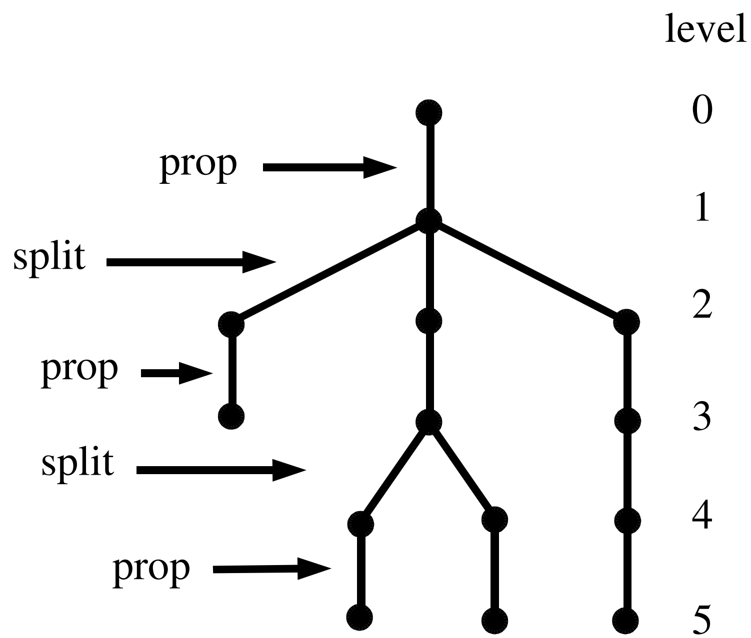
Search Trees

Apt's book p. 299-315

- Definition
- Labeling trees
- Complete labeling trees
- Reduced labeling trees
- *prop* labeling trees
- Sizes of the trees: an example

Consider a CSP $\mathcal{P}$ with a sequence of variables X. By a **search tree for** $\mathcal{P}$ we mean a finite tree such that

- its nodes are CSP's,
- its root is $\mathcal{P}$,
- the nodes at an even level have exactly one direct descendant,
- if $\mathcal{P}_1,\ldots,\mathcal{P}_m$ are direct descendants of $\mathcal{P}_0$, then the union of $\mathcal{P}_1,\ldots,\mathcal{P}_m$ is equivalent w.r.t. X to $\mathcal{P}_0$.



Labeling Trees

Intuition: labeling rule from Chapter 3.2

$$\frac{x \in \{a_1, \ldots, a_k\}}{x \in \{a_1\} \mid \ldots \mid x \in \{a_k\}}$$

Labeling trees are specific search trees for **finite** CSP’s.

- splitting consists of **labeling** of a domain of a variable
- constraint propagation consists of a domain reduction method

Complete Labeling Trees

Constrain propagation **absent**.

Given:

- a CSP $\mathcal{P}$ with non-empty domains,
- $x_1,\ldots,x_n$ the sequence of its variables linearly ordered by $\prec$.

Complete labeling tree associated with $\mathcal{P}$ and $\prec$: a tree such that

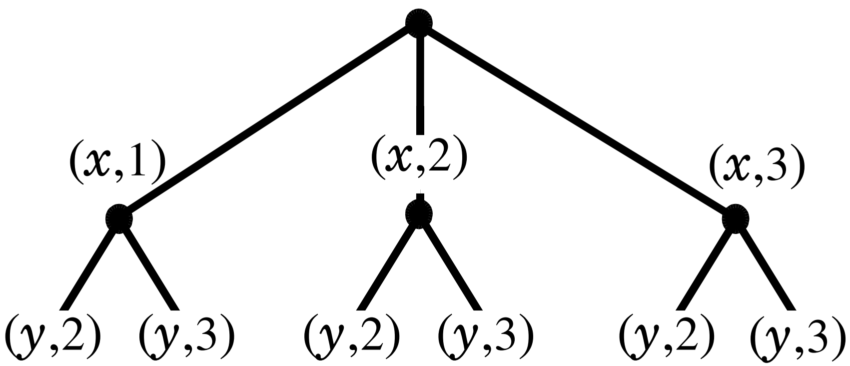
- the direct descendants of the root are of the form (x_1,d),
- the direct descendants of a node (x_j,d), where $j \in [1..n\text{-}1]$, are of the form (x_{j+1},e),
- its branches determine all the instantiations with the domain $\{x_1,\ldots,x_n\}$.

Examples

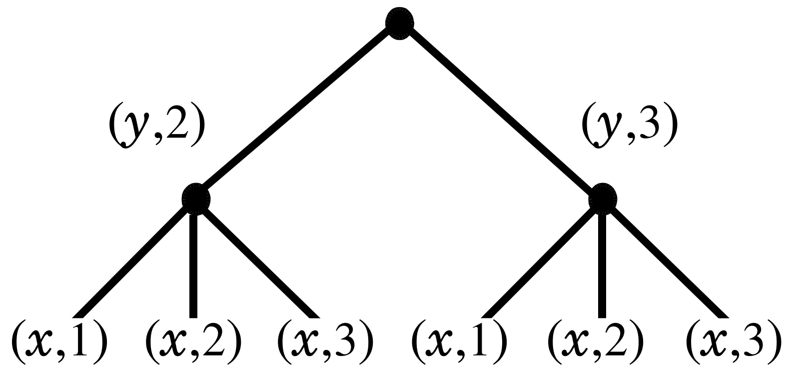
Complete labeling trees for

$\langle\, x < y;\, x \in \{1,2,3\},\, y \in \{2,3\} \,\rangle$

1. with the ordering $x \prec y$



2. with the ordering $y \prec x$



$x \prec y$: $1+3+3 \cdot 2 = 10$ nodes

$y \prec x$: $1+2+2 \cdot 3 = 9$ nodes

Number of nodes depends on ordering, number of leaves doesn't.

Sizes of Complete Labeling Trees

Given:

- a CSP $\mathcal{P}$ with non-empty domains,
- $x_1,...,x_n$ the sequence of its variables linearly ordered by $\prec$,
- $D_1,...,D_n$ the corresponding domains. Then

The number of nodes in the complete labeling tree associated with $\prec$ is

$$1 + \sum_{i=1}^{n} \left(\prod_{j=1}^{i} |D_j|\right),$$

$|A|$: the cardinality of the set A.

A complete labeling tree has the least number of nodes if the variables are ordered by their domain sizes in an **increasing** order.

Reduced Labeling Trees

An instantiation I is **along the ordering** $x_1,\ldots,x_n$ if its domain is $\{x_1,\ldots,x_j\}$ for some $j\in[1..n]$.

Given:

- a CSP $\mathcal{P}$ with non-empty domains,
- $x_1,\ldots,x_n$ the sequence of its variables linearly ordered by $\prec$.

Reduced labeling tree associated with $\mathcal{P}$ and $\prec$: a tree such that

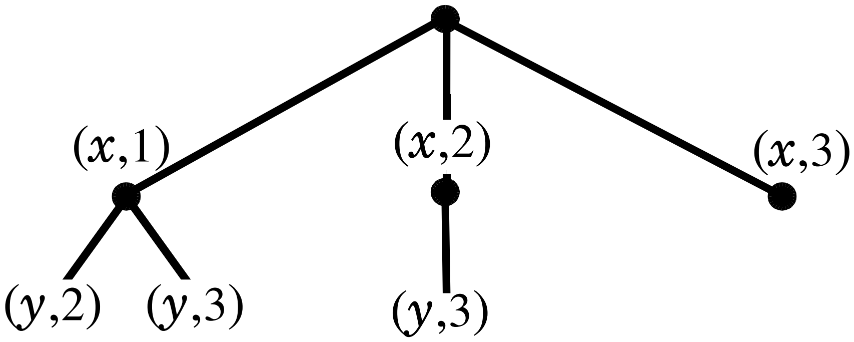
- the direct descendants of the root are of the form (x_1,d),
- the direct descendants of a node (x_j,d), where $j\in[1..n\text{-}1]$, are of the form (x_{j+1},e),
- its branches determine all consistent instantiations along the ordering $x_1,\ldots,x_n$.

Examples

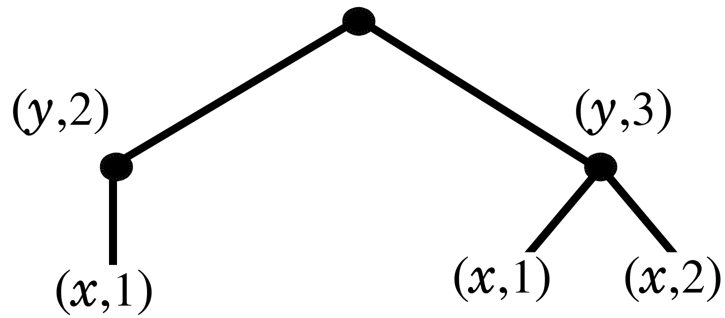
Reduced labeling trees for

$\langle\, x < y;\ x \in \{1,2,3\},\ y \in \{2,3\}\, \rangle$

1. with the ordering $x \prec y$



2. with the ordering $y \prec x$



Both the number of nodes **and the number of leaves** depends on ordering.

Labeling Trees with Constraint Propagation

Given:

$$\mathcal{P} := \langle \mathcal{C} \,;\, x_1 \in D_1, \ldots, x_n \in D_n \rangle$$

- Assume fixed form of **constraint propagation** *prop(i)* in the form of a domain reduction, where $i \in [0..n\text{-}1]$.
- i determines the sequence $x_{i+j}, \ldots, x_n$ of the variables to the domain of which *prop(i)* is applied.
- Given current variable domains $E_1, \ldots, E_n$, constraint propagation *prop(i)* transforms only $E_{i+j}, \ldots, E_n$.
- *prop(i)* depends on the original constraints $\mathcal{C}$ of $\mathcal{P}$ and on the domains $E_1, \ldots, E_i$.

prop Labeling Trees (1/2)

prop **labeling tree associated with** $\mathcal{P}$:

a tree such that

- its nodes are sequences of the domain expressions $x_1 \in E_1, \ldots, x_n \in E_n$,
- its root is $x_1 \in D_1, \ldots, x_n \in D_n$,
- each node at an **even** level is of the form

  $x_1 \in \{d_1\}, \ldots, x_i \in \{d_i\}, x_{i+1} \in E_{i+1}, \ldots, x_n \in E_n$.

  If $i=n$, this node is a leaf. Otherwise it has exactly one direct descendant, obtained using *prop(i)*:

  $x_1 \in \{d_1\}, \ldots, x_i \in \{d_i\}, x_{i+1} \in E'_{i+1}, \ldots, x_n \in E'_n$ where $E'_j \subseteq E_j$ for $j \in [i+1..n]$

(cont.)

prop Labeling Trees (2/2)

(cont.)

- each node at an **odd** level is of the form

  $x_1 \in \{d_1\}, \ldots, x_i \in \{d_i\}, x_{i+1} \in E_{i+1}, \ldots, x_n \in E_n$.

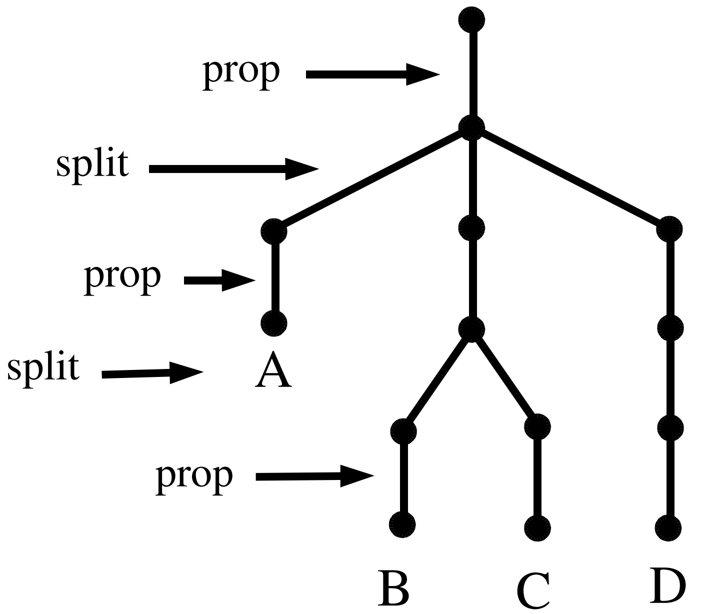
  If $E_j = \emptyset$ for some $j \in [i+1..n]$, this node is a leaf. Otherwise it has direct descendants of the form

  $x_1 \in \{d_1\}, \ldots, x_i \in \{d_i\}, x_{i+1} \in \{d\},$
  $x_{i+2} \in E_{i+2}, \ldots, x_n \in E_n,$

  for all $d \in E_{i+1}$ such that the instantiation $\{(x_1, d_1), \ldots, (x_i, d_i),$ $(x_{i+1}, d_{i+1})\}$ is consistent.

Example of a *prop* labeling tree

Consider a CSP with two variables with the order $x_1 \prec x_2$.



level	past variables	current variable	future variables
0			x_1, x_2
1		x_1	x_2
2		x_1	x_2
3	x_1	x_2	
4	x_1	x_2	
5	x_1, x_2		

A is a **failed** node and B,C and D are **success** nodes.

Example: SEND + MORE = MONEY (1/2)

SEND	$S, M \in [1..9]$
+MORE	$E, N, D, O, R, Y \in [0..9]$
MONEY	`all_different`(S,E,N,D,M,O,R,Y)

- Complete labeling tree
  Total number of leaves: $9^2 \cdot 10^6 = 81000000$.

- Reduced labeling tree
  Total number of leaves:
  $10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 - 2 \cdot (9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4) = 483840$.
  Gain: 99.4% with respect to the complete labeling tree.

Example: SEND + MORE = MONEY (2/2)

- *prop* labeling tree

  Using as *prop(i)* the domain reduction rules for linear constraints over integer intervals from Chapter 6. Initial application of *prop(i)* reduces the domains to S=9, E∈[4..7], N∈[5..8], D,R,Y∈ [2..8], M=10, O=0
  Except for E, the application of *prop* reduces the domain of each variable to a singleton set before it is split.

  Total number of leaves: 4.