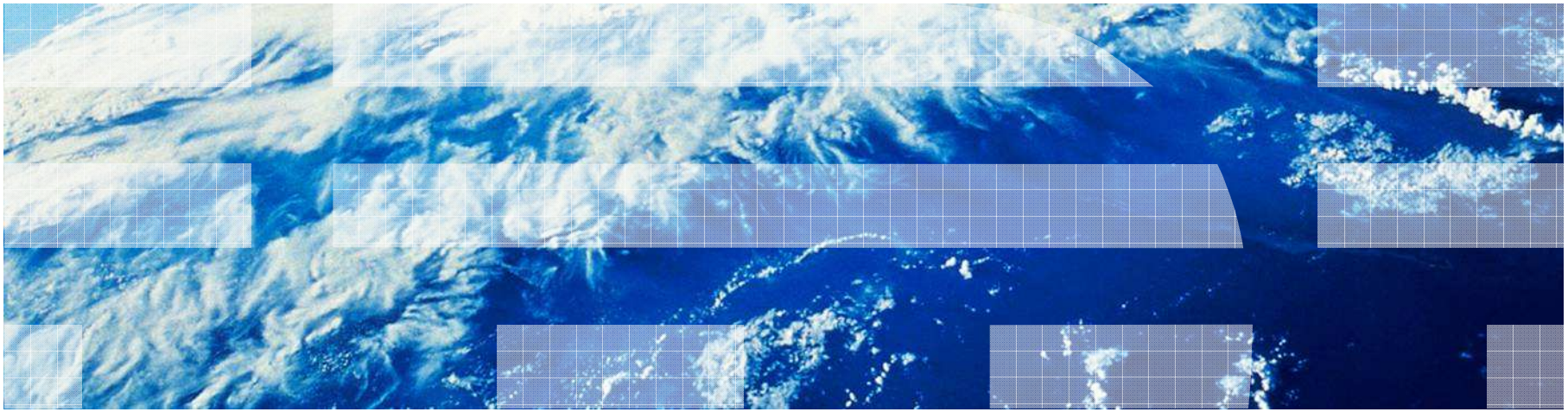
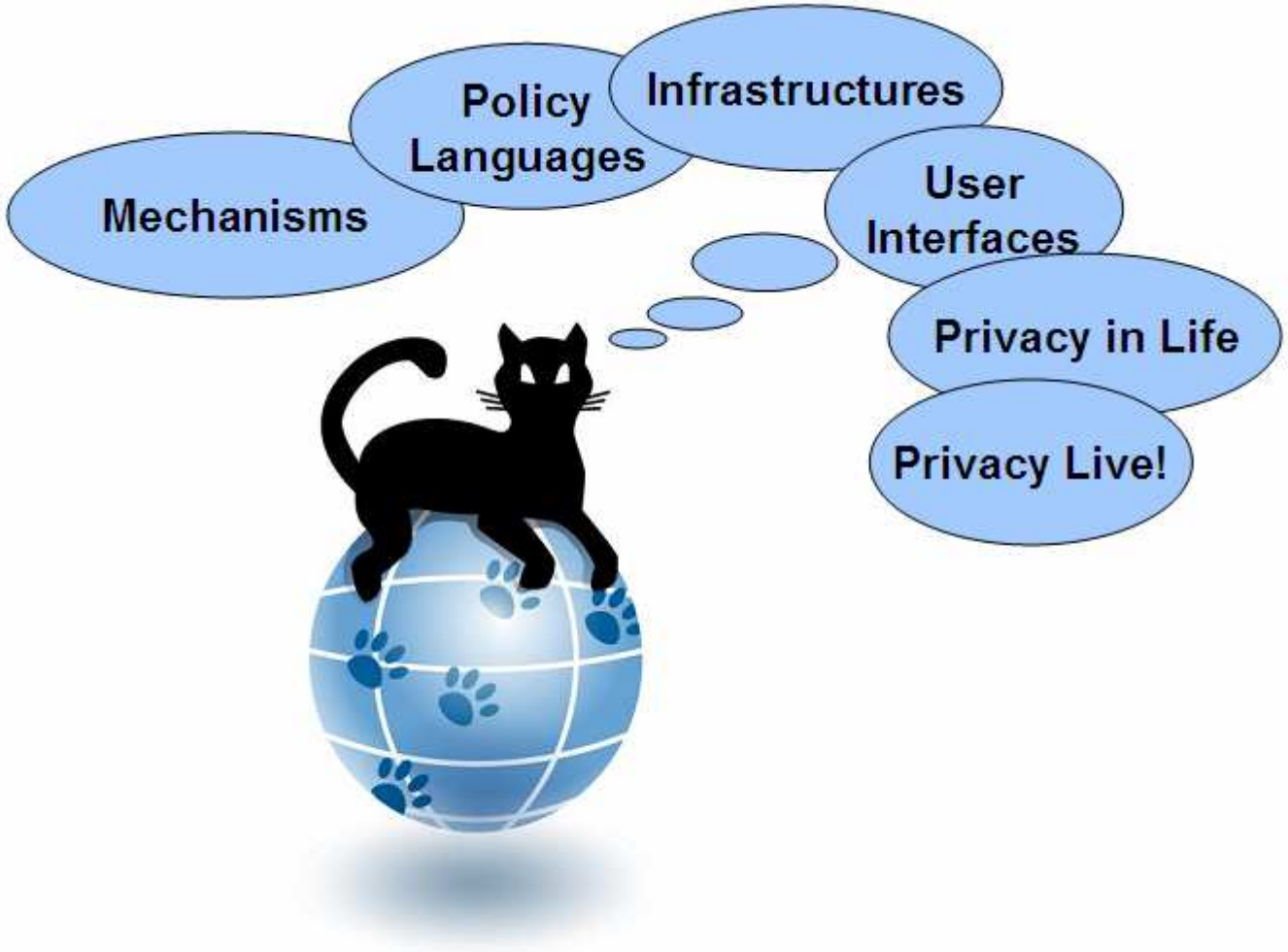
Gregory Neven, IBM Research – Zurich
SACML TC Confcall, November 16, 2010

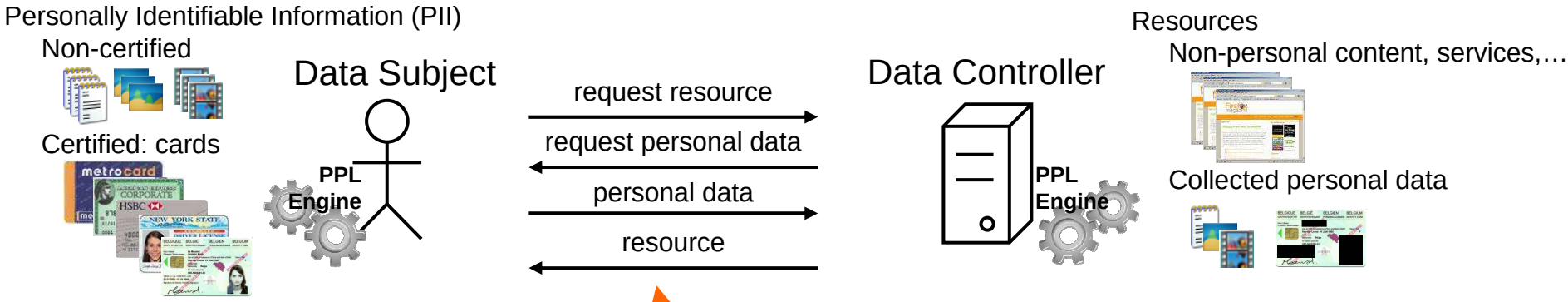
Asserting attribute predicates in SAML and XACML



Privacy and Identity Management for Life



The PrimeLife Policy Language



Specific Policy:
over specific personal data (e.g. birth date)
- **Access control policy (ACP):**
  who can access (e.g. PrivacySeal silver)
- **Data handling preferences (DHPrefs):**
  how is to be treated when revealed
  - **Authorizations** (e.g. marketing purposes, forwarded to PrivacySeal gold)
  - **Obligations** (e.g. delete after ≤2y)

Generic Preferences:
DHPrefs over implicitly revealed personal data (e.g. IP address, cookies,…)
- **Authorizations** (e.g. admin purposes)
- **Obligations** (e.g. delete after ≤2y)

SAML

XACML

Specific Policy:
over specific resource (e.g. BuyService)
- **Access control policy (ACP):**
  who can access
  - cards to possess (e.g. ID card)
  - personal data to reveal (e.g. nationality)
  - conditions to satisfy (e.g. age>18)
- **Data handling policy (DHP):**
  how revealed personal data will be treated
  - **Authorizations** (e.g. marketing purposes)
  - **Obligations** (e.g. delete after 1y)

Generic Policy:
DHP over implicitly revealed personal data (e.g. IP address, cookies,…)
- **Authorizations** (e.g. admin purposes)
- **Obligations** (e.g. delete after 1y)

Main features of PPL

- Privacy-friendly card-based access control
  - attributes bundled in cards
  - technology independence
  - multi-card claims
  - **support anonymous credentials (Identity Mixer, U-Prove)**
  - **reveal attributes vs. prove predicates over attributes**
- Policy sanitization
- Integrated data handling
  - two-sided detailed data handling preferences/policies
  - automated matching procedure
  - extensible vocabularies
  - downstream usage

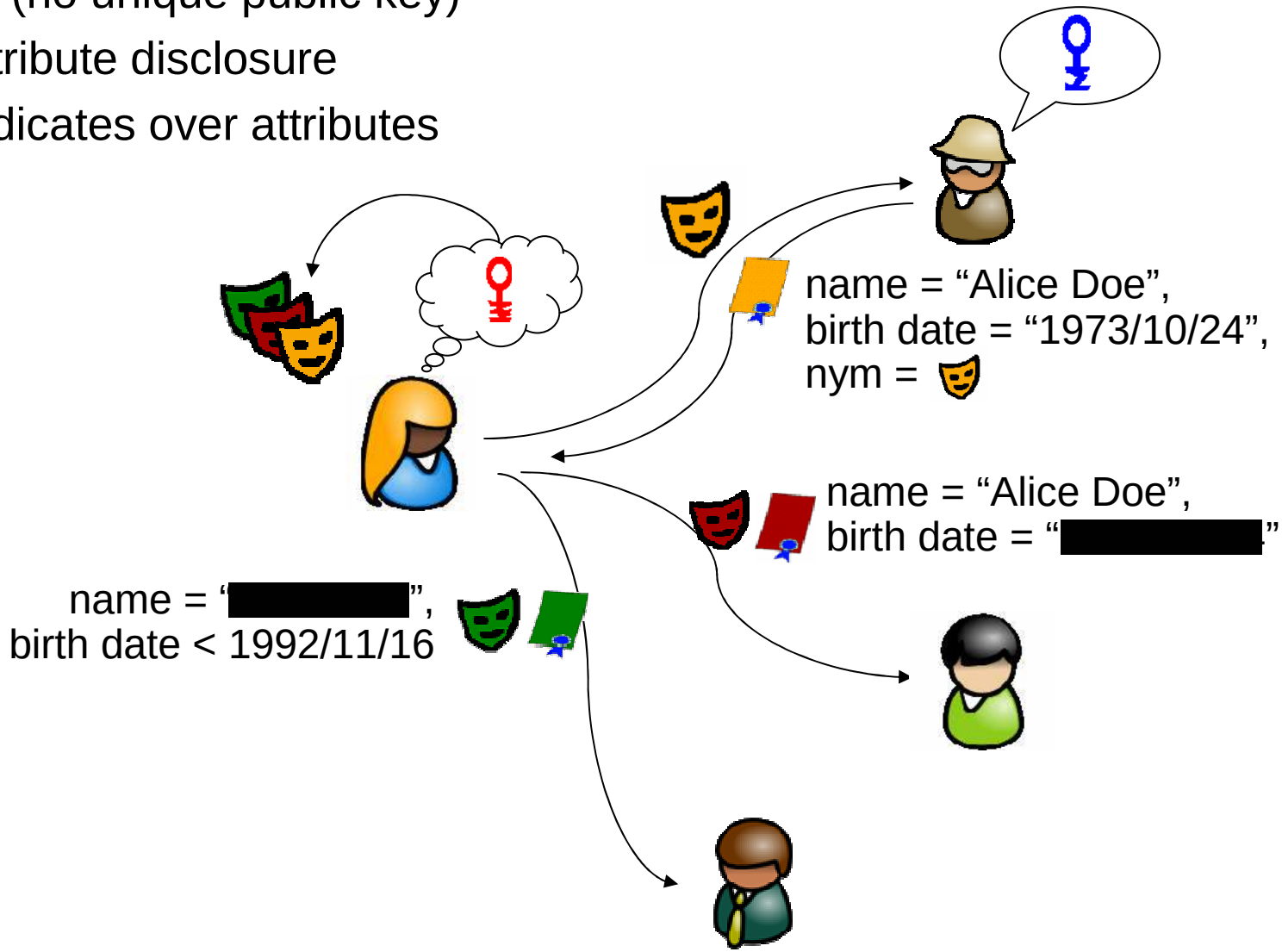
What to standardize

- Card-based access control
  - Advanced concepts
  - Market demand for multi-card claims?
  - **Breaks open XACML schema & data flow**
- Integrated data handling policies/preferences
  - **Breaks open XACML schema & data flow**
  - Quite orthogonal, could be embedded in any language
  - See W3C Boston workshop
- Suggestion: conditions over attributes in SAML + profile for XACML
  - allow IDPs to assert predicates over attributes rather than full values (standard signatures if online IDP, anonymous creds if offline)
  - allow certified predicates to be fed into XACML evaluation process challenge: without breaking XACML schema/architecture

Anonymous credentials

e.g., Identity Mixer, U-Prove

- unlinkability (no unique public key)
- selective attribute disclosure
- proving predicates over attributes



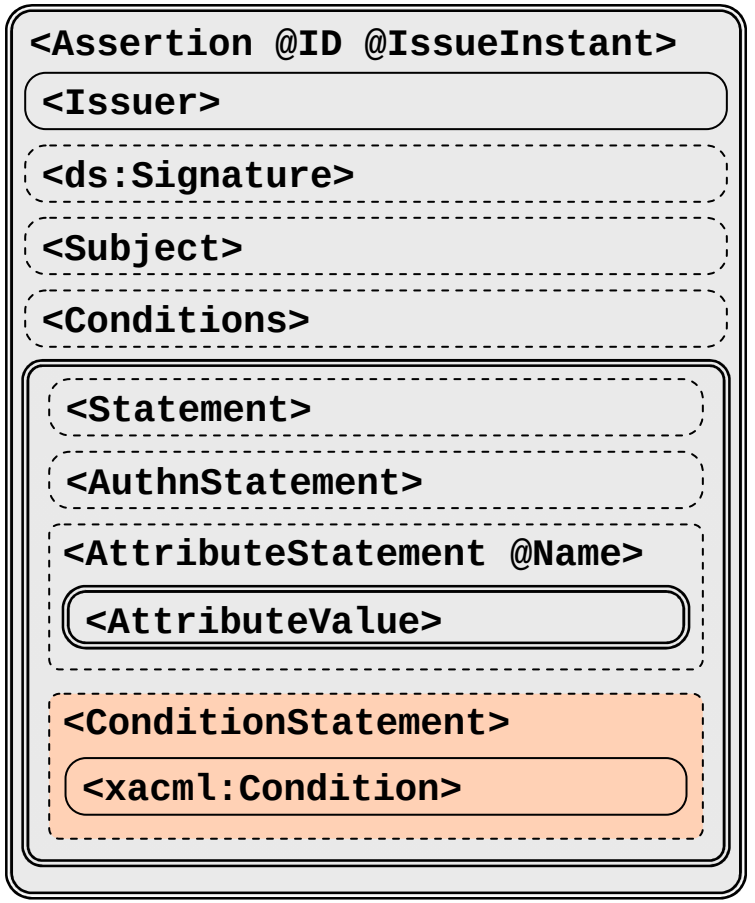
Example attribute predicates

- birthdate < 2010/11/16
- frequent flyer status > gold
- phone number starts with +4144 (i.e., Zurich landline)
- 92100 < zip code < 92200 (i.e., address in San Diego)
- domain of email address is ibm.com
- issuedate > birthdate + 18Y (i.e., issued when holder was over 18)
- …

PPL claims format

- saml:Statement is abstract
- Profiles can define new statement types

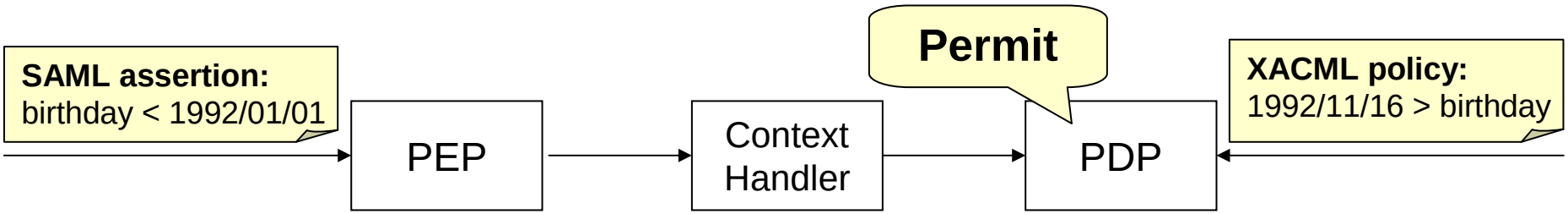
  e.g., ppl:ConditionStatement
- Borrow schema and functions ontology from xacml:Condition
- Already in PPL, fairly straightforward to write up proposal



Attribute predicates in XACML

How to feed predicates over attributes into XACML?

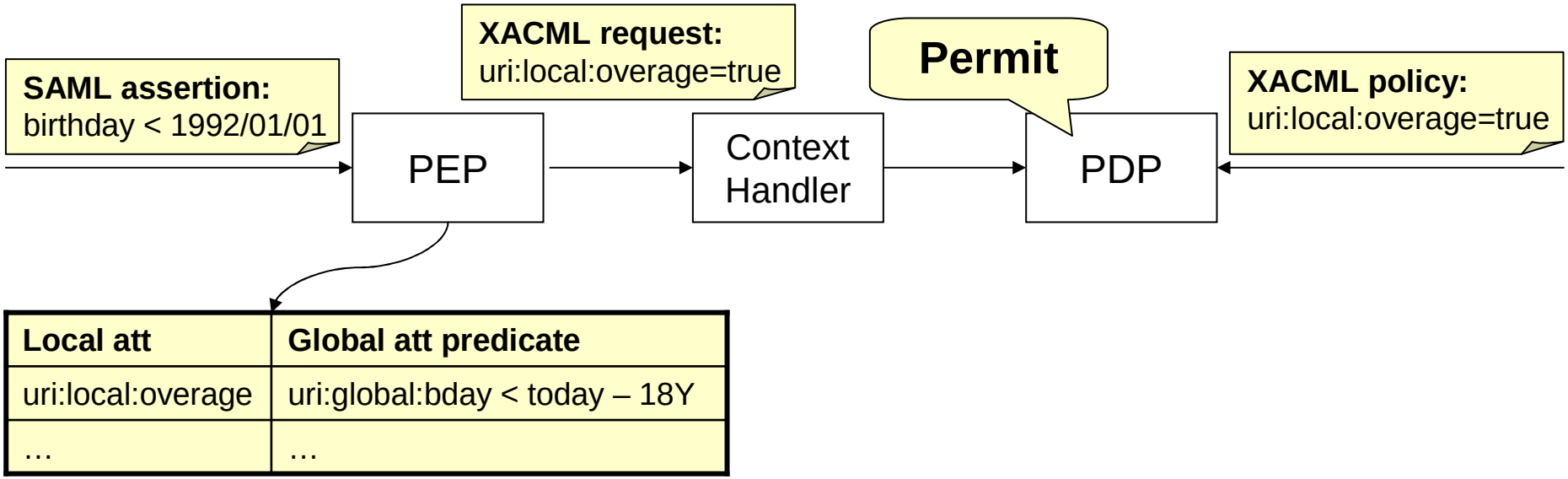
cfr. SAML profile of XACML



A simple solution

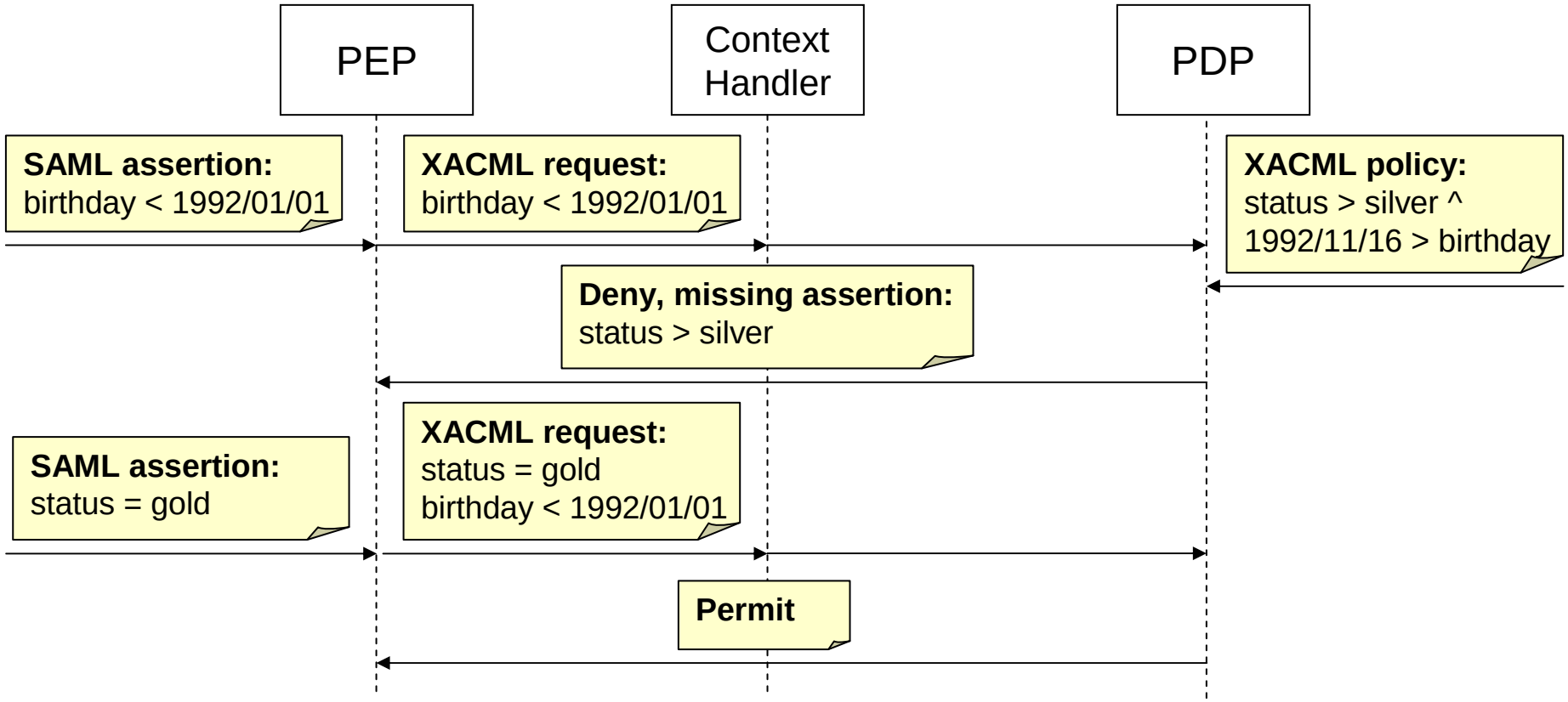
Policy defined in terms of boolean, locally defined attributes

PEP knows mapping to predicates over globally meaningful attributes



Local att	Global att predicate
uri:local:overage	uri:global:bday < today – 18Y
…	…

A more challenging solution



Issues:

1. How to communicate asserted predicates to PDP?
2. How to determine “missing predicates”?
3. How to evaluate policy, given set of asserted predicates?

Communicating conditions to PDP

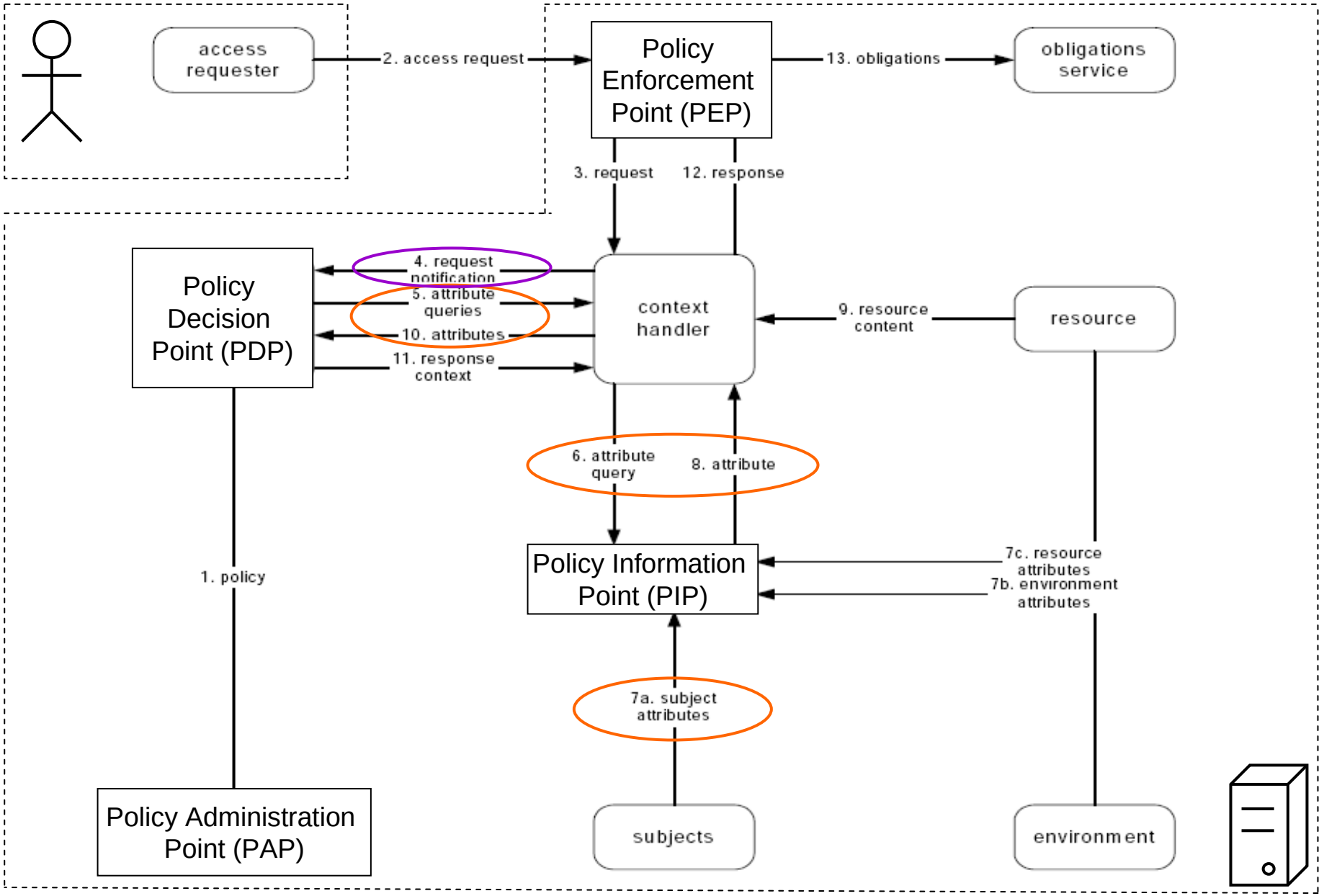
1. How to communicate asserted predicates to PDP?
   - Insert into request context

     → break open `xacml:Request` schema

     XACML 3.0: *"However a conforming PDP is not required to actually instantiate the context in the form of an XML document."*
   - Insert into attribute queries/responses – schema?
     - SAML?
     - Indeterminate response with missing attributes in status detail?
     - no schema at all?

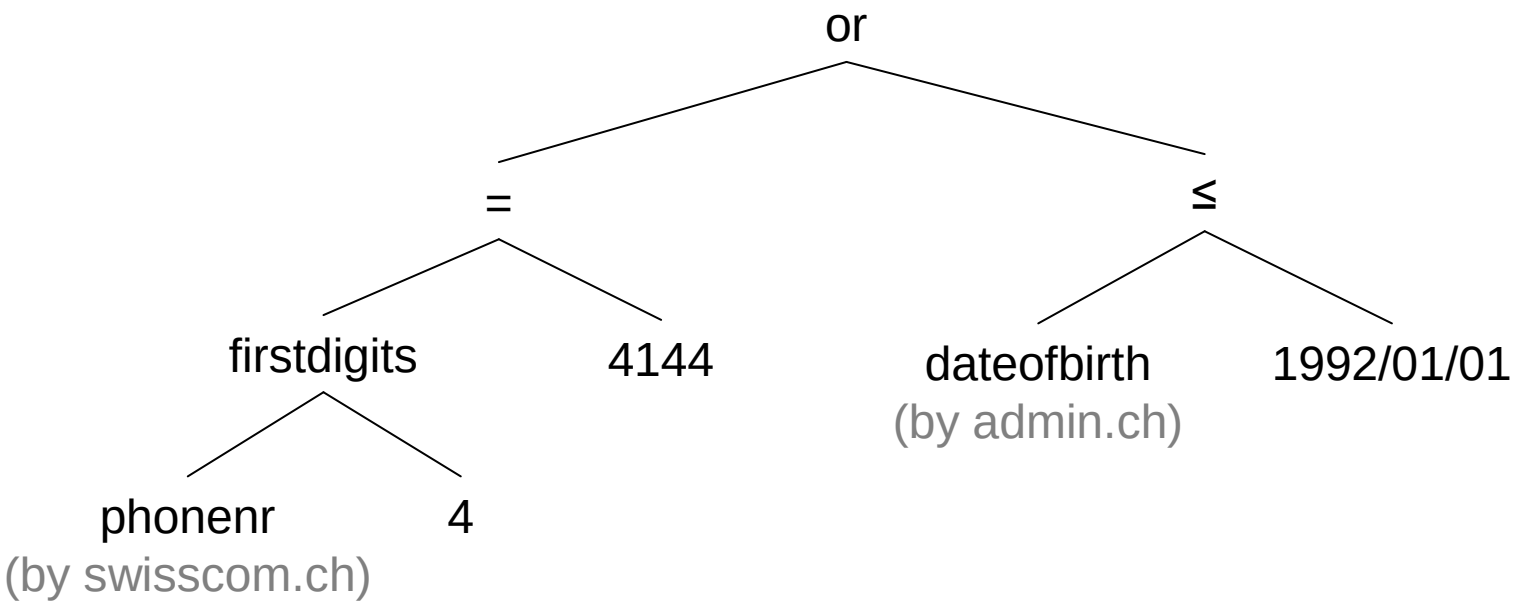
     can introduce our own without breaking schema?

XACML data flow



Missing conditions

2. How to determine “missing predicates”?
   - Lowest expressions with boolean result
   - Highest expressions with attributes by same issuer
   - Entire condition from rule



Evaluating policies

3. How to evaluate policy wrt given set of asserted predicates?
   - String equality
   - XML tree equivalence
   - Reasoning engine to test implication

     e.g., (dateofbirth ≤ 1992/11/16) ⇒ (1992/01/01 ≥ dateofbirth) ?

   How new evaluation mechanism triggered?

Candidate approaches

Approach 1: PPL

- Asserted predicates embedded in request context
- Request full condition in rule
- Evaluation by string/XML equality + value substitution
- Triggered by modied PDP code

Very invasive in schema/architecture

Approach 2: dedicated attributes

personal favorite on short term

- Policy in terms of dedicated, locally defined, boolean attributes
- PIP or PEP knows mapping to predicates over globally defined attributes
  e.g., urn:mypolicy:underage → (urn:unitednations:birthdate ≤ 1992/11/16)
- Values of local attributes passed in request context
- Missing local attribute → request corresponding predicate over global atts

Minimal impact on XACML schema/architecture

Burden on policy author of determining recurring predicates

Candidate approaches (2)

Approach 3a: dedicated function per condition

- Insert predicates into request context
- Function implementation knows mapping to predicate over global atts
- Fetches directly if missing, returns TRUE iff satisfied

Policy author needs to program Java/… for each relevant predicate

Need to somehow initialize function with asserted predicate

Approach 3b: generic boolean function

- Predicate to be asserted encoded as function argument (string)
- Function implementation requests specified predicate if missing, returns TRUE iff satisfied

No programming required

Predicate looks ugly (&nbgt;)

Need implication reasoner, function initialization

Candidate approaches (3)

personal favorite on long term

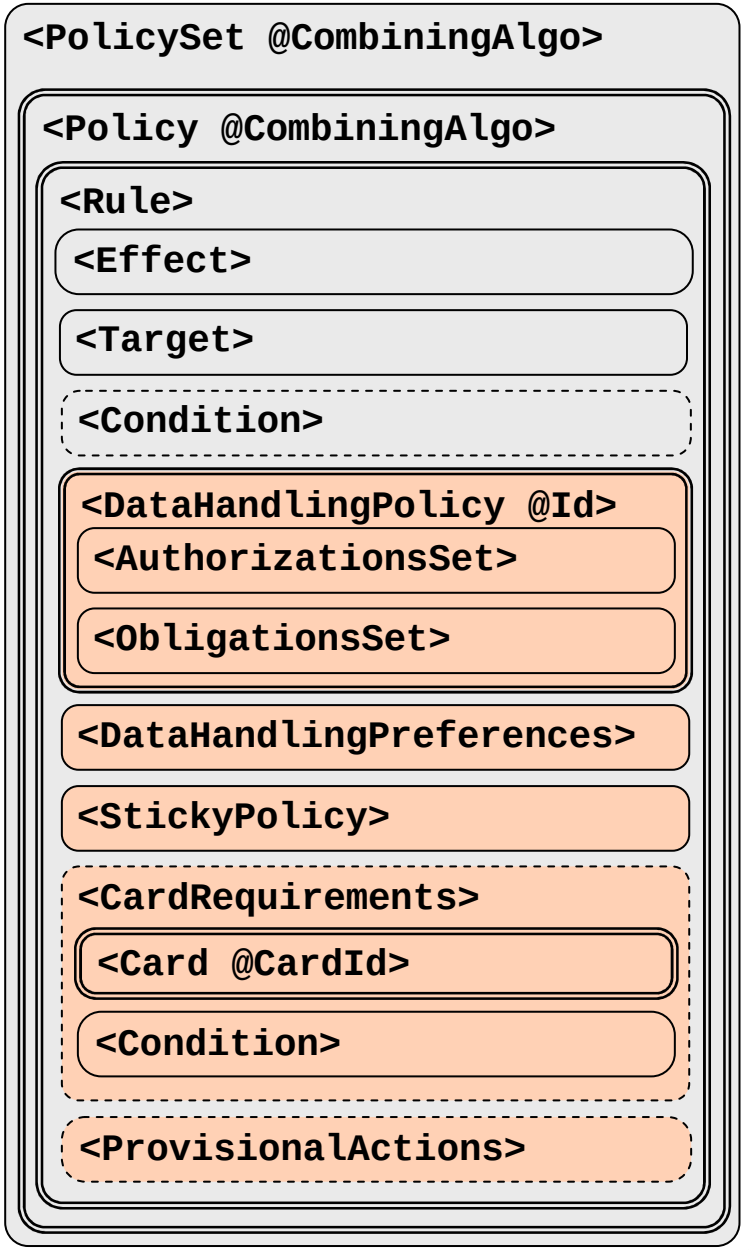
Approach 4: the full monty

- Asserted predicates embedded in request context
- Request lowest-boolean or highest-same-issuer predicates
- Evaluation by implication reasoner
- Triggered by modified PDP code

Very invasive in schema/architecture

Need implication reasoner

PPL policy format



Proposed data handling policies for revealed attributes

Requested authorizations

Promised obligations

Preferences how target resource should be treated

Agreed-upon sticky policy for target resource

Card-based access control for target resource

Cards to be presented

Required condition over card attributes

Actions to be performed, e.g., reveal attribute under referenced DHP, sign statement, limited spending,…

PPL claims format

One assertion per card, plus cross-card assertion

Reference to sticky policy associated to attribute value

New statement type to carry sticky policies

New statement type to carry conditions over attributes

New statement type to carry other (non-XML-signature) types of card evidence