A Methodology for Coupling Fragments of XPath with Structural Indexes for XML Documents

George Fletcher, Dirk Van Gucht, Yuqing Wu, Marc Gyssens, Sofia Brenes, & Jan Paredaens

DBPL 2007
Vienna, Austria

Indices for XML Data

* XPath: expressions specify patterns
* Expression evaluation is aided by indices:
  - value-based: consider node values
  - structure-based: values & document structure

What about techniques for using structural indices in query evaluation?

(1) For which fragments of XPath are particular structural indices ideally suited?

e.g., in the relational world, range queries and B-trees

(2) For these fragments, how are its expressions optimally evaluated with the index?

(3) Can the answers to (1) & (2) be bootstrapped to provide general techniques for evaluation of arbitrary XPath expressions with indices?

In this paper ...

* Develop **general framework and methodology** for investigating pairings of query languages and structural indexes
* **Illustrate this methodology** on important special case of XPath and A(k)/P(k) indexes

Let's focus on (1):

For which class of XPath expressions are the P(k) partitions ideally suited?

A(k) Indices: Localized Bisimilarity

* 1-Index/Dataguide much too fined-grained, and hence too large for practical use ...
* Kaushik et al (ICDE '02) proposed restricting 1-index to "k-neighborhood"
* Substantially smaller than 1-index/ DataGuide
* Distinguishes nodes by labels and incoming paths of length k

Data Model

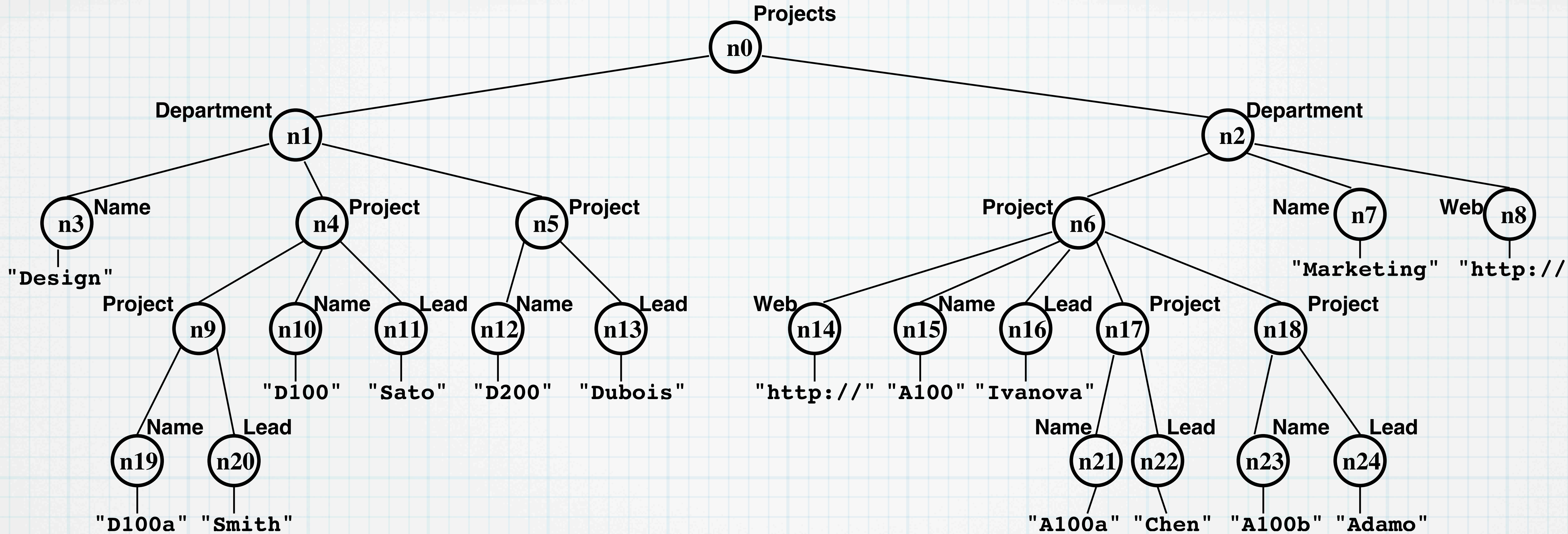
$$D = (V, Ed, r, \lambda)$$

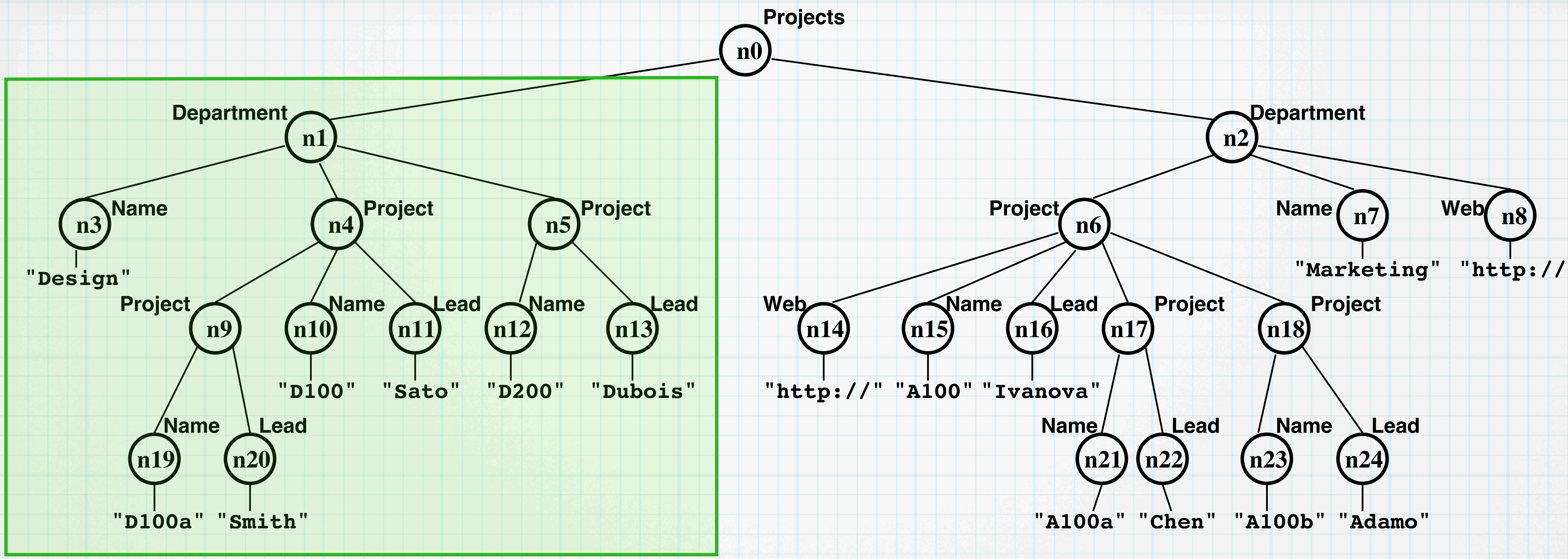
* Documents are finite unordered node-labeled trees:
  - nodes V
  - edges $Ed \subseteq V \times V$
  - root $r \in V$
  - labels $\lambda : V \to \mathcal{L}$

The A(k) Partition of a Document

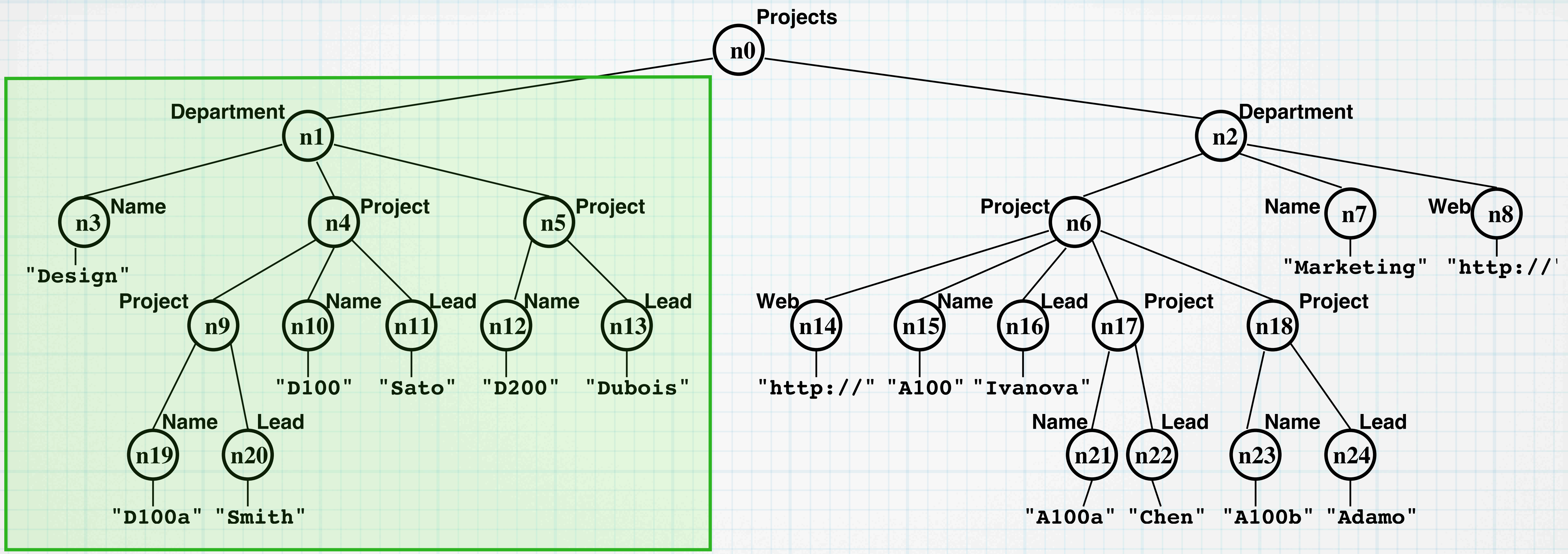
$$n_1 \equiv_{A(k)} n_2$$

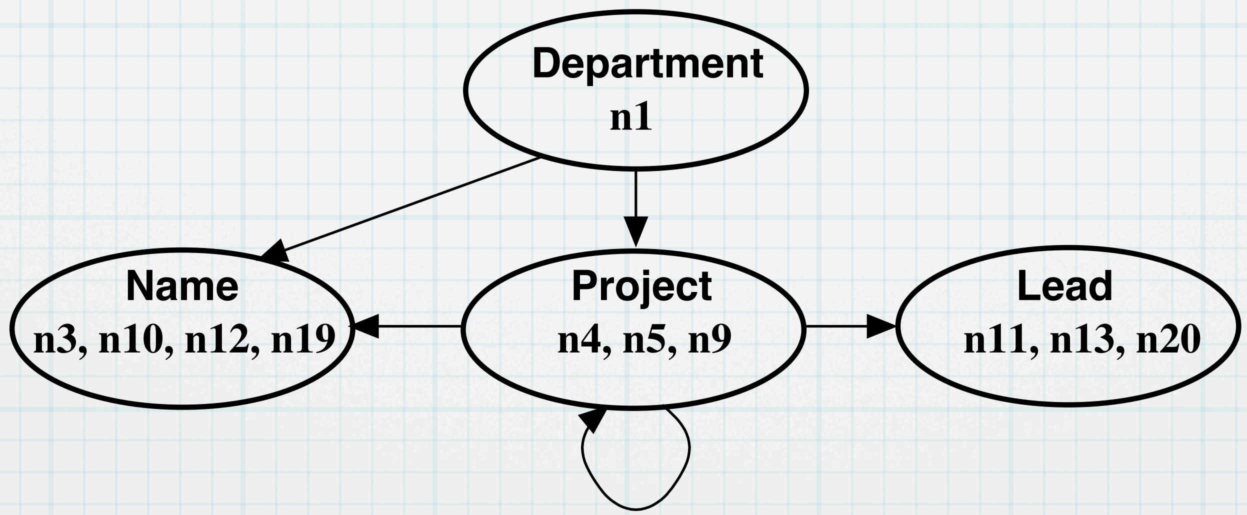
* For nodes n1 and n2, we have that they are A(k)-equivalent if
    - they have the same label, and
    - for k>0, if one has a parent, so does the other and, furthermore, their parents are A(k-1)-equivalent
* The partition induced by this relation on nodes is called the A(k) partition of the document


Projects
n0
Department
n1
Department
n2
n3
Name
"Design"
n4
Project
n5
Project
Project
n6
Name
n7
"Marketing"
Web
n8
"http://'
Project
n9
n10
Name
"D100"
n11
Lead
"Sato"
n12
Name
"D200"
n13
Lead
"Dubois"
Web
n14
"http://"
n15
Name
"A100"
n16
Lead
"Ivanova"
n17
Project
n18
Project
Name
n19
"D100a"
Lead
n20
"Smith"
Name
n21
"A100a"
Lead
n22
"Chen"
Name
n23
"A100b"
Lead
n24
"Adamo"

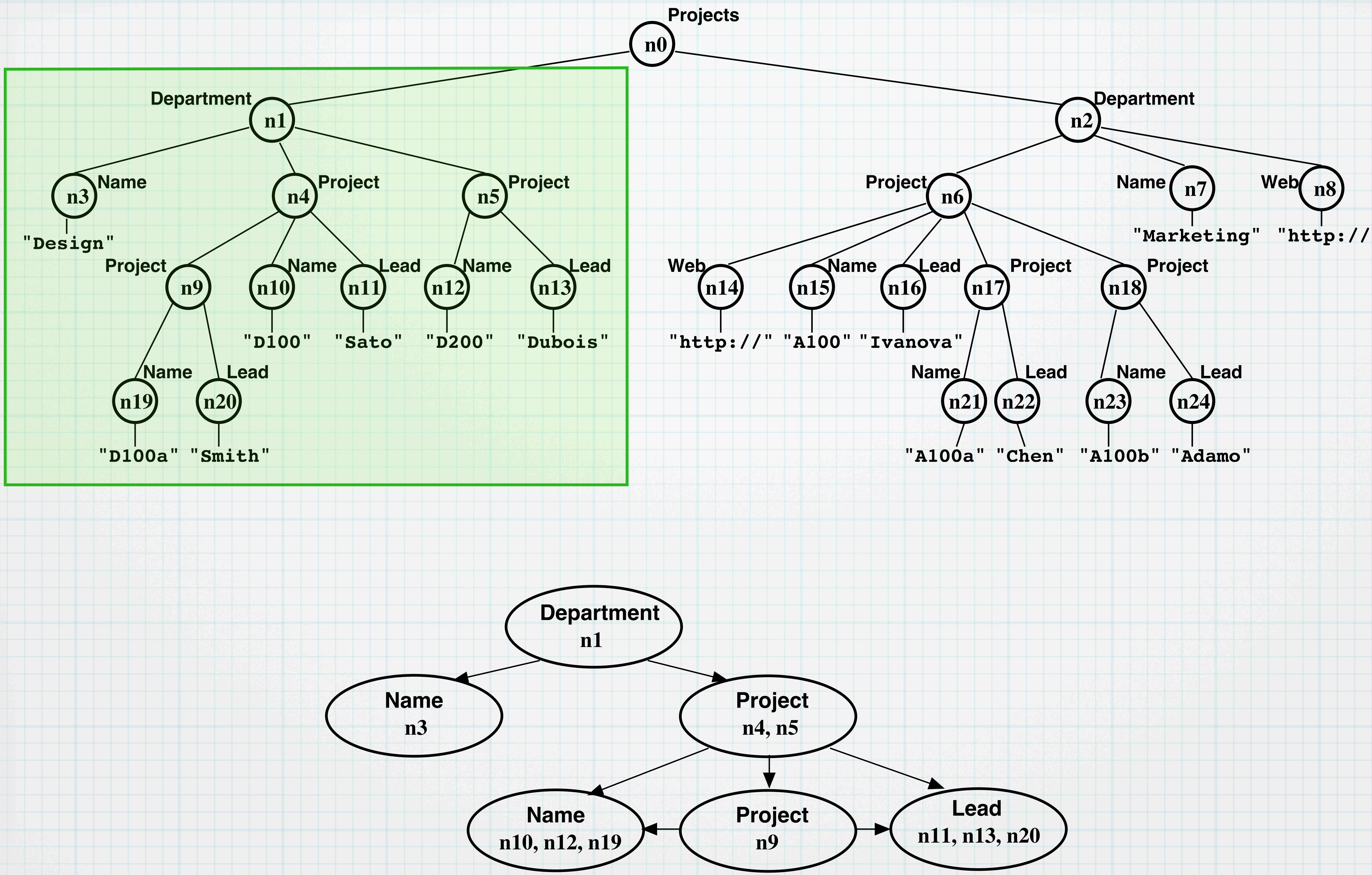


Consider A(k) indices on the "Design" department subtree

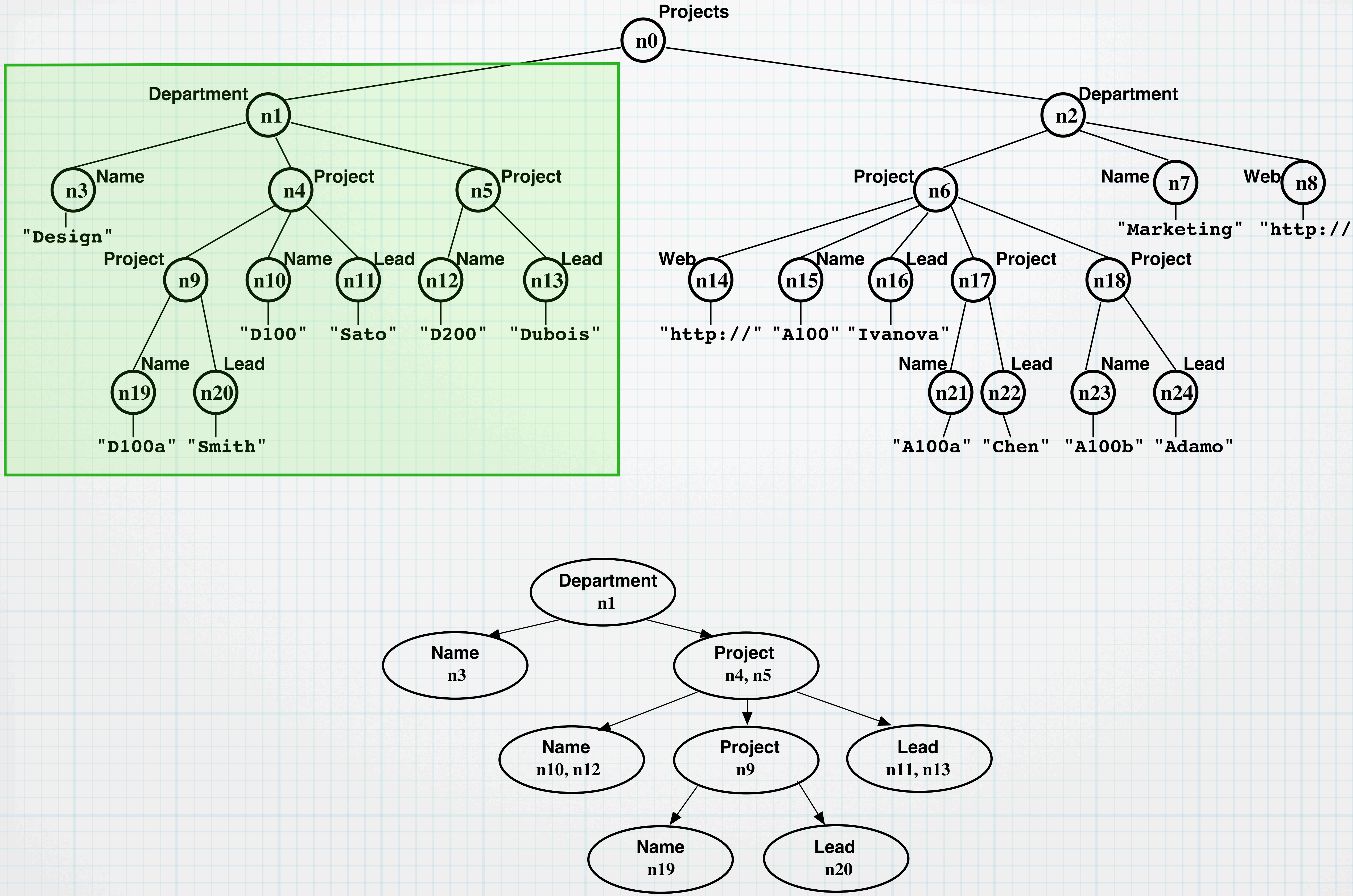




A(0) index on "Design" department subtree



A(1) index on "Design" department subtree



A(2) index on "Design" department subtree

The P(k) Partition of a Document

$$(n_1, m_1) \equiv_{P(k)} (n_2, m_2)$$

* For nodes n1, m1, n2, and m2 we have that (n1, m1) and (n2, m2) are P(k)-equivalent if
  - (n1, m1) and (n2, m2) are in UpPaths(D,k)
  - the distance from n1 to m1 in the document is the same as that from n2 to m2, and
  - $n_1 \equiv_{A(k)} n_2$
* The partition induced by this relation on node pairs in UpPaths(D, k) is called the P(k) partition of the document

XPath Algebra

$$D = (V, Ed, r, \lambda)$$

$$\begin{aligned}
\varepsilon(D) &= \{(m,m) \mid m \in V\} \\
\emptyset(D) &= \emptyset \\
\downarrow(D) &= Ed \\
\uparrow(D) &= Ed^{-1} \\
\ell(D) &= \{(m,m) \mid m \in V \text{ and } \lambda(m) = \ell\}
\end{aligned}$$

XPath Algebra

$$D = (V, Ed, r, \lambda)$$

$$
\begin{array}{rcl}
\varepsilon(D) & = & \{(m,m) \mid m \in V\} \\
\emptyset(D) & = & \emptyset \\
\downarrow(D) & = & Ed \\
\uparrow(D) & = & Ed^{-1} \\
\ell(D) & = & \{(m,m) \mid m \in V \text{ and } \lambda(m) = \ell\} \\
E_1 \cup E_2(D) & = & E_1(D) \cup E_2(D) \\
E_1 \cap E_2(D) & = & E_1(D) \cap E_2(D) \\
E_1 - E_2(D) & = & E_1(D) - E_2(D) \\
E_1 \circ E_2(D) & = & \{(m,n) \mid \exists w \colon (m,w) \in E_1(D) \,\&\, (w,n) \in E_2(D)\} \\
E_1[E_2](D) & = & \{(m,n) \in E_1(D) | \; \exists w \colon (n,w) \in E_2(D)\}.
\end{array}
$$

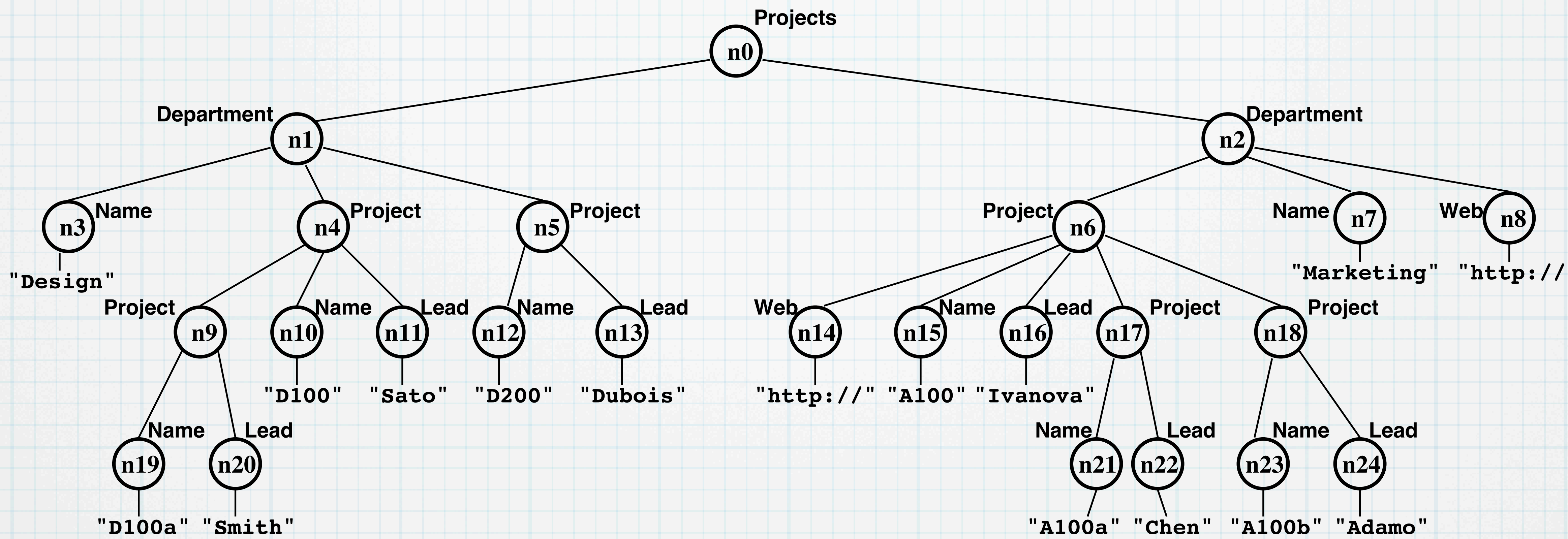
XPath Algebra

$$D = (V, Ed, r, \lambda)$$

* Global semantics of expressions: binary relation over V
* Local semantics of expressions: for $m \in V$

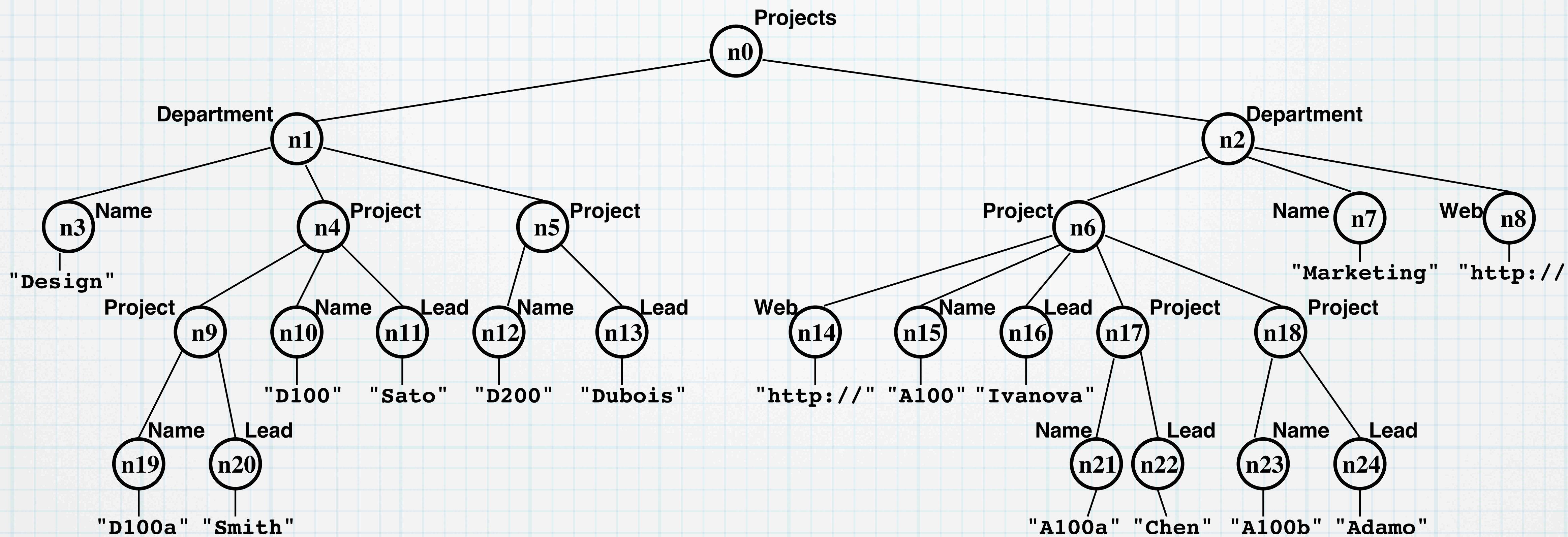
$$E(D)(m) = \{n \in V \mid (m, n) \in E(D)\}$$

XPath Algebra



"Retrieve all department names"

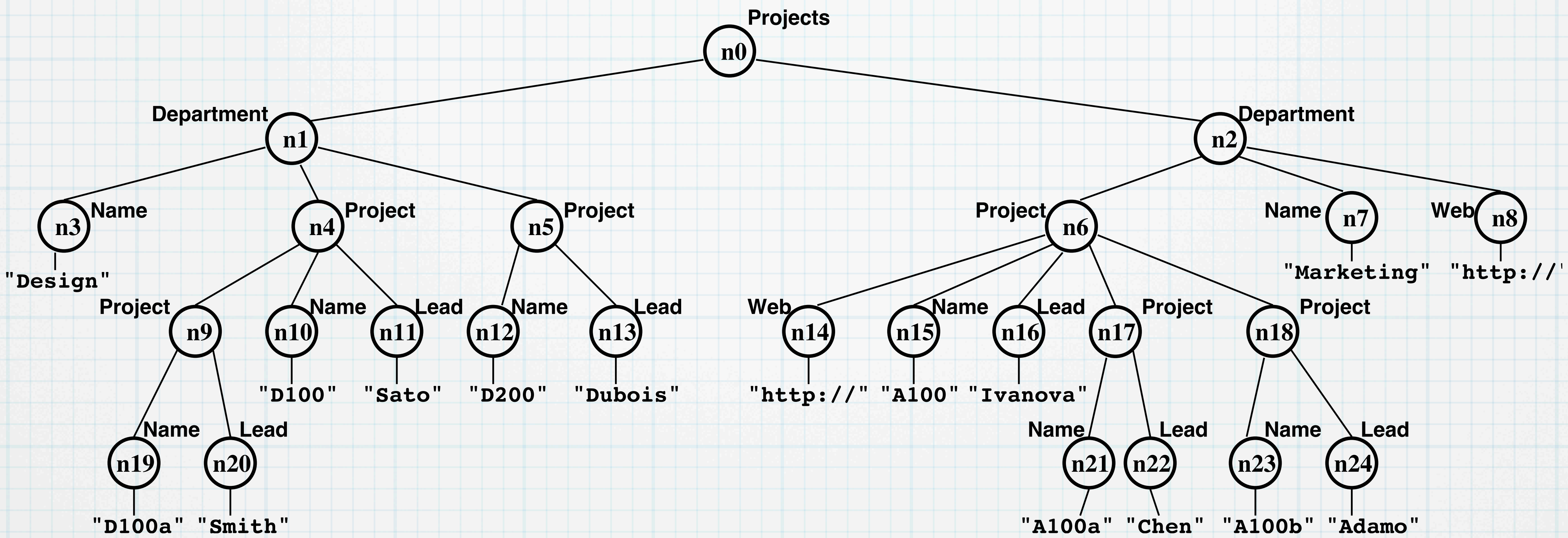
XPath Algebra



"Retrieve all department names"

$$E = \texttt{Projects} \circ \downarrow \circ \texttt{Department} \circ \downarrow \circ \texttt{Name}$$

XPath Algebra

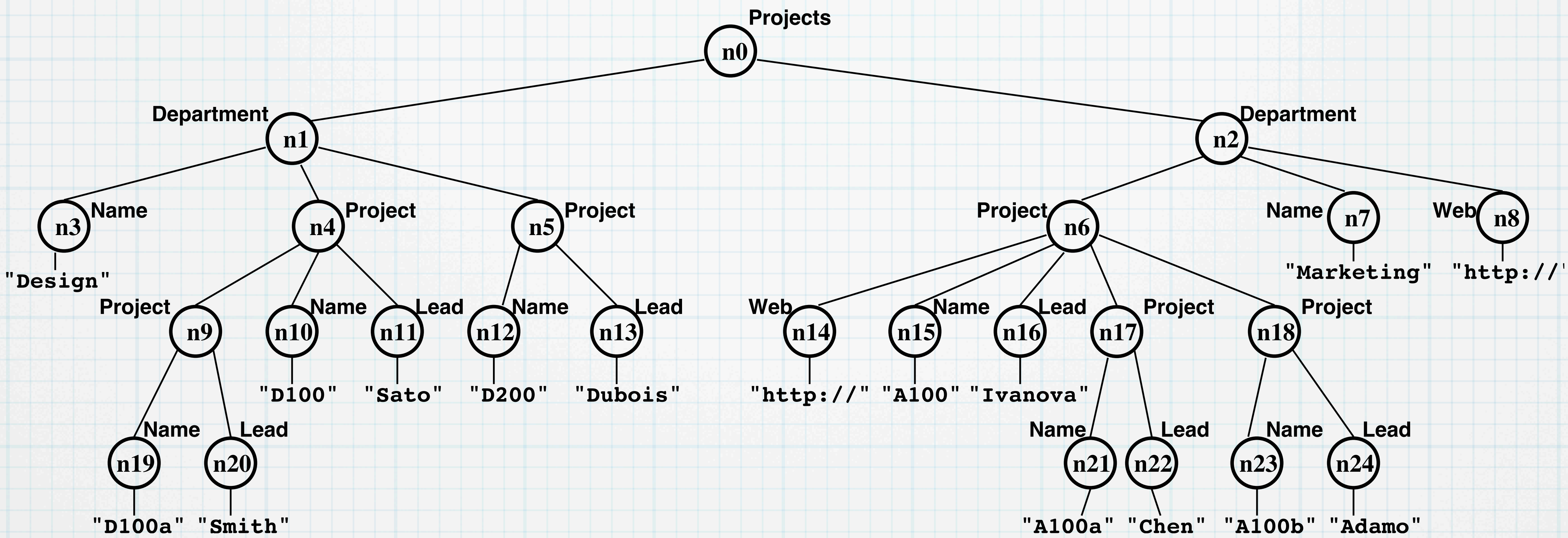


"Retrieve all department names"

$$E = \mathtt{Projects} \circ \downarrow \circ \mathtt{Department} \circ \downarrow \circ \mathtt{Name}$$

$$E(D) = \{(n_0, n_3), (n_0, n_7)\}$$

XPath Algebra



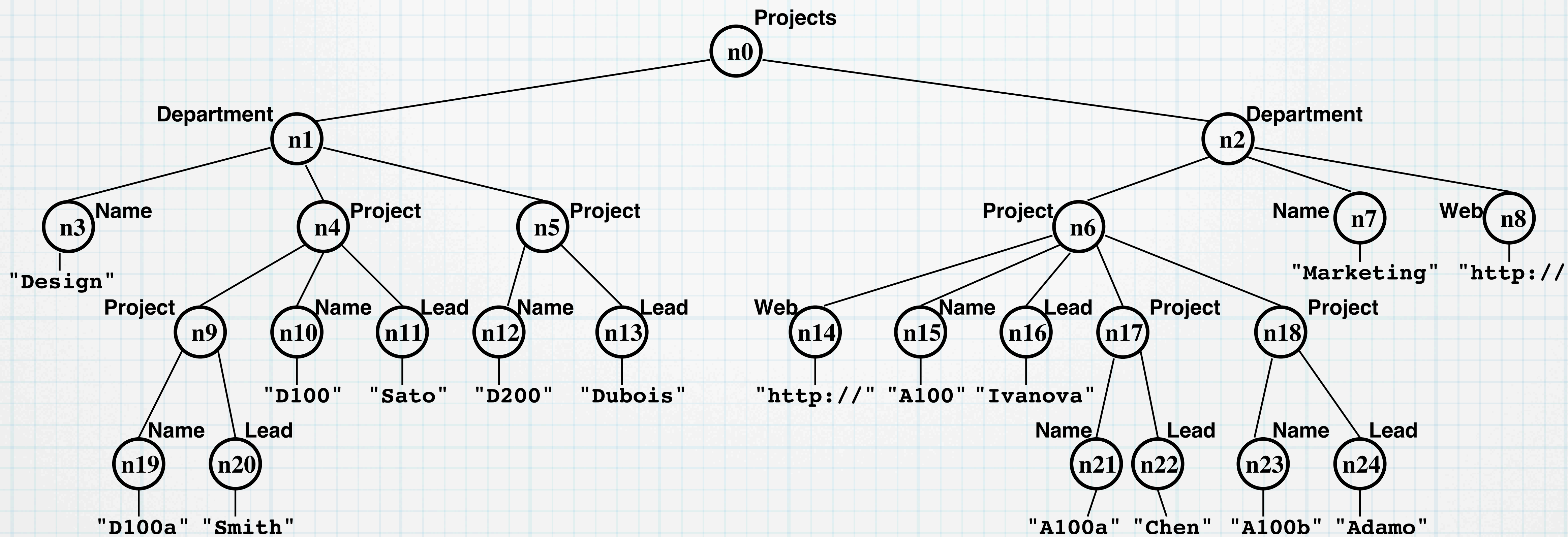
"Retrieve all department names"

$$E = \texttt{Projects} \circ \downarrow \circ \texttt{Department} \circ \downarrow \circ \texttt{Name}$$

$$E(D) = \{(n_0, n_3), (n_0, n_7)\}$$

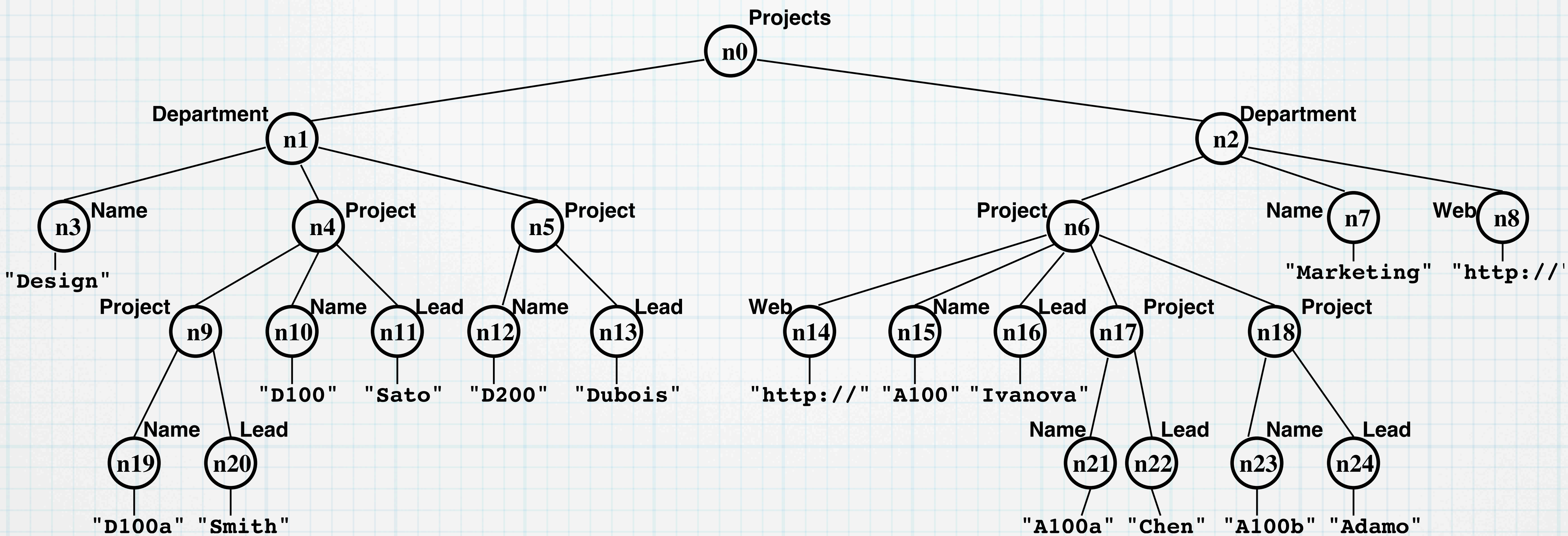
$$E(D)(n_0) = \{n_3, n_7\}$$

XPath Algebra



"Retrieve all projects which are sub-projects of projects with a website"

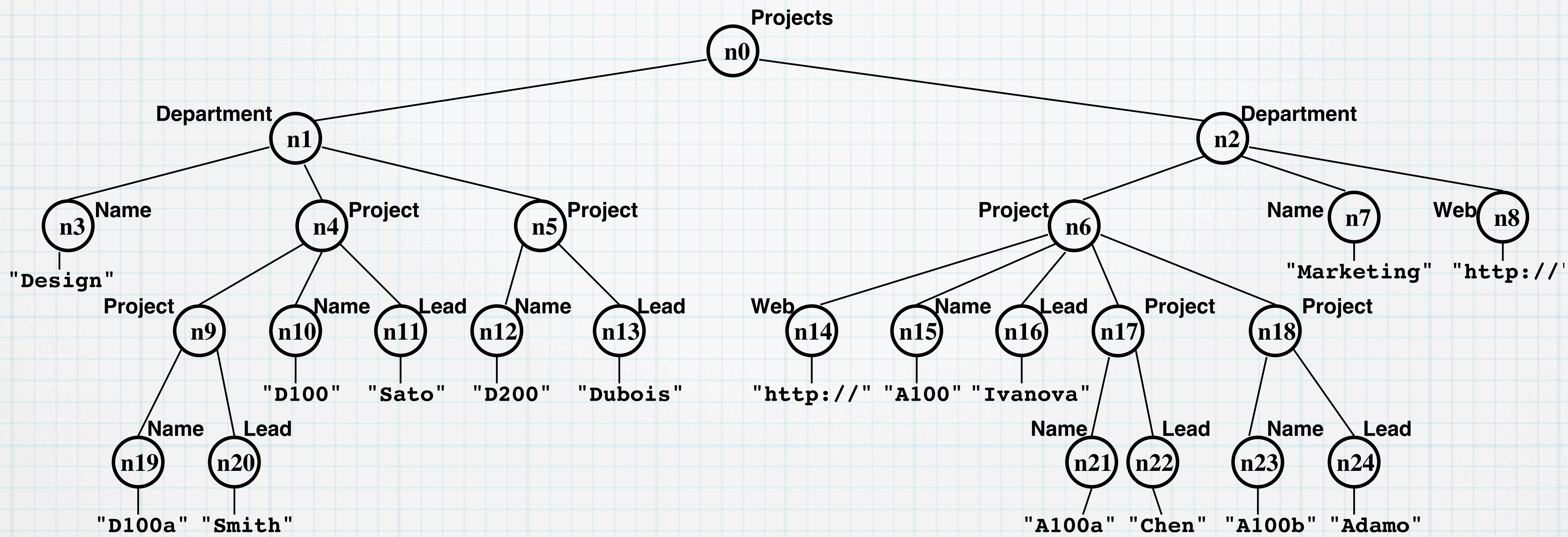
XPath Algebra



"Retrieve all projects which are sub-projects of projects with a website"

$$E = \texttt{Project}[\uparrow \circ \texttt{Project} \circ \downarrow \circ \texttt{Web}]$$

XPath Algebra



"Retrieve all projects which are sub-projects of projects with a website"

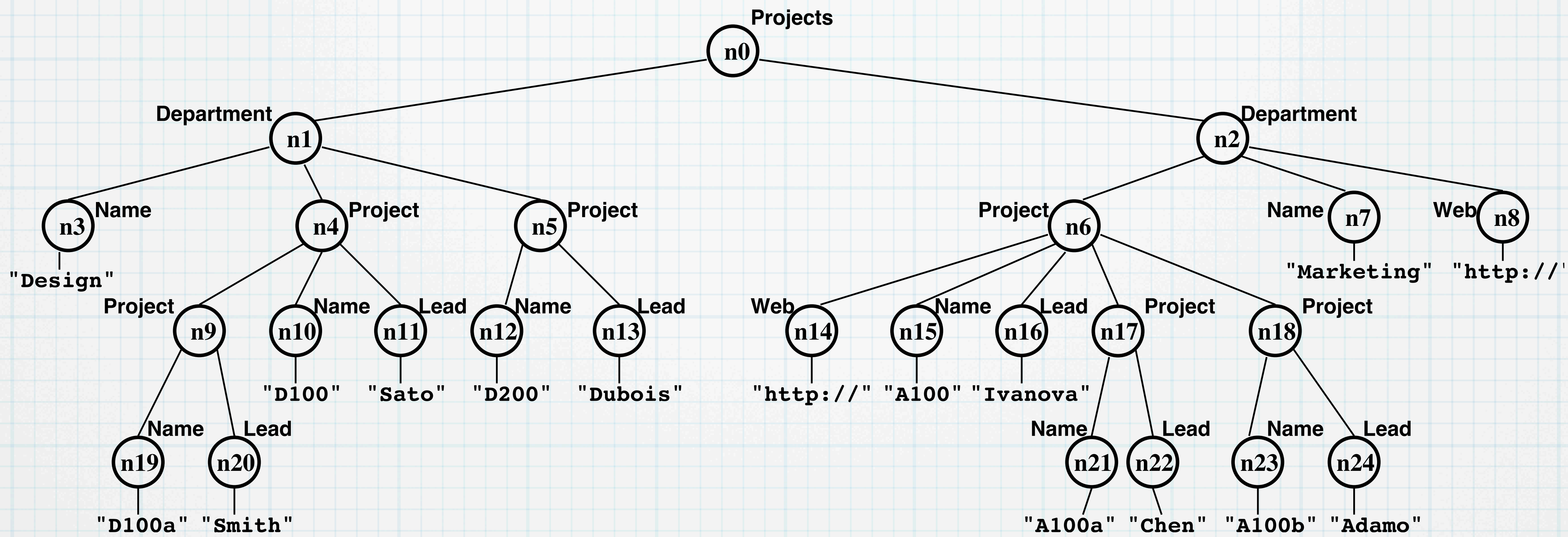
$$E = \texttt{Project}[\uparrow \circ \texttt{Project} \circ \downarrow \circ \texttt{Web}]$$

$$E(D) = \{(n_{17}, n_{17}), (n_{18}, n_{18})\}$$

Upward-k Algebras

Upward-k Algebras: for k >= 0, U(k) is the fragment of the XPath-Algebra with expressions that do not use the ↓ primitive and have at most k uses of the ↑ primitive in a "path"

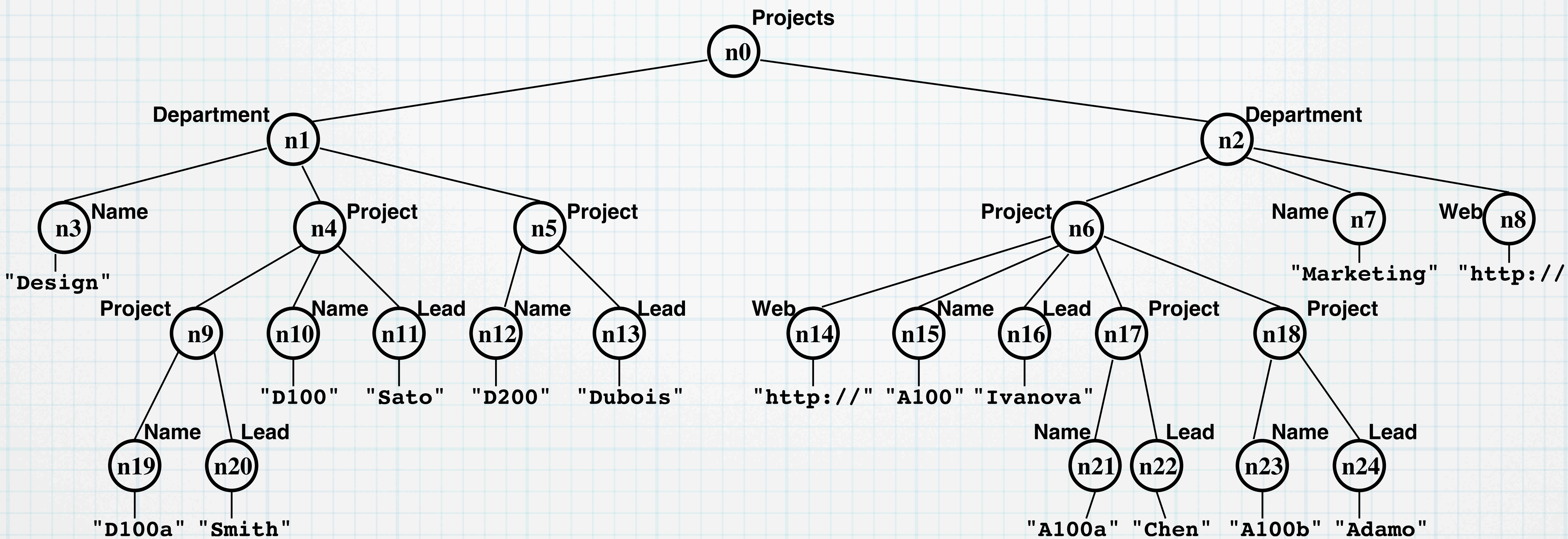
Upward-k Algebras



"Retrieve sub-project leaders"

$$E = \texttt{Lead}[\uparrow \circ \texttt{Project} \circ \uparrow \circ \texttt{Project}]$$

Upward-k Algebras



"Retrieve sub-project leaders"

$$E = \texttt{Lead}[\uparrow \circ \texttt{Project} \circ \uparrow \circ \texttt{Project}]$$

In U(2) but not in U(1)!

Language Indistinguishability

$$(n_1, m_1) \equiv_{\mathcal{F}} (n_2, m_2)$$

For fragment $\mathcal{F}$ of the XPath algebra, we say node pairs (n1, m1) and (n2, m2) are indistinguishable by $\mathcal{F}$ if for any expression E in $\mathcal{F}$, it is the case that $(n_1, m_1) \in E(D) \iff (n_2, m_2) \in E(D)$

Coupling P(k) and U(k) Indistinguishability

Coupling Theorem:

Let D be a document and $k \in \mathbb{N}$. The $P(k)$-partition of D and the $U(k)$-partition of D are the same.

Proof:

* P(k) indistinguishability implies U(k) indistinguishability, via induction on U(k) expressions
* P(k) distinguishability implies U(k) distinguishability, via construction of partition-block labeling expressions

Coupling P(k) and U(k) Indistinguishability

Block-Union Theorem:

Let D be a document, $k \in \mathbb{N}$, and $E \in U(k)$. Then there exists a class $\mathfrak{B}_E$ of partition blocks of the $P(k)$-partition of D such that $E(D) = \bigcup_{B \in \mathfrak{B}_E} B$.

... follows directly from Coupling Theorem

These results provide a precise linguistic characterization of A(k) and P(k) partitions, in answer to question (1).

Now let's consider question (2):

How are U(k) expressions to be evaluated with the help of P(k) partitions?

Upward Algebra Eval

* for expressions in U(k), direct look-up in P(k) index
* for expressions in U(l), l > k, then by decomposition into U(k) sub-expressions and joining sub-results

Finally, let's consider question (3):

Can these results be bootstrapped to provide general techniques for evaluation of full XPath?

XPath Algebra Eval

* Via predicate elimination and inversion of remaining "downward" subexpressions into Upward-Algebra subexpressions:

$$
\begin{array}{c}
E \rightarrow E^{-1} \\
\hline
\epsilon \rightarrow \epsilon \\
\emptyset \rightarrow \emptyset \\
\downarrow \rightarrow \uparrow \\
\hat{\lambda} \rightarrow \hat{\lambda} \\
E_1 \cup E_2 \rightarrow E_1^{-1} \cup E_2^{-1} \\
E_1 \cap E_2 \rightarrow E_1^{-1} \cap E_2^{-1} \\
E_1 - E_2 \rightarrow E_1^{-1} - E_2^{-1} \\
E_1 \diamond E_2 \rightarrow E_2^{-1} \diamond E_1^{-1}.
\end{array}
$$

* then proceed as before with Upward-Algebra eval

XPath Algebra Eval

Suppose we have a document D, the P(2) partition of D, and the query $\downarrow[\downarrow]$

XPath Algebra Eval

Suppose we have a document D, the P(2) partition of D, and the query $\downarrow[\downarrow]$

then ... $\downarrow[\downarrow](D)$

XPath Algebra Eval

Suppose we have a document D, the P(2) partition of D, and the query $\downarrow [\downarrow]$

then ...

$$\downarrow [\downarrow](D)$$

$$\downarrow$$

$$\downarrow \diamond \downarrow \diamond \uparrow (D)$$

XPath Algebra Eval

Suppose we have a document D, the P(2) partition of D, and the query $\downarrow[\downarrow]$

then ...

$$
\begin{tikzcd}
& \downarrow[\downarrow](D) \arrow[d] & \\
& \downarrow \diamond \downarrow \diamond \uparrow (D) \arrow[dl] \arrow[dr] & \\
\downarrow \diamond \downarrow (D) & & \uparrow (D)
\end{tikzcd}
$$

XPath Algebra Eval

Suppose we have a document D, the P(2) partition of D, and the query $\downarrow[\downarrow]$

then ...

$$
\begin{tikzcd}
& \downarrow[\downarrow](D) \arrow[d] & \\
& \downarrow \diamond \downarrow \diamond \uparrow (D) \arrow[dl] \arrow[dr] & \\
\downarrow \diamond \downarrow (D) \arrow[d] & & \uparrow (D) \\
((\downarrow \diamond \downarrow)^{-1}(D))^{-1} & &
\end{tikzcd}
$$

XPath Algebra Eval

Suppose we have a document D, the P(2) partition of D, and the query $\downarrow[\downarrow]$

then ...

$$\downarrow[\downarrow](D)$$

$$\downarrow \diamond \downarrow \diamond \uparrow (D)$$

$$\downarrow \diamond \downarrow (D) \qquad \uparrow (D)$$

$$((\downarrow \diamond \downarrow)^{-1}(D))^{-1}$$

$$(\uparrow \diamond \uparrow (D))^{-1}$$

XPath Algebra Eval

Suppose we have a document D, the P(2) partition of D, and the query $\downarrow[\downarrow]$

then ...

$$\begin{tikzcd}
 & \downarrow[\downarrow](D) \arrow[d] & \\
 & \downarrow \diamond \downarrow \diamond \uparrow (D) \arrow[dl] \arrow[dr] & \\
\downarrow \diamond \downarrow (D) \arrow[d] & & \uparrow (D) \arrow[dd] \\
((\downarrow \diamond \downarrow)^{-1}(D))^{-1} \arrow[d] & & \\
(\uparrow \diamond \uparrow (D))^{-1} \arrow[rr] & & \boxed{(\uparrow \diamond \uparrow (D))^{-1} \bowtie \uparrow (D)}
\end{tikzcd}$$

... and this can be evaluated directly with the P(2) parition

Research Directions

* Currently developing new data structures leveraging these results
  - develop fast P(k) partition block look-up algorithms
* Further study of query decomposition and inversion algorithms
* Study workload driven index creation
* Study localized branching-path queries and develop appropriate index structures

Thanks!

Questions?