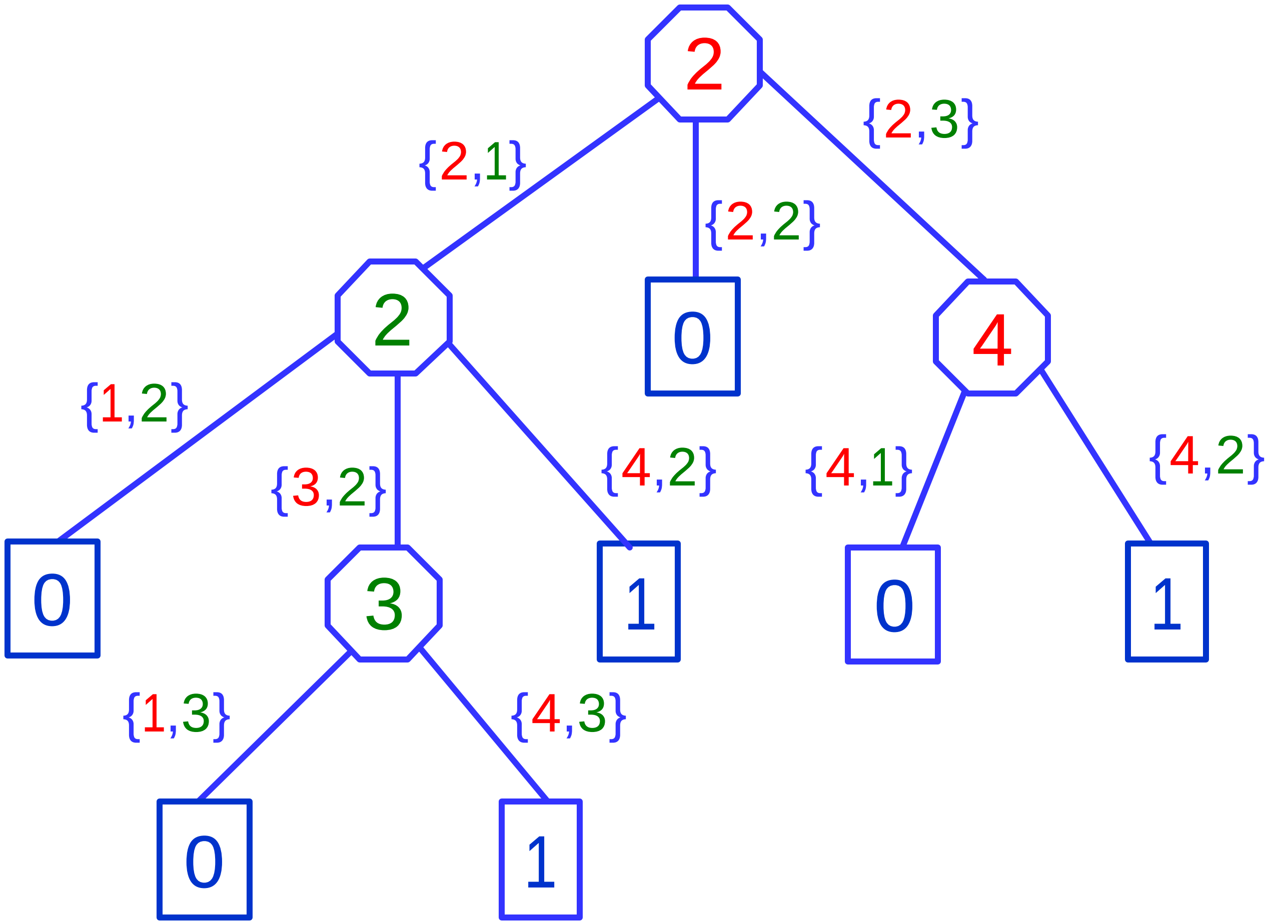


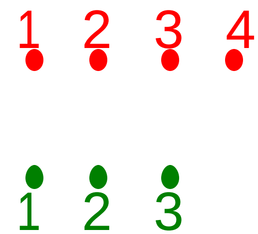
- Let $\mathbf{R^{k,n}}$ be the set of all partial matching restrictions that leave exactly $\mathbf{k}$ holes unset

Bipartite matching decision trees

- Queries are either
  - the name of a pigeon, or
    - answer is the mapping edge for that pigeon
  - the name of a hole, or
    - answer is the mapping edge for that hole
- Every path corresponds to a partial matching between pigeons and holes
    - No repetition of a node name that was already used higher in the tree
- Leaves are labelled 0 or 1

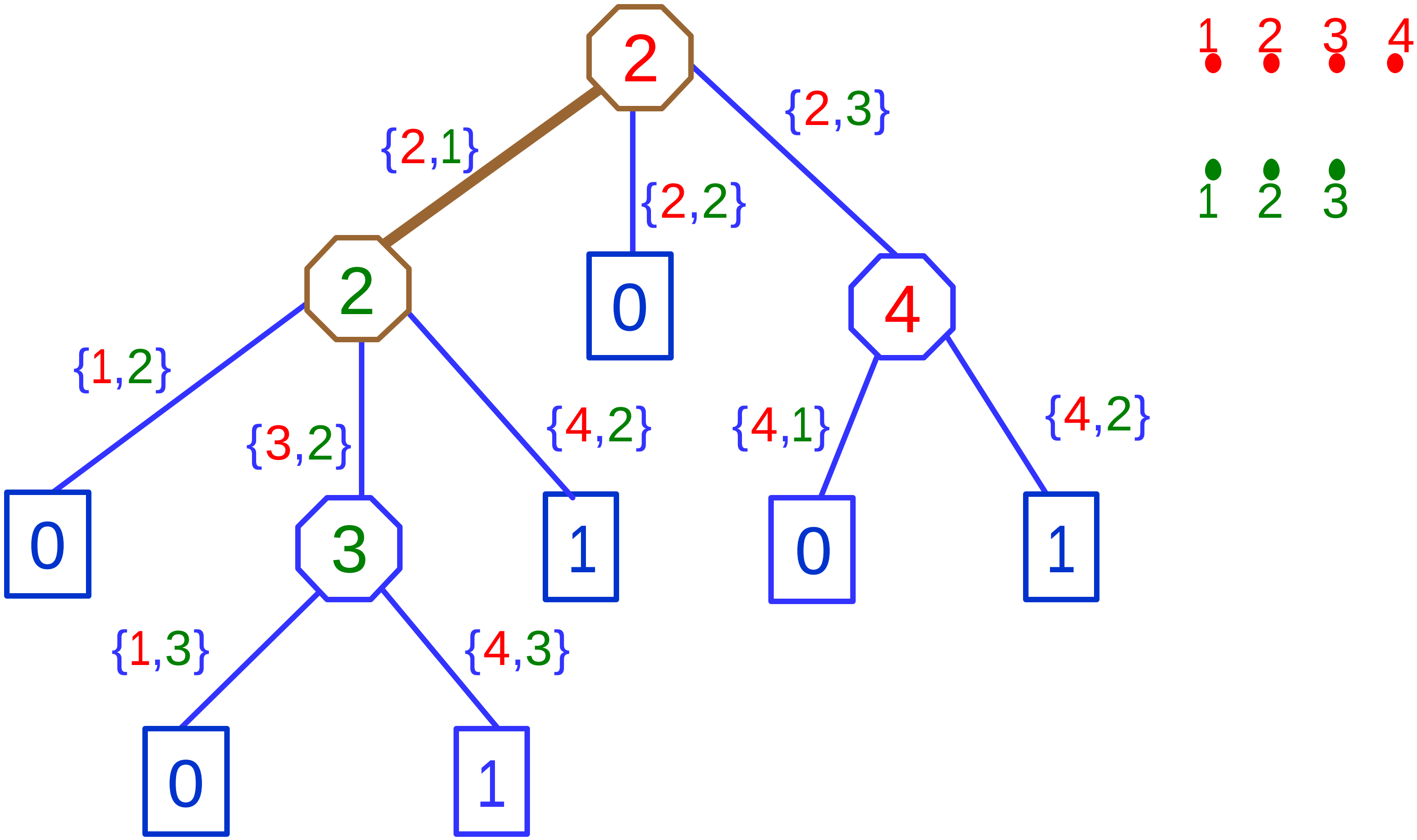
A matching decision tree





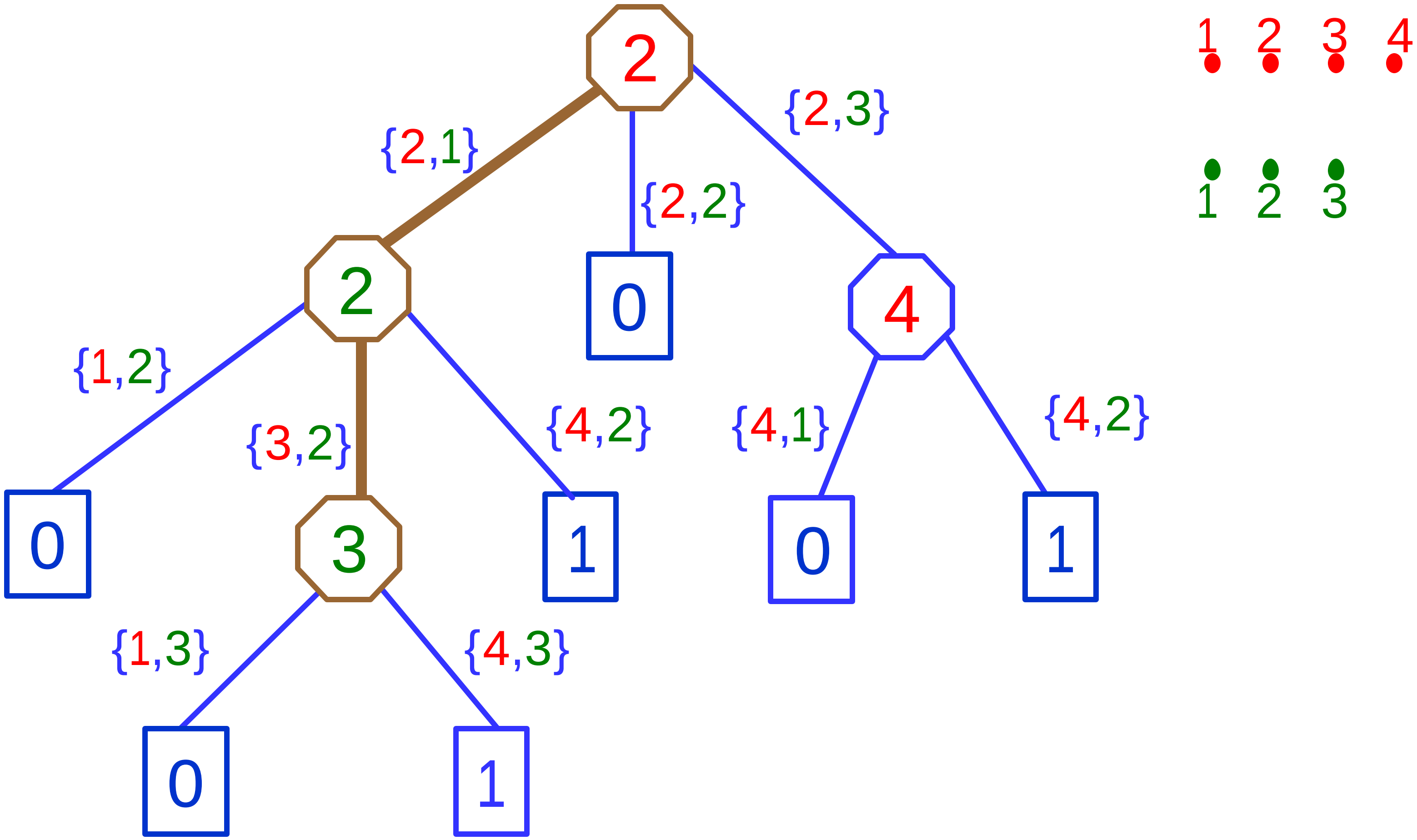
$$P_{21}P_{32}P_{43} \vee P_{21}P_{42} \vee P_{23}P_{42}$$

A matching decision tree



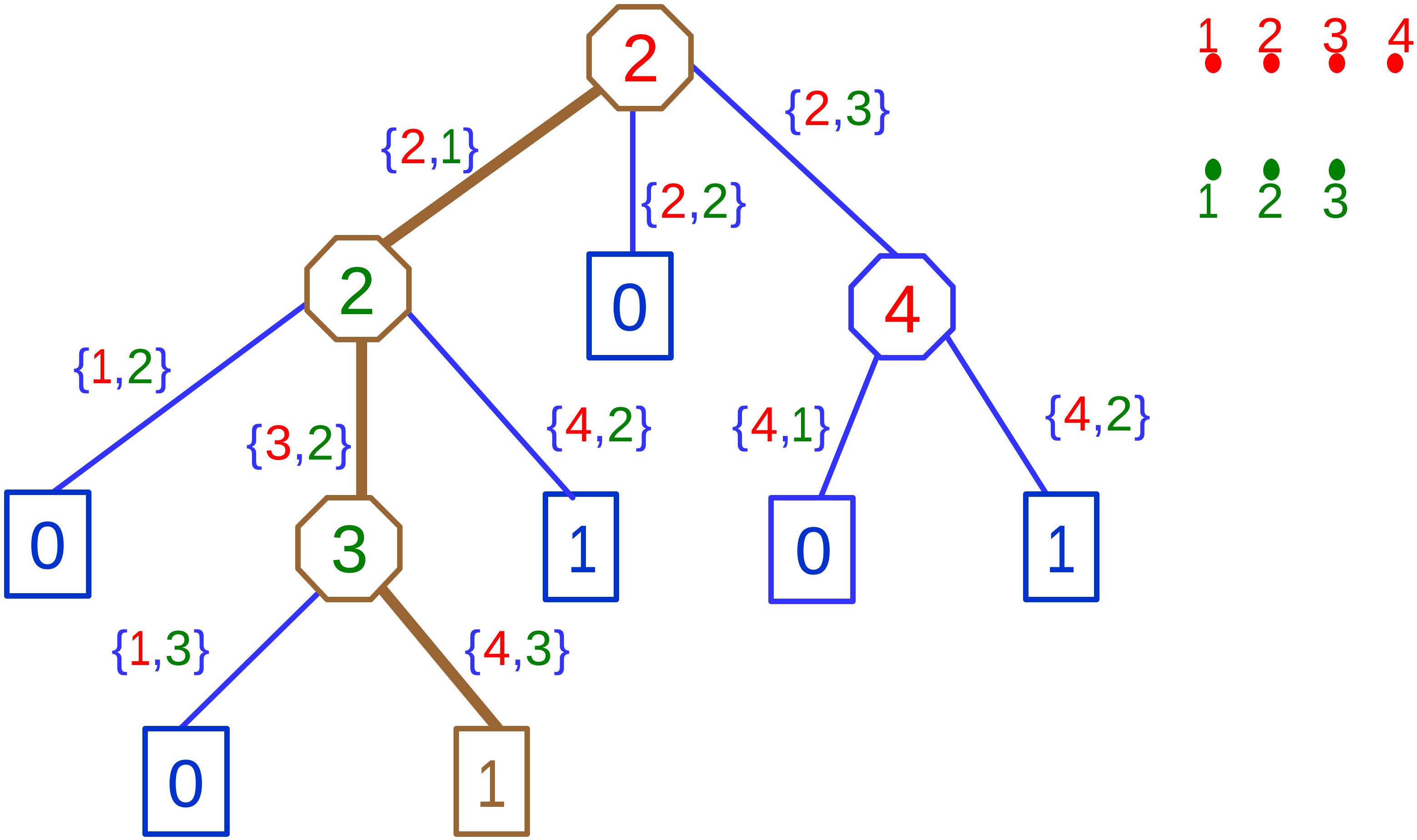
$$P_{21}P_{32}P_{43} \vee P_{21}P_{42} \vee P_{23}P_{42}$$

A matching decision tree



$$P_{21}P_{32}P_{43} \vee P_{21}P_{42} \vee P_{23}P_{42}$$

A matching decision tree



$$P_{21}P_{32}P_{43} \vee P_{21}P_{42} \vee P_{23}P_{42}$$

Associating matching trees with formulas

- **T(P$_{ij}$)** queries **i** & has height **1**
- **T(Øg)** is **T(g)** with leaf labels toggled
- To get tree for **h=(g$_1$Ú ... Úg$_t$)**
  - take DNF formula **F$_h$=T(g$_1$)Ú ... ÚT(g$_t$)**
  - do **canonical conversion** of **F$_h$** into a matching decision tree
    - **like conversion for ordinary decision trees**
      - **go term by term left-to-right simplifying future terms based on partial assignments**
      - **query both endpoints of every variable in each term**

Ideas for $PHP^{n+1 \to n}$ lower bound

- Restrictions are kind to matching decision trees
  - Analog of Hastad switching lemma for canonical conversion of DNF to matching decision trees
  - If proof is small trees can be made short
- Matching decision trees of height < n
  - for $PHP^{n+1 \to n}$ has all 0's on its leaves
  - for an axiom has all 1's on it leaves
  - preserve this property of all 1's on the leaves under inference rules

Extensions to extra axioms

- Same sorts of restrictions and decision trees
- Must also prove that extra axioms convert to trees with all 1's on their leaves
  - Surprisingly, this follows in each case from Nullstellensatz degree lower bounds for the extra axioms!

The Frontier