Static enforceability of XPath-based access control policies

James Cheney
University of Edinburgh

DBPL 2013
August 30, 2013

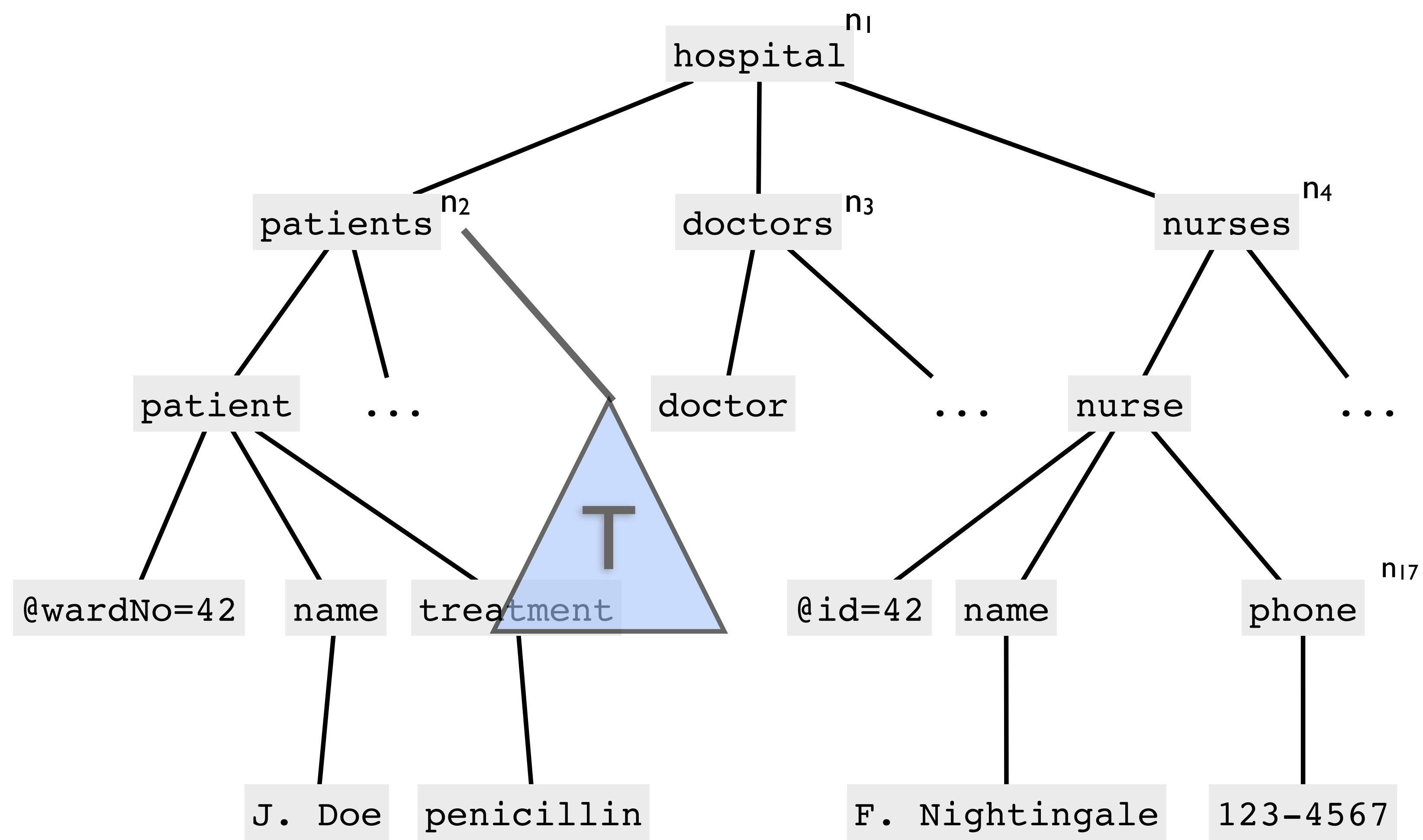
Background

- Access control for XML databases
- Read-only
  - security views [Stoica & Farkas 2002, Fan et al. 2004]
  - filtering [Luo et al. 2004]
  - annotations [Yu et al 2004, ...]
  - static analysis [Murata et al 2006]
- Access control in presence of updates: less studied
  - annotations [Koromilas et al. 2009]
  - schema-based [Bravo et al. 2007, 2012]

What about updates?

- Security views
  - require solving view update problems
- Dynamic enforcement
  - by filtering - inappropriate for updates (unpredictable)
  - by annotations - checks fast but updates require maintaining annotations
  - by queries - no annotations, but expensive checks
- Static enforcement
  - no dependence on data, but incomplete

Dynamic



insert(n_2,T)?

Allowed

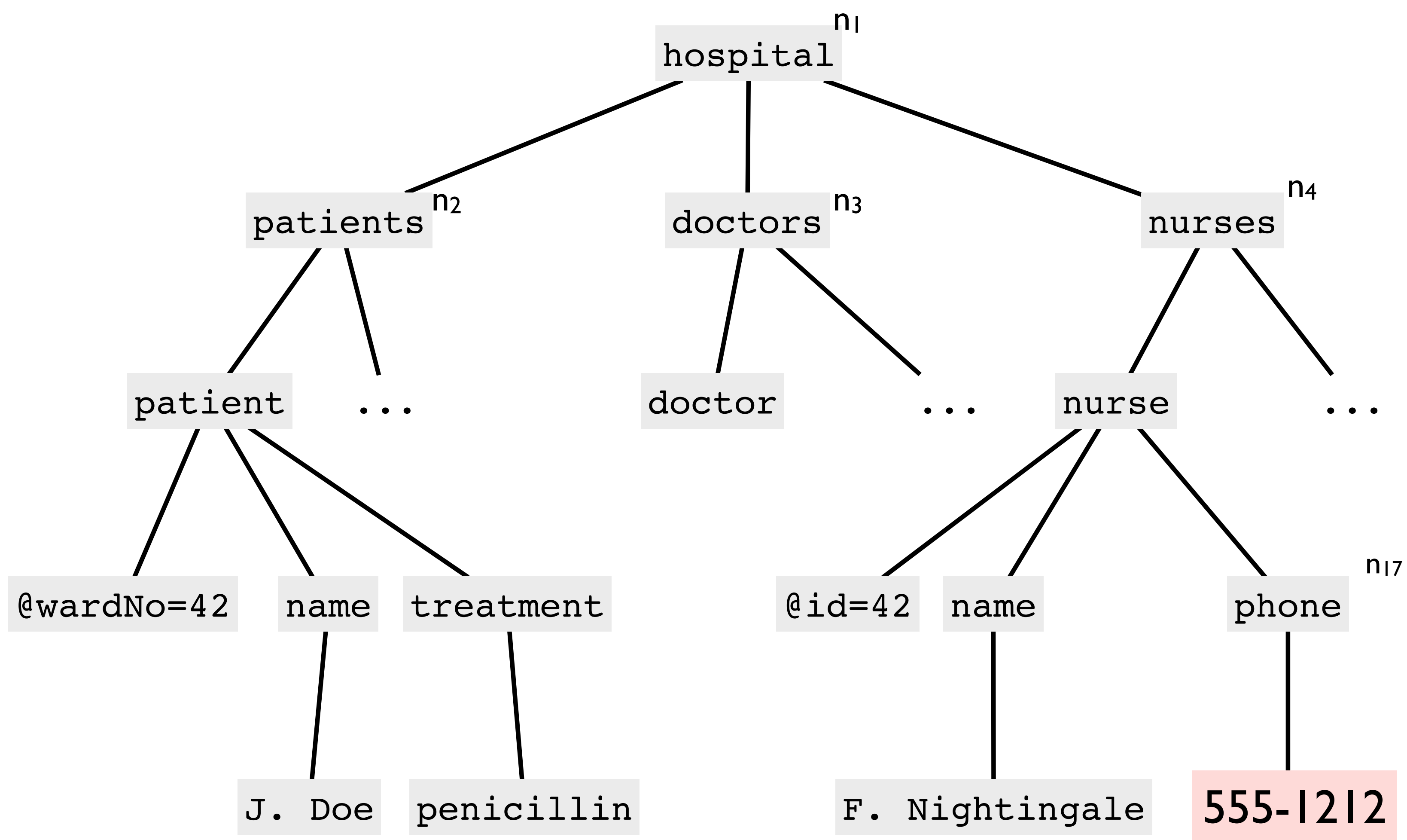
$Nurse(\$wn, \$uid)$:

R_1 : $+insert(//\texttt{patient}//*, *)$
R_2 : $+update(//\texttt{patient}[\texttt{@wardNo} = \$wn]/*, *)$
R_3 : $+update(//\texttt{nurse}[\texttt{@id} = \$uid]/\texttt{phone}/*, \texttt{text}())$
R_4 : $-insert(//*, \texttt{treatment})$
R_5 : $-update(//\texttt{treatment}, *)$

matches R_1

does not match R_4,R_5

Dynamic



update(n17,...)?

Allowed

Nurse($\$wn$, $\$uid$):

R_1 : $+insert(//\texttt{patient}//*, *)$

R_2 : $+update(//\texttt{patient}[@\texttt{wardNo} = \$wn]/*, *)$

R_3 : $+update(//\texttt{nurse}[@\texttt{id} = \$uid]/\texttt{phone}/*, \texttt{text}())$

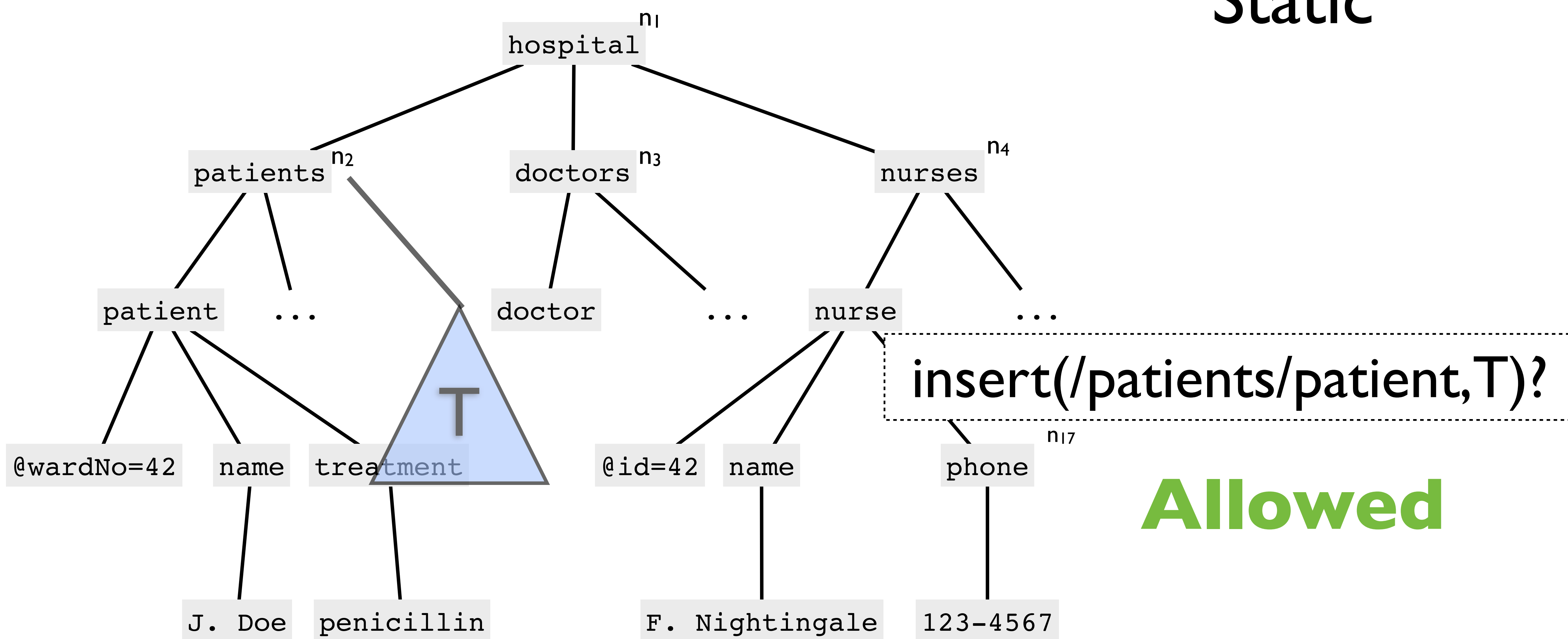
R_4 : $-insert(//*, \texttt{treatment})$

R_5 : $-update(//\texttt{treatment}, *)$

matches R_3

does not match R_4,R_5

Static



Allowed

Nurse($wn, $uid):

R_1 : $+insert(//\texttt{patient}//*, *)$

R_2 : $+update(//\texttt{patient}[@\texttt{wardNo} = \$wn]/*, *)$

R_3 : $+update(//\texttt{nurse}[@\texttt{id} = \$uid]/\texttt{phone}/*, \texttt{text}())$

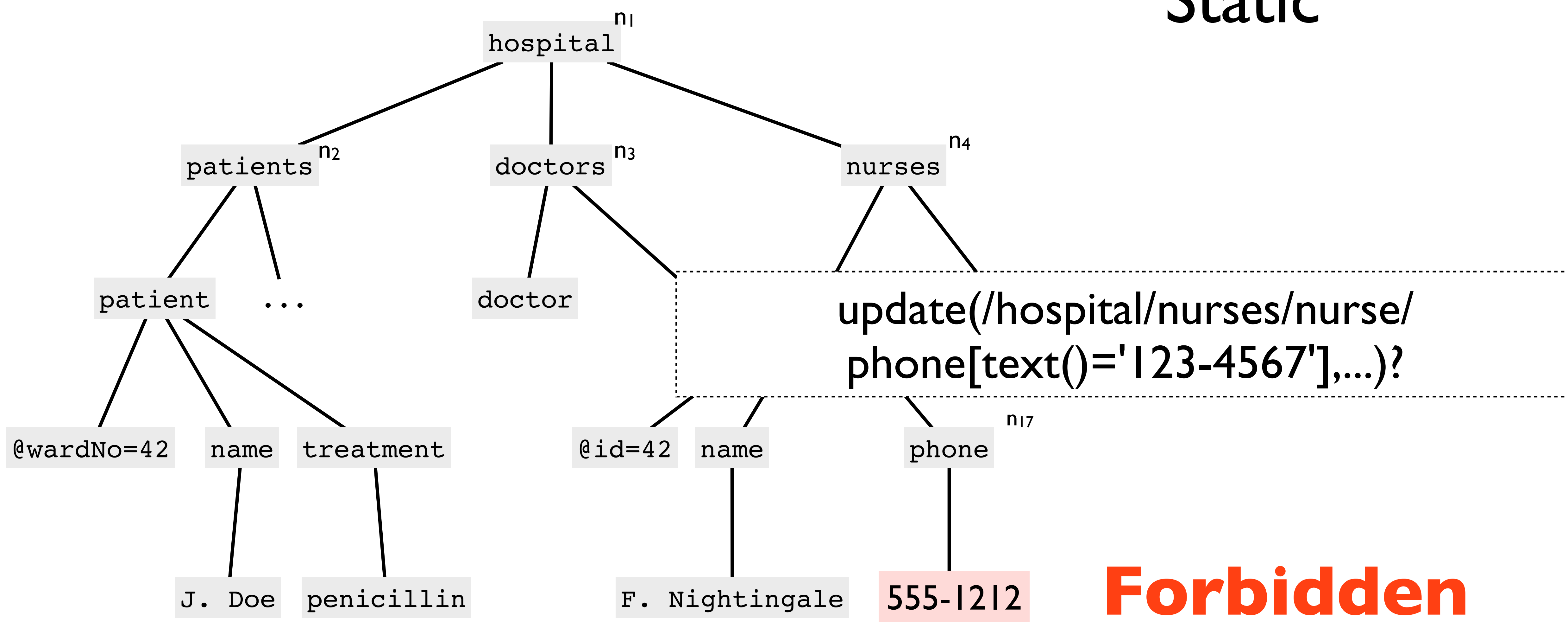
R_4 : $-insert(//*, \texttt{treatment})$

R_5 : $-update(//\texttt{treatment}, *)$

contained in R_1

does not overlap R_4,R_5

Static



Forbidden
(should be allowed!)

$Nurse(\$wn, \$uid)$:

R_1 : $+insert(//\texttt{patient}//*, *)$

R_2 : $+update(//\texttt{patient}[@\texttt{wardNo} = \$wn]/*, *)$

R_3 : $+update(//\texttt{nurse}[@\texttt{id} = \$uid]/\texttt{phone}/*, \texttt{text}())$

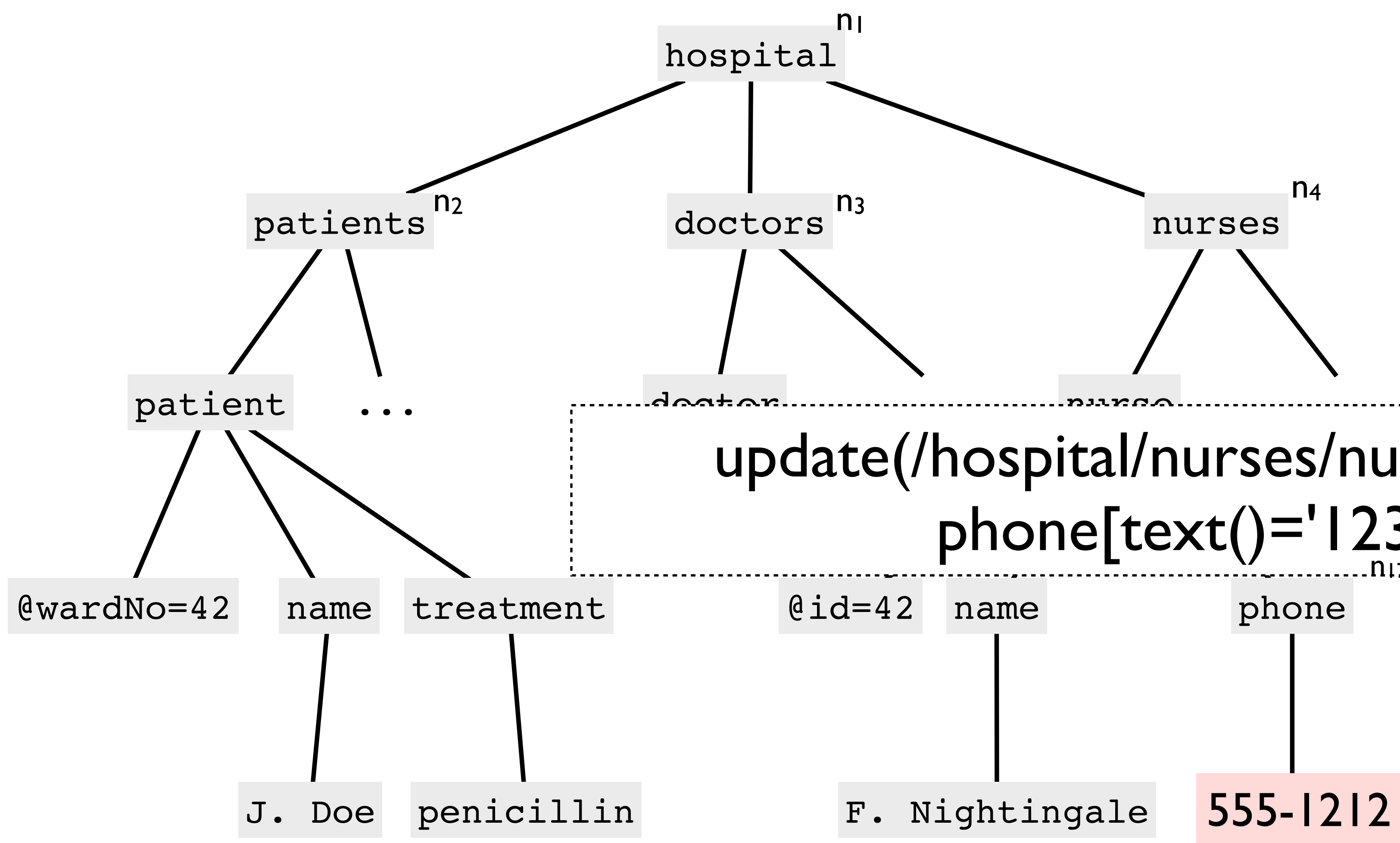
R_4 : $-insert(//*, \texttt{treatment})$

R_5 : $-update(//\texttt{treatment}, *)$

not contained in R_3

does not overlap R_4,R_5

Static



update(/hospital/nurses/nurse/nurse[@id=42]/phone[text()='123-4567'],...)?

Allowed

$Nurse(\$wn, \$uid)$:

$R_1 : \quad +insert(//\texttt{patient}//*, *)$

$R_2 : \quad \cancel{+update(//\texttt{patient}[@\texttt{wardNo} = \$wn]/*, *)}$

$R_3 : \quad +update(//\texttt{nurse}[@\texttt{id} = \$uid]/\texttt{phone}/*, \texttt{text()})$

$R_4 : \quad -insert(//*, \texttt{treatment})$

$R_5 : \quad -update(//\texttt{treatment}, *)$

contained in R3

does not overlap R4,R5

Question

- Static checking is always **sound**
  - all accepted updates are dynamically allowed
- but **incomplete**:
  - but may reject some updates that should be allowed
- **Key problem:** Given a policy language $\mathscr{P}$ and update language $\mathscr{U}$
  - When is static checking for updates from $\mathscr{U}$ against policies from $\mathscr{P}$ **complete** ?
- We call this property **fairness**
  - (to avoid confusion with other notions of completeness)
  - (but possibly introducing confusion with other notions of fairness...)

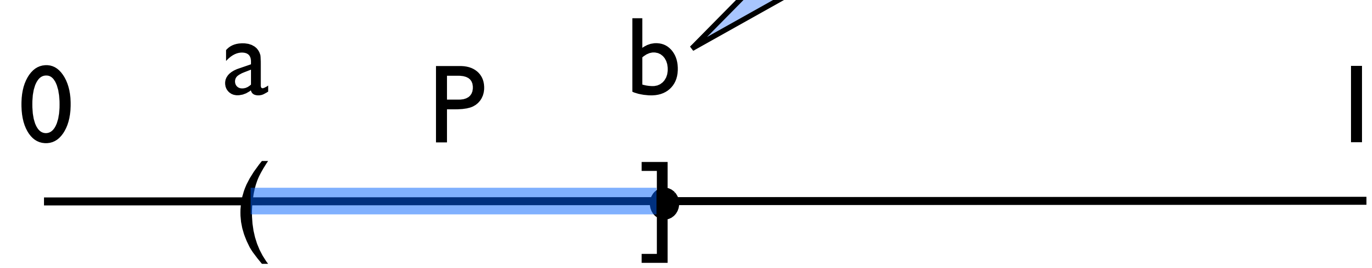
This paper

- XPath-based policies
  - Policies allow "positive" and "negative" rules
  - Simple XACML-style conflict resolution/ default semantics
- Key insight: view update capabilities as forming basis for a **topology**
  - Then policy is fair if it denotes an **open set**

Intuition

- Forget XPath for a minute
  - suppose we wan
  - Requests specifi
  - Interval allowed

Fairness fails if there is a point s.t. every covering update request also goes outside P



Background

- XPath expressions

Paths	p	$::=$	$\alpha :: \phi \mid p/p' \mid p[q]$
Filters	q	$::=$	$p \mid q \text{ and } q \mid @f = d \mid \text{true}$
Axes	α	$::=$	$\text{self} \mid \text{child} \mid \text{descendant} \mid \text{attribute}$
Node tests	ϕ	$::=$	$l \mid * \mid f \mid \text{text}()$

- Atomic updates

$$u ::= \mathit{insert}(n, T') \mid \mathit{update}(n, T') \mid \mathit{delete}(n)$$

- Update capabilities

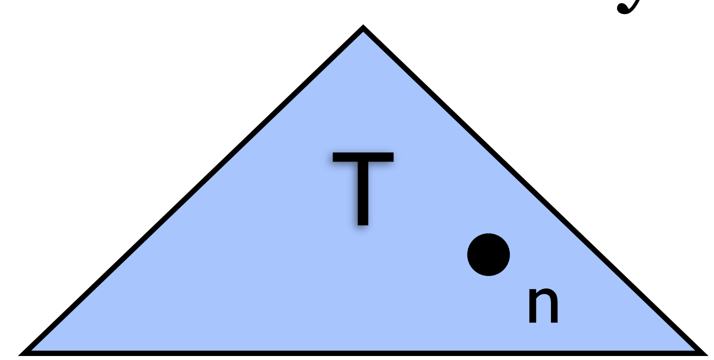
$$U ::= \mathit{insert}(p, \phi) \mid \mathit{update}(p, \phi) \mid \mathit{delete}(p)$$

Policies

- $P = (ds, cr, A, D)$
  - A = allowed capabilities
  - D = denied capabilities
  - ds = default semantics (+ or -)
    - what to do if no rule applies
  - cr = conflict resolution policy (+ or -)
    - what to do if both A and D rule applies

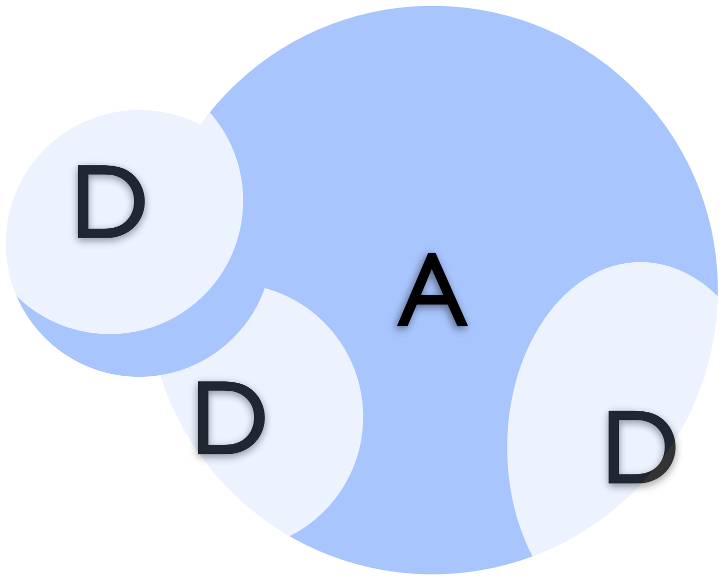
Semantics

- Conventional semantics $[\![p]\!](T) = \{n_1,\ldots,n_k\}$
- Instead, take $\langle\!\langle p \rangle\!\rangle = \{(T,n) \mid n \in [\![p]\!](T)\}$
  - a "point" (T,n) is a tree T with a designated node n
  - essentially a "tree pattern" with only child edges



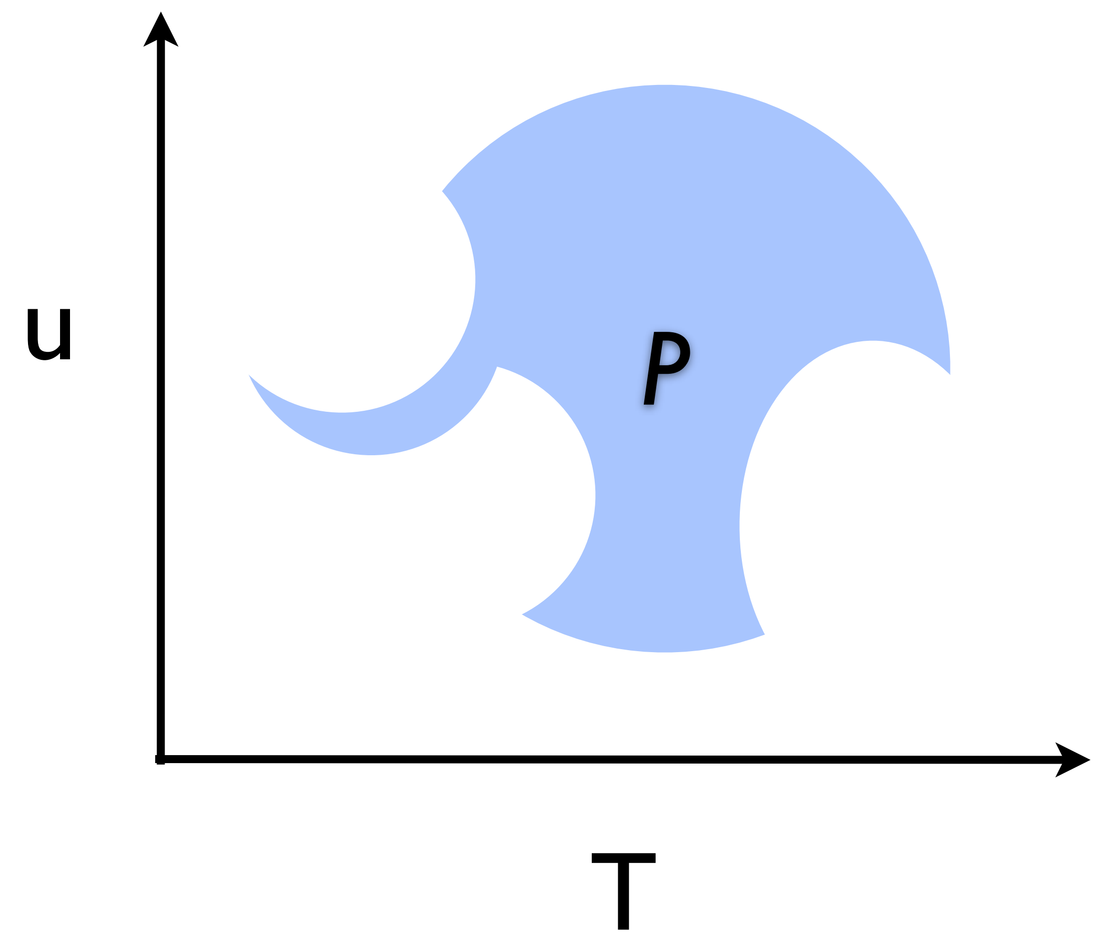
Intuition

- Simple case (-,-, A,D) - only "delete(p)"
  - policy = positive rules - negative rules.



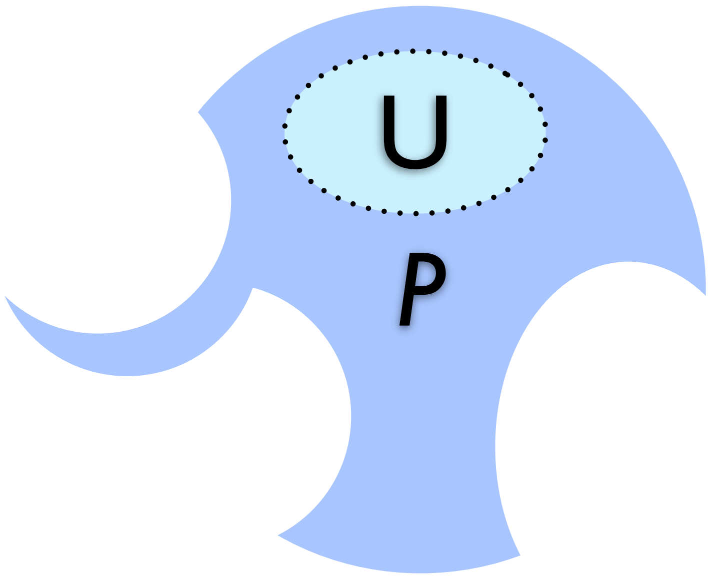
Intuition

- Think of $\langle\!\langle P \rangle\!\rangle$ as 2-dimensional region
  - x-axis: trees, y-axis: atomic updates



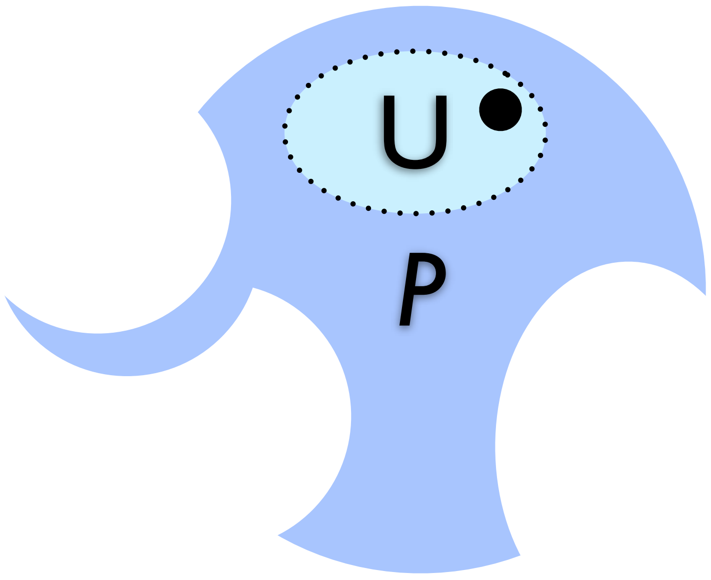
Intuition

- Basic open sets = sets definable by update capabilities



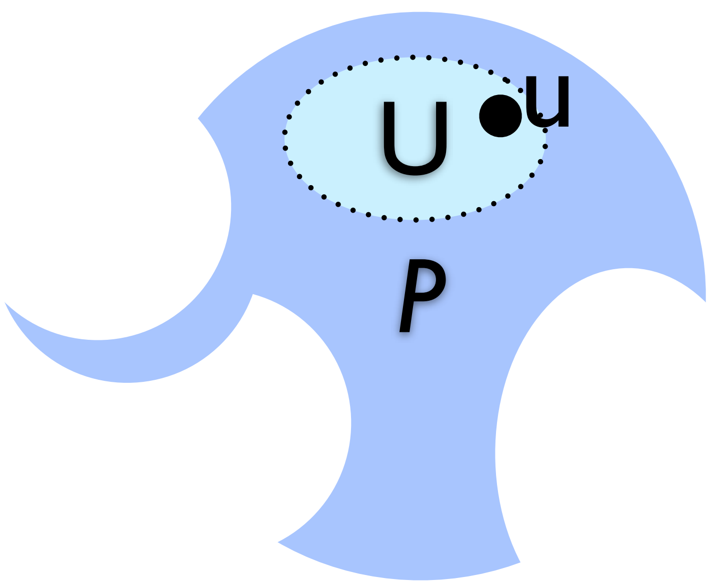
Intuition

- Openness means each point is in an open neighborhood contained in *P*



Intuition

- Fairness means each atomic update is contained in an update capability *U* contained in *P*



- (*U* contained in *P* == statically allowed)

Definition

- P is *fair* (with respect to updates in $\mathscr{U}$) if and only if
    - for every (T,u) in $\langle\!\langle P \rangle\!\rangle$, there exists $U \in \mathscr{U}$ such that (T,u) in $\langle\!\langle U \rangle\!\rangle \subseteq \langle\!\langle P \rangle\!\rangle$
- equivalently:
    - P is open in the topology generated by the sets $\langle\!\langle U \rangle\!\rangle$
    - (note: need to show these sets form a basis, which they do for all examples we care about)

Results

- We consider two scenarios:
- $\mathscr{P} = \mathrm{XP}^{(/,//,*)}$ ($\mathscr{U} = \mathrm{XP}^{(/)}$ or larger)
  - All policies are open / fair
  - (the basic open sets form a partition)
- $\mathscr{P} = \mathrm{XP}^{(/,//,*,[])}$ ($\mathscr{U} = \mathrm{XP}^{(/,[])}$ or larger)
  - All policies with **only positive filters** are open/fair
  - Checking fairness in general (for $\mathscr{U} = \mathrm{XP}^{(/,[])}$) is coNP-complete

$\mathrm{XP}^{(/,//,*)}$

- Key idea: show each path is = to union of linear path sets (basic open sets)

$$\begin{aligned}
\mathsf{LP}(\mathsf{self} :: \phi) &= \{\mathsf{self} :: l \mid l \in [\![\phi]\!]\} \\
\mathsf{LP}(\mathsf{child} :: \phi) &= \{\mathsf{child} :: l \mid l \in [\![\phi]\!]\} \\
\mathsf{LP}(\mathsf{descendant} :: \phi) &= \mathsf{LP}(\mathsf{child} :: *)^* \cdot \mathsf{LP}(\mathsf{child} :: \phi) \\
\mathsf{LP}(p/p') &= \mathsf{LP}(p) \cdot \mathsf{LP}(p')
\end{aligned}$$

- The basic open sets partition the space of (T,n)'s, hence all open sets are also closed
  - hence finite boolean combinations are always open

$\mathrm{XP}^{(/,//,*,[])}$

- Linear path sets no longer suffice
  - /a[b] not open w.r.t. linear path basis
- Instead, consider filter path sets

$$
\begin{array}{rcl}
\mathsf{FP}(ax :: \phi) & = & \mathsf{LP}(ax :: \phi) \\
\mathsf{FP}(p/p') & = & \mathsf{FP}(p) \cdot \mathsf{FP}(p') \\
\mathsf{FP}(p[q]) & = & \{p'[q'] \mid p' \in \mathsf{FP}(p), q' \in \mathsf{FP}^{\mathsf{Q}}(q)\} \\
\mathsf{FP}^{\mathsf{Q}}(p) & = & \mathsf{FP}(p) \\
\mathsf{FP}^{\mathsf{Q}}(q_1 \text{ and } q_2) & = & \{q'_1 \text{ and } q'_2 \mid q'_1 \in \mathsf{FP}^{\mathsf{Q}}(q), q'_2 \in \mathsf{FP}^{\mathsf{Q}}(q')\} \\
\mathsf{FP}^{\mathsf{Q}}(\mathsf{true}) & = & \{\mathsf{true}\}
\end{array}
$$

$XP^{(/,//,*,[])}$

- Again, all paths denote open sets (taking filter path sets to be open)
- But complements not necessarily open
  - /a[b] open, but not /a - /a[b]
  - "can witness presence of b but not absence"
- Proof: filter path sets are closed under homomorphisms, and /a - /a[b] is not
- *(NB. Adding negation /a[not(b)] would help but make containment much harder.)*

Complexity of fairness

- Question: Given policy P over $XP^{(/,//,*,[])}$, is it fair (w.r.t $\mathscr{U} = XP^{(/,[])}$)?
- Hardness:
  - Reduce from coNP-hardness of Path containment (Miklau & Suciu 2004)
  - $p \sqsubseteq p' \Leftrightarrow$ /*[p] - /*[p'] open (in fact empty)
  - $\Leftrightarrow$ (-,-,{/*[p]},{/*[p']}) fair

Complexity of fairness

- Upper bound: need to show that unfairness has a small (polynomial size) counterexample
- Basic idea: similar to coNP argument for XPath containment [Miklau & Suciu 2004]
  - assume a witness is given (consisting of T, T' and homomorphism)
  - shrink to polynomial-size while preserving witness property

Complexity of enforcement

- In general, enforcing policy statically requires solving
  - overlap: PTIME for $XP^{(/,//,*,[])}$
  - containment: coNP-complete for $XP^{(/,//,*,[])}$
- However, $p \sqsubseteq p'$ can be solved in PTIME if p has a bounded number of // steps
  - i.e. if we restrict updates to have small number of // steps (which is reasonable).
  - again, drawing on Miklau & Suciu's results

Extensions

- Attributes
  - seem straightforward but need to take uniqueness into account
  - negative attribute filters may be OK & would be useful
- Schemas
  - complicates containment, overlap tests
- Richer classes of XPath-based capabilities & policies
  - increasing expressiveness typically makes fairness easier (cf. negation) but increases complexity of static analysis

Conclusions

- Fine-grained XML access control can be expensive to enforce dynamically
- In general, static enforcement is incomplete
- Fortunately, it is complete in common cases
  - polynomial time static enforcement also possible
  - checking fairness can be expensive in general
- Analysis of policy fairness problem reveals an interesting connection between topology
  - should be applicable to other settings also