University of Oxford
Department of Computer Science

ShareAlike Your Data

Self-Referential Usage Policies for the Semantic Web

Markus Krötzsch
University of Oxford

Sebastian Speiser
Karlsruhe Institute of Technology

October 25, 2011

Motivation

"This is the reference implementation of the self-referential joke."

– Randall Munroe

Motivation

"This is the reference implementation of the self-referential joke."

– Randall Munroe

- Sentence taken from xkcd.com
- Published under a policy (license):

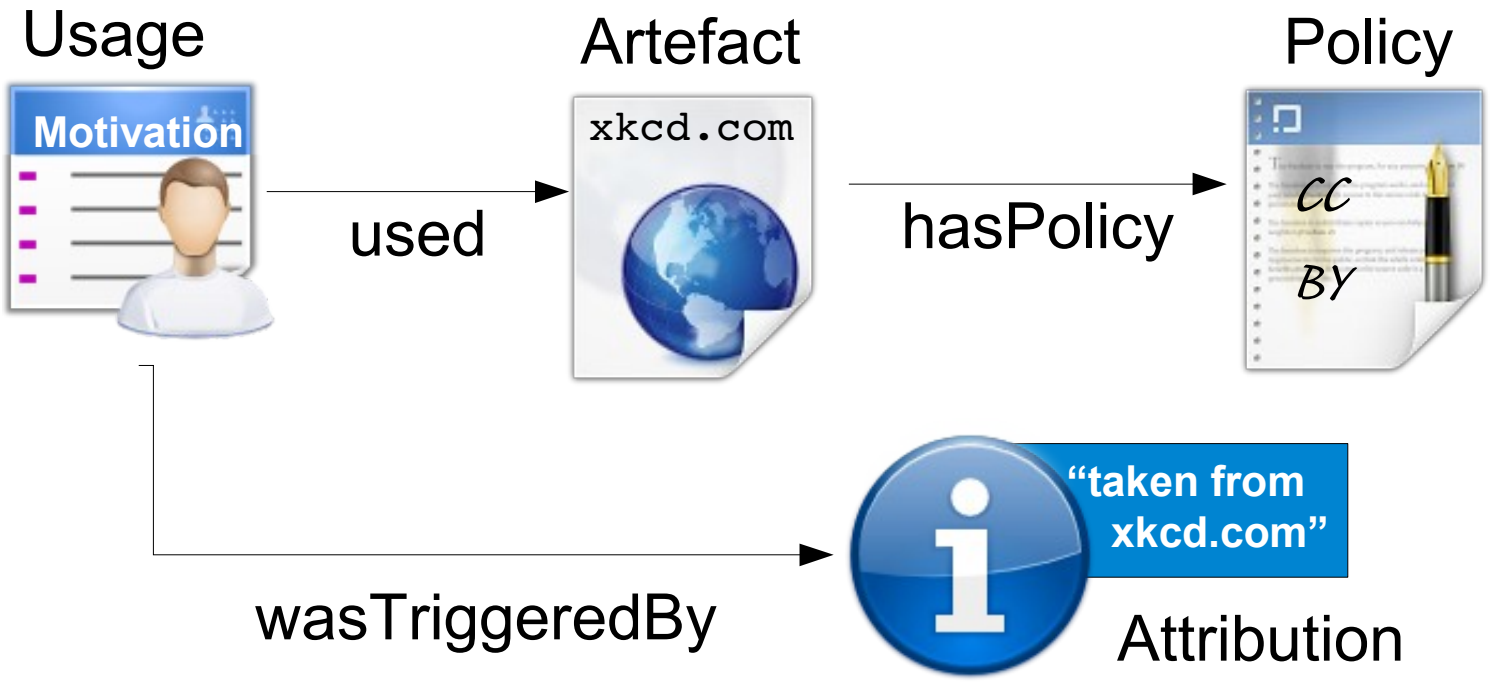
  Creative Commons Attribution Non-Commercial (CC BY-NC)

→ Usage in this talk is permitted

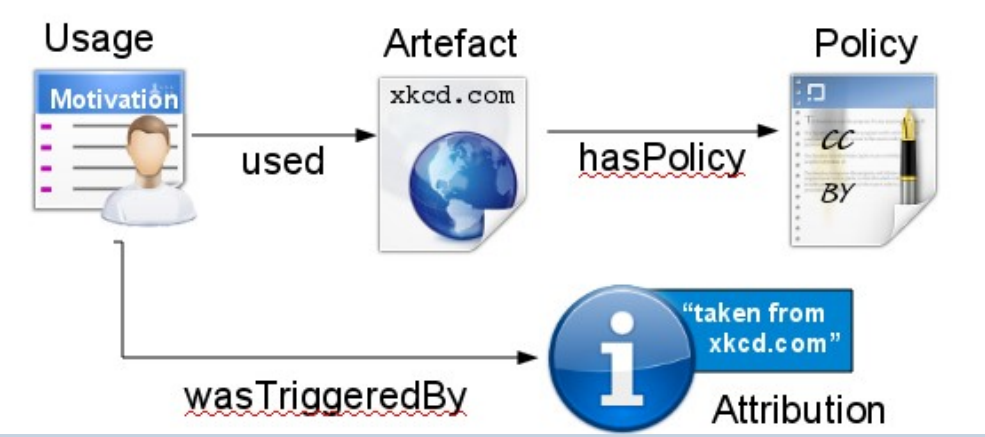
Can We Formalise Policy Conformance?

Can We Formalise Policy Conformance?

- A simple provenance model:



Can We Formalise Policy Conformance?



- Policy conformance can be captured in logic:

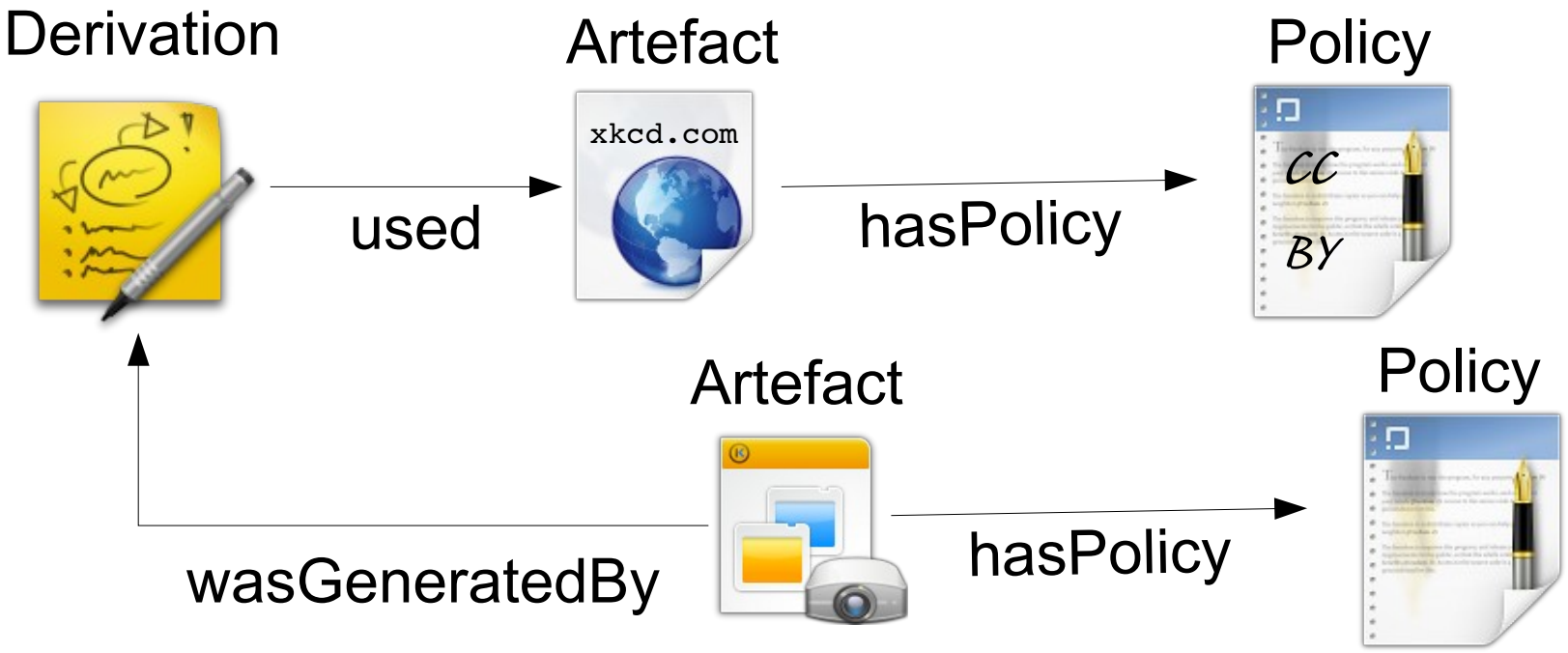
conformsTo(x,CCBy) ↔
Usage(x), wasTriggeredBy(x,y), Attribution(y)

→ Content-based Policies [Speiser & Studer, ISWC '10]

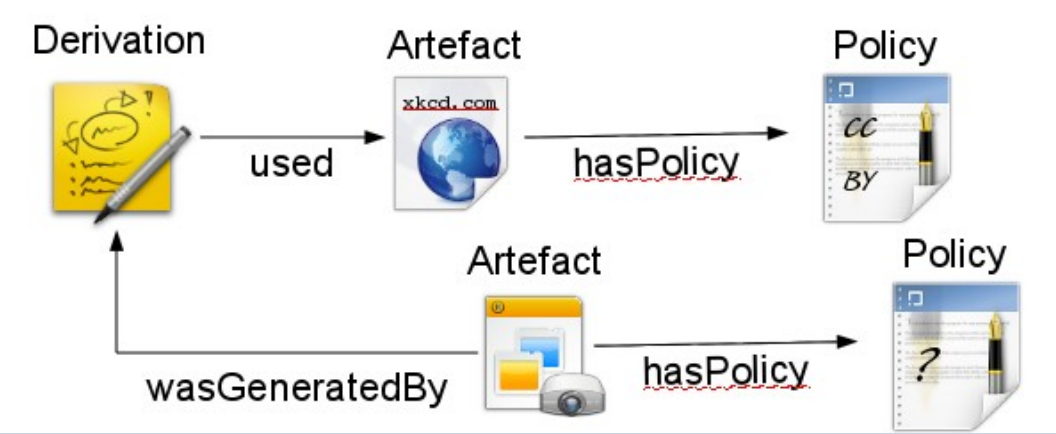
Example: Derivations under Copyleft Licenses

Example: Derivations under Copyleft Licenses

- Modelling the derivation of an artefact:



Example: Derivations under Copyleft Licenses



- Policy description:

conformsTo(x,CCBy) ↔
Derivation(x), wasGeneratedBy(z,x), Artefact(z),
hasPolicy(z,v), *AllowedCCByDerivationPolicy*(v)

Which Policies are Allowed for Derivations?

- Existing licenses are **name-based.**
- **GPL:**
  - Derivation must use GPL
  - Desired viral effect
- **Creative Commons:**
  - Derivation must use a compatible CC license
  - *Undesired* restriction to CC

Content-Based Policies with Self-Reference

Content-Based Policies with Self-Reference

- Describe **relevant features** instead of listing permitted license names:

conformsTo(x,CCBy) ↔
Derivation(x), wasGeneratedBy(z,x), Artefact(z),
hasPolicy(z,v), containedIn(v,CCBy)

where containedIn(y,z) means:

containedIn(v,w) ↔ ∀x.conformsTo(x,v) → conformsTo(x,w)

Content-Based Policies with Self-Reference

- **Policy Modelling in General:**
  Policies are described by formula of one free variable

  $\text{conformsTo}(x,p) \leftrightarrow F_p[x]$

  …

  (for all policies p, q, …)

  where $F_p[x]$ may use containedIn and we have:

$$\text{containedIn}(v,w) \leftrightarrow \forall x.\text{conformsTo}(x,v) \rightarrow \text{conformsTo}(x,w)$$

Why First-order Logic is not Enough

- Formalising two similar licences:

$$\text{conformsTo}(x,\text{CCByNC}) \leftrightarrow \text{NonCommercial}(x), \text{Derivation}(x), \text{wasGeneratedBy}(z,x), \text{Artefact}(z), \text{hasPolicy}(z,v), \text{containedIn}(v,\text{CCByNC})$$

$$\text{conformsTo}(x,\text{CCByNC}') \leftrightarrow \text{NonCommercial}(x), \text{Derivation}(x), \text{wasGeneratedBy}(z,x), \text{Artefact}(z), \text{hasPolicy}(z,v), \text{containedIn}(v,\text{CCByNC}')$$

$$\text{containedIn}(v,w) \leftrightarrow \forall x.\text{conformsTo}(x,v) \rightarrow \text{conformsTo}(x,w)$$

Why First-order Logic is not Enough

- Formalising two similar licences:

conformsTo(x,CCByNC) ↔
NonCommercial(x), Derivation(x), wasGeneratedBy(z,x),
Artefact(z), hasPolicy(z,v), containedIn(v,CCByNC)

conformsTo(x,CCByNC') ↔
NonCommercial(x), Derivation(x), wasGeneratedBy(z,x),
Artefact(z), hasPolicy(z,v), containedIn(v,CCByNC')

containedIn(v,w) ↔ ∀x.conformsTo(x,v) → conformsTo(x,w)

does **not** imply

containedIn(CCByNC,CCByNC')

A Greatest Fixed Point Semantics

- Idea: Assume as many containedIn relations as possible; then eliminate unsustainable containments

$$P(CI) = \{\ containedIn(p,q) \mid CI \models \forall x.F_p[x] \rightarrow F_q[x]\ \}$$

A Greatest Fixed Point Semantics

- Idea: Assume as many containedIn relations as possible; then eliminate unsustainable containments

$$P(CI) = \{\ containedIn(p,q) \mid T_{ci} \cup CI \models \forall x.F_p[x] \rightarrow F_q[x]\ \}$$

where

$$T_{ci} = \{\ \forall x,y,z.containedIn(x,y) \wedge containedIn(y,z) \rightarrow containedIn(x, z),\ \forall x.containedIn(x,p\text{-}top),\ \forall x.containedIn(p\text{-}bot,x)\ \}$$

A Greatest Fixed Point Semantics

- Idea: Assume as many containedIn relations as possible; then eliminate unsustainable containments

$$P(CI) = \{\ containedIn(p,q) \mid T_{ci} \cup CI \models \forall x.F_p[x] \rightarrow F_q[x]\ \}$$

- Computation: iterate P from top until fixpoint gfp(P)
- Semantics: Assume gfp(P) to hold

- Note: easy to compute, based on standard first-order logic

Examples and Problems

- CCByNC' example:

  we obtain containedIn(CCByNC,CCByNC')

- Another example:

$F_p[x] : A(x) \wedge \text{containedIn}(p, q)$
$F_q[x] : B(x)$

Examples and Problems

- CCByNC' example:

  we obtain containedIn(CCByNC,CCByNC')

- Another example:

  $F_p[x] : A(x) \wedge \text{containedIn}(p, q)$
  $F_q[x] : B(x)$

  gfp(P) does not contain containedIn(p,q)
  but first-order theory implies containedIn(p,q)

Connecting Policy Descriptions

- Which policy descriptions lead to meaningful results?
- **Connected Fragment:** require guard atoms that "connect" formula variable structure

$\exists x.\mathrm{Usage}(x) \wedge \exists y.\mathrm{wasTriggeredBy}(x,y) \wedge \mathrm{Attribution}(y)$

$\forall x.\mathrm{Usage}(x) \vee \forall x.\mathrm{Derivation}(x)$

$\exists x.A(x) \wedge \mathrm{containedIn}(p,q)$

Connecting Policy Descriptions

- Which policy descriptions lead to meaningful results?
- **Connected Fragment:** require guard atoms that "connect" formula variable structure
- **Theorem:**
  If policy descriptions are connected, then
  the containedIn facts in gfp(P) are exactly those that follow when assuming gfp(P) in first-order logic.

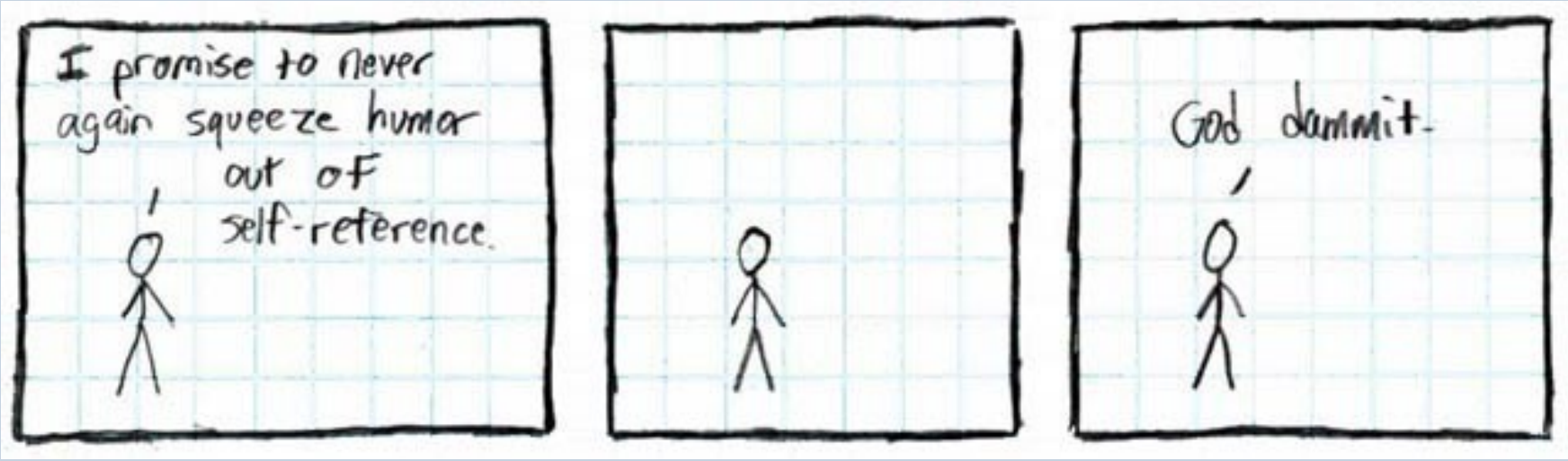
Self-Referential Policies in Practice

- Approach works in all first-order modelling languages
  → OWL DL, Datalog rules, …
- Compatible with background knowledge (ontologies)
- Can use standard tools (OWL reasoners, …)
- Syntactic restrictions often very mild/easy to check

Summary

Self-reference in content-based policies is possible

- under a **meaningful semantics**
- that is **easy to compute** and
- **compatible** with existing technologies (OWL, Datalog).



http://xkcd.com/33/ CC BY-NC