Technical Overview II

Overview

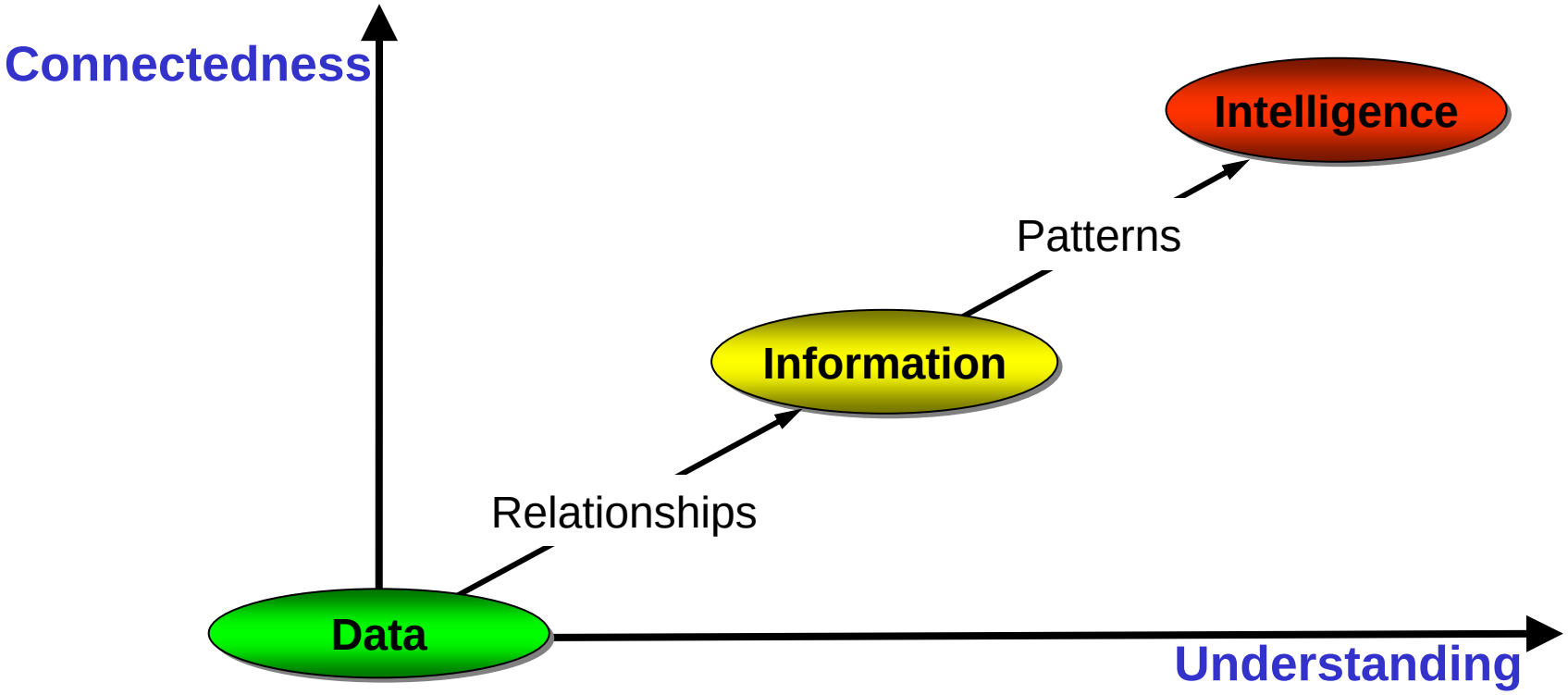
- Data Associations
- Relationship Mechanisms
- Type Augmentation

Data Associations

Why Are Relationships Essential?

- Data is inherently nonrelational/nonhierarchical
  - Essentially just a collection of objects
- Unrelated data has low value
  - Bart Simpson
  - Sideshow Bob
- Properly related data is invaluable
  - Sideshow Bob has a restraining order preventing his approach to Bart Simpson's residence

Data Is Just Data



Relationship Mechanisms

- NIEM has 4 basic mechanisms for relating data
  - Hierarchical Inclusion of Data
  - Associations
  - Roles
  - Metadata

- The last 3 rely on "XML Referencing"
  - Use XML ID/IDREF/IDREFS constructs

Hierarchical Inclusion

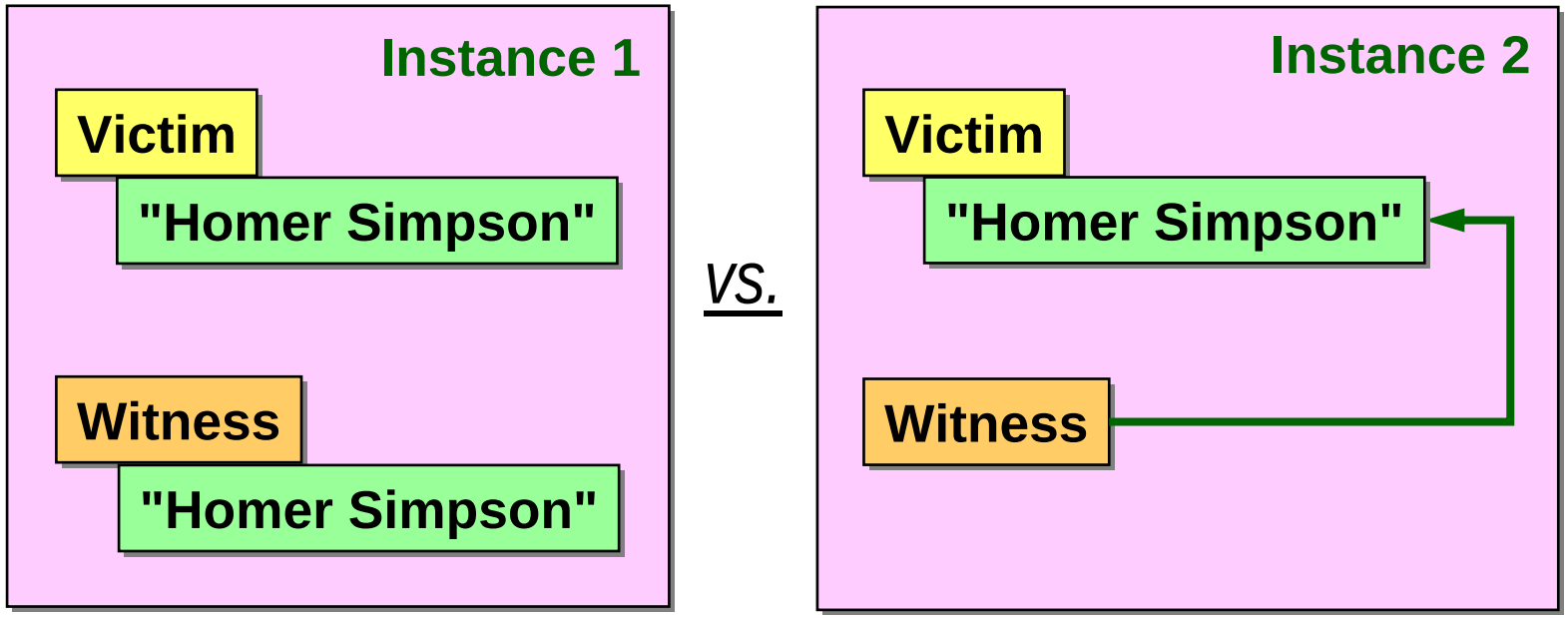
- Associated data is imbedded within another element
  - Simple to interpret
  - Simple to process
  - Most people are familiar with it
- Used in 2 different ways in NIEM 1.0
  - Relates conceptually similar types
    - E.g., PersonType and PersonNameType
  - Relates disparate types that are non-person related
    - E.g., c:IncidentType and c:IncidentLocation

Example of Hierarchical Inclusion

```
<c:Incident/>
    <u:ActivityDescriptionText>
        Something happened
    </u:ActivityDescriptionText>
    <c:IncidentLocation>
        <u:LocationAddress>
            <u:AddressFullText>
                742 Evergreen Terrace
            </u:AddressFullText>
            <u:LocationCityName>Springfield</u:LocationCityName>
        </u:LocationAddress>
    </c:IncidentLocation>
</c:Incident>
```

XML Referencing

- Avoids redundancy
- Conveys equivalence



NIEM Types Implementing Referencing

```
<attribute name="id" type="ID"/>
<attribute name="metadata" type="IDREFS"/>
<attribute name="ref" type="IDREF"/>

<complexType name="ComplexObjectType" abstract="true">
    <attribute ref="s:id"/>
    <attribute ref="s:metadata"/>
    <attribute ref="s:linkMetadata"/>
</complexType>

<complexType name="MetadataType" abstract="true">
    <attribute ref="s:id"/>
</complexType>

<complexType name="ReferenceType" final="#all">
    <attribute ref="s:id"/>
    <attribute ref="s:ref"/>
    <attribute ref="s:linkMetadata"/>
</complexType>
```

Define attributes resolving to ID, IDREF, or IDREFS

NIEM Reference Types

Representation Term of the name identifies this as being a reference type in NIEM

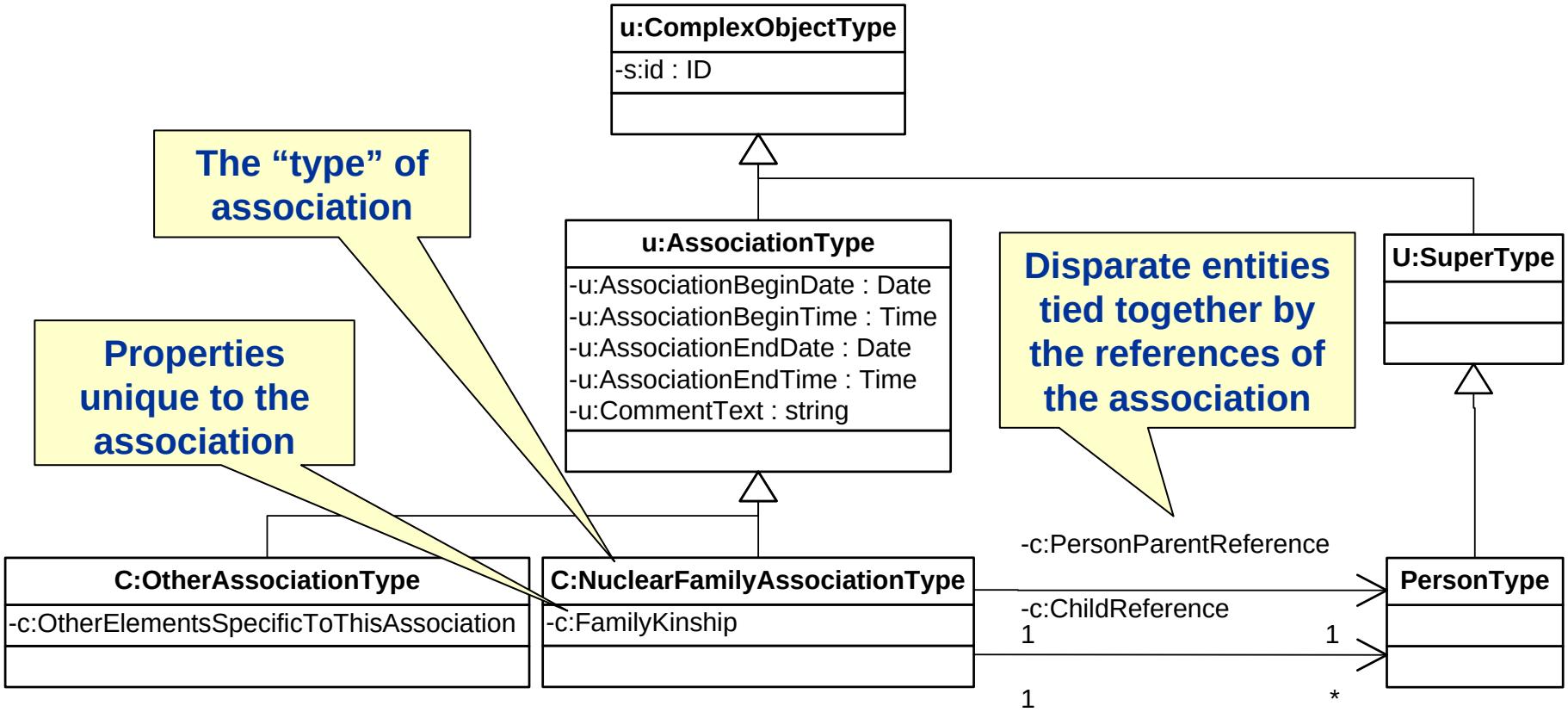
```
<xsd:element name="EmployeeReference" type="s:ReferenceType"/>
```

ReferenceType specifies that ref attribute will be used to reference another element

```
<EmployeeReference ref ="A unique reference in the instance"/>
```

Must reference an ID in the same instance

NIEM Association Structure (partial)

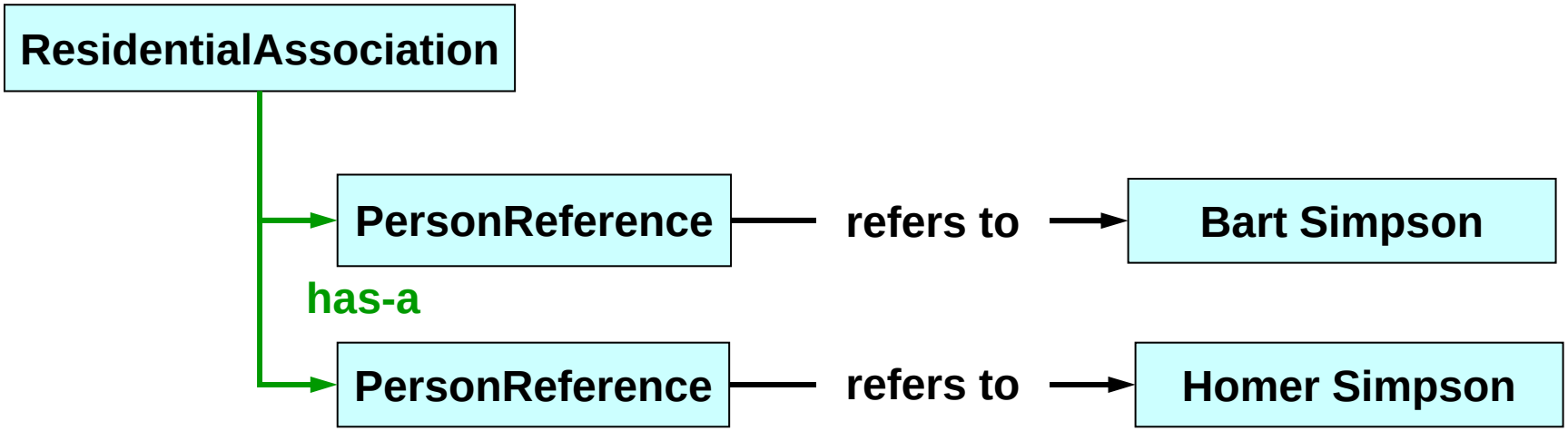


Associations Example

```
<c:NuclearFamilyAssociation>
    <c:PersonParentReference ref="A1"/>
    <c:PersonChildReference ref="B1"/>
</c:NuclearFamilyAssociation >
<u:Person id="A1" >
    <u:PersonName>
        <u:PersonGivenName>Homer</u:PersonGivenName>
        <u:PersonSurName>Simpson</u:PersonSurName>
    </u:PersonName>
</u:Person>
<u:Person id="B1" >
    <u:PersonName>
        <u:PersonGivenName>Bart</u:PersonGivenName>
        <u:PersonSurName>Simpson</u:PersonSurName>
    </u:PersonName>
</u:Person>
```

Practical Exercise: Associations

- Both Bart and Homer reside at 742 Evergreen Terrace, Springfield
  - Using the previous example as a guide, write the NIEM XML instance.



0:00

Solution

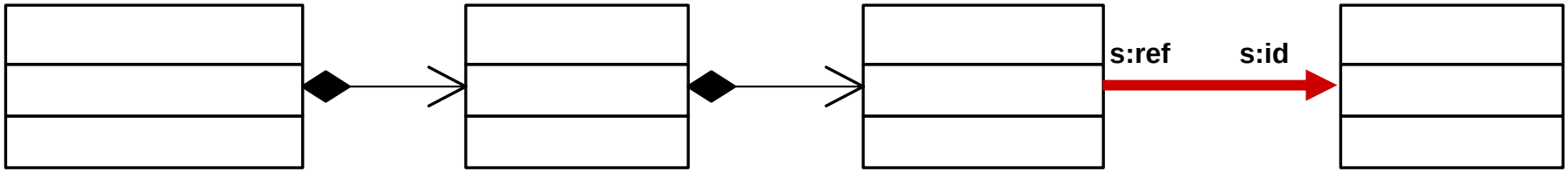
```
<u:Person id="A1">
    <u:PersonName>
        <u:PersonGivenName>Homer</u:PersonGivenName>
    </u:PersonName>
</u:Person>
<u:Person id="B1">
    <u:PersonName>
        <u:PersonGivenName>Bart</u:PersonGivenName>
    </u:PersonName>
</u:Person>
<u:Location id="L1">
    <u:LocationAddress>
        <u:AddressFullText>742 Evergreen Terrace</u:AddressFullText>
        <u:LocationCityName>Springfield</u:LocationCityName>
    </u:LocationAddress>
</u:Location>
<c:ResidentialAssociation>
    <u:PersonReference ref="A1"/>
    <u:PersonReference ref="B1"/>
    <u:LocationReference ref="L1"/>
</c:ResidentialAssociation>
```

Roles

- Representation of traits related to an entity for a specific period of time
  - Most prevalent examples relate to people
    - Project Manager
    - Friend
    - Instructor
    - Spouse
  - The number of roles that an entity "plays" at the same time is not limited

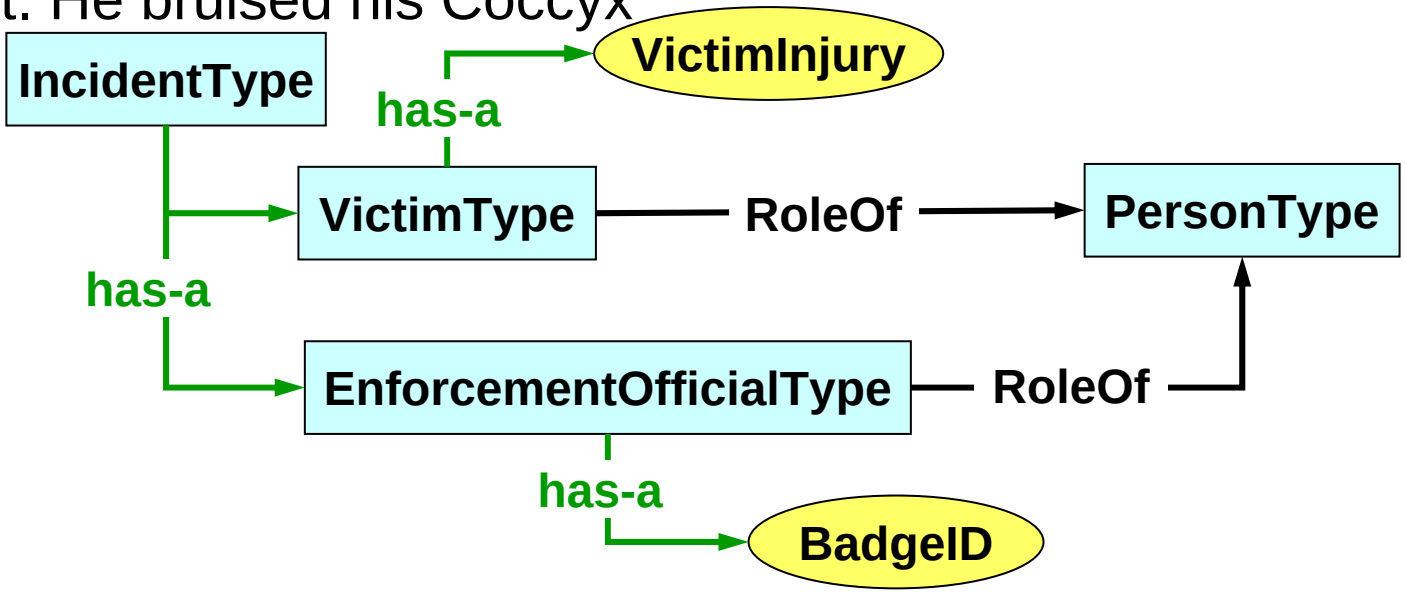
RoleOf Structure

- Roles are implemented using substitution groups
- Entities considered roles are stand-alone types
  - Contain either:
    - **RoleOfPerson**
    - **RoleOfOrganization**, or
    - **RoleOfProperty**
  - References the relevant target at run-time



RoleOf Example

- Chief Clancy Wiggum slipped on a bar of soap thrown by Bart Simpson when responding to a service call by Apu Nahasapeemapetilon for trying to steal a candy bar from the Kwik-E-Mart. He bruised his Coccyx

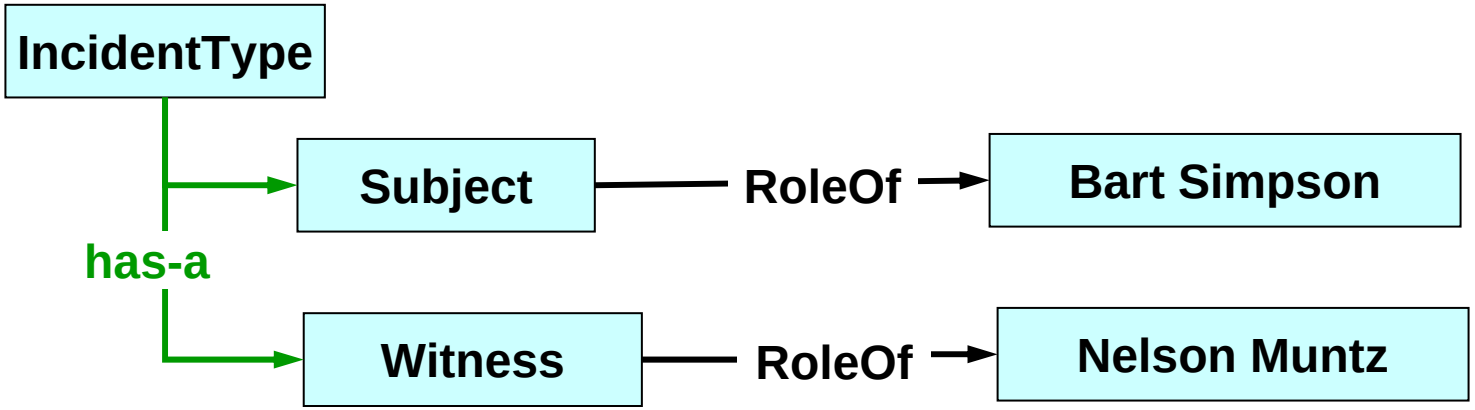


Role Example XML Instance

```
<u:Person s:id="P1">
  <u:PersonName>
    <u:PersonGivenName>Clancy</u:PersonGivenName>
    <u:PersonSurName>Wiggum</u:PersonSurName>
  </u:PersonName>
  <u:PersonBirthDate>1960-10-07</u:PersonBirthDate>
</u:Person>
<j:Incident s:id="I1">
  <j:IncidentResponseOfficial s:id="RO1"/>
    <u:RoleOfPersonReference s:ref="P1"/>
    <u:EnforcementOfficialBadgeID>
      <j:IDType>
        <u:ID>421</u:ID>
      </j:IDType>
    </u:EnforcementOfficialBadgeID>
  </j:IncidentResponseOfficial>
  <j:IncidentVictim s:id="V1">
    <u:RoleOfPersonReference s:ref="P1"/>
    <j:VictimInjury>
     <c:InjuryTypeText>Bruised Coccyx</j:IDType>
    </j:VictimInjury>
  </j:IncidentVictim>
</u:Incident>
```

Practical Exercise

- Add to the previous example by writing out the NIEM XML Instances associating Bart Simpson as the Subject. Chief Wiggum also took a statement from Nelson Muntz, who saw the whole thing.



0:00

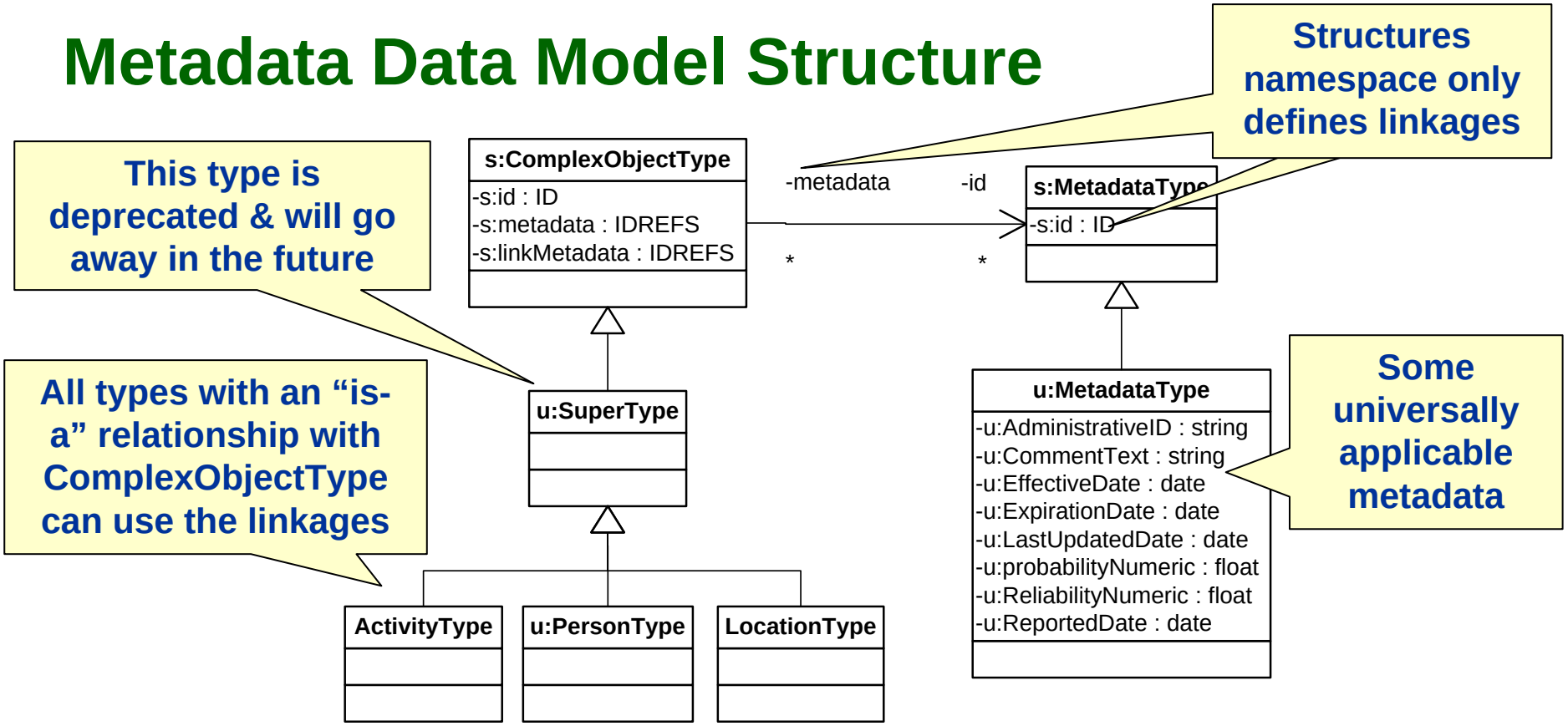
Solution

```
<j:Person s:id="P2">
    <u:PersonName>
        <u:PersonGivenName>Bart</u:PersonGivenName>
        <u:PersonSurName>Simpson</u:PersonSurName>
    </u:PersonName>
</j:Person>
<j:Person s:id="P3">
    <u:PersonName>
        <u:PersonGivenName>Nelson</u:PersonGivenName>
        <u:PersonSurName>Muntz</u:PersonSurName>
    </u:PersonName>
</j:Person>
<j:Incident s:id="I1">
  <!– the other incident related stuff from the previous example -->
  <j:IncidentSubject s:id="S1">
    <u:RoleOfPersonReference s:ref="P2"/>
  </j:IncidentSubject>
  <j:IncidentWitness s:id="W1">
    <u:RoleOfPersonReference s:ref="P3"/>
  </j:IncidentWitness>
</j:Incident>
```

Metadata

- Is data about data, or data about information
  - Typically captures info on how to …
    - Retrieve, navigate, etc.
      - e.g., topical information such as name, class, or category
    - Assess data pedigree or origin
      - e.g., accuracy, precision, reliability, belief, timeliness
- Is context dependent!
  - Data in one exchange may be metadata in another

Metadata Data Model Structure

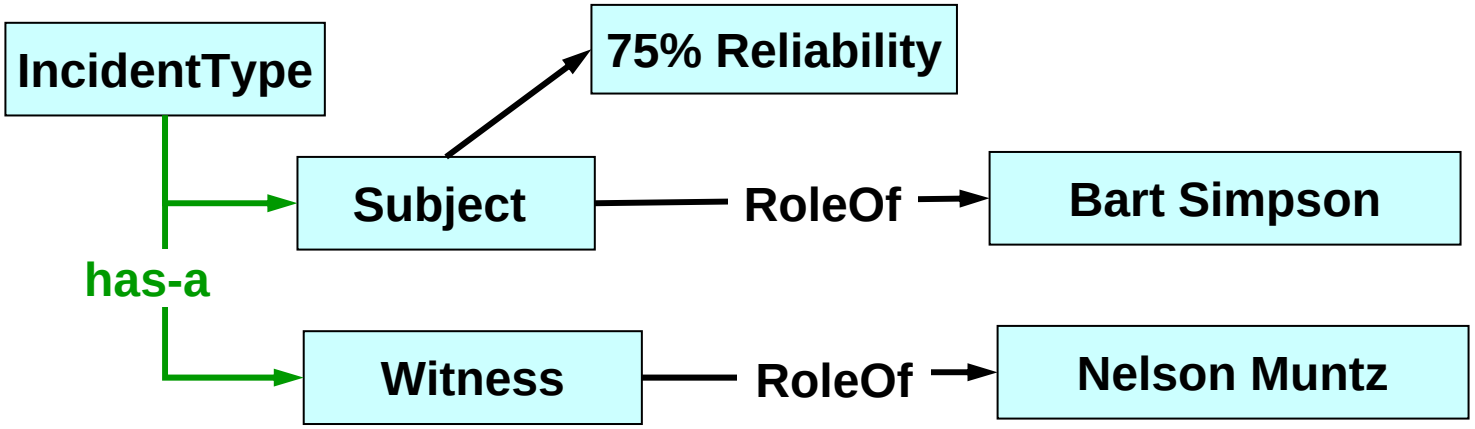


Metadata Instance Example

```
<u:Person s:metadata="M1 M2">
   <u:PersonName>
      <u:PersonGivenName>Adam</u:PersonGivenName>
      <u:PersonSurName>Brooks</u:PersonSurName>
   </u:PersonName>
   <u:PersonBirthDate>1960-10-07</u:PersonBirthDate>
</u:Person>
<u:SuperTypeMetadata s:id="M1">
   <u:ReportedDate>2005-08-01</u:ReportedDate>
</u:SuperTypeMetadata>
<my:Metadata s:id="M2">
   <my:DatabaseID>2829019291</my:DatabaseID>
</my:Metadata>
```

Practical Exercise

- During his interview, Nelson states that he wasn't sure that who he saw was really Bart because he had an obstructed view and is only 75% sure it was Bart. Add metadata describing the uncertainty.



0:00

Solution

```
<j:IncidentSubject s:id="S1" s:metadata="M2">
   <u:RoleOfPersonReference s:ref="P2"/>
</j:IncidentSubject>

<u:Metadata s:id="M2">
   <u:ReliabilityNumeric>0.75</u:ReliabilityNumeric>
</u:Metadata>
```

Type Augmentation

In This Section Students Will Learn To

- Understand the need for Type Augmentation
- Understand the mechanism used to implement Type Augmentation
- Learn how to extend objects via Type Augmentation

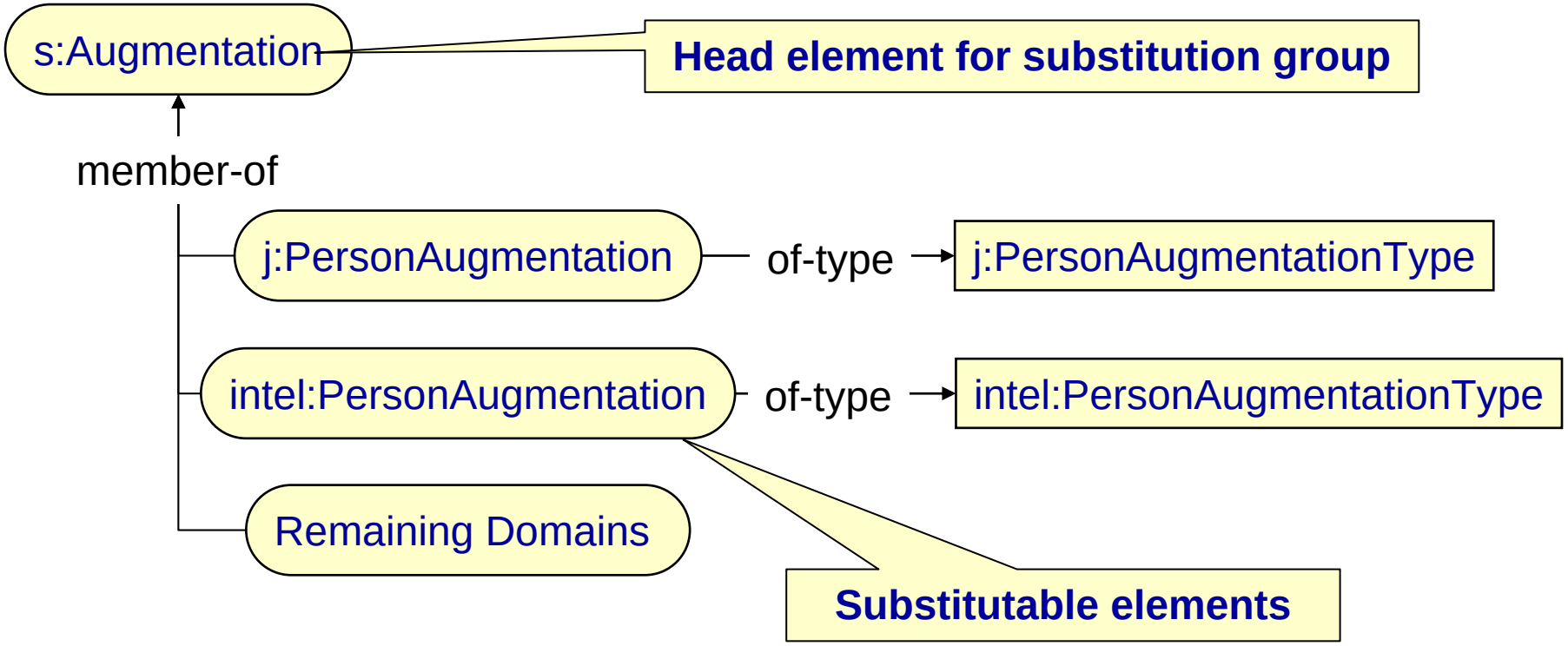
Purpose

- To enable reuse of type extensions that occur within one domain to be used in another
  - Minimizes duplicate extensions in multiple domains
  - In Object Oriented terms, it is a means of addressing the multiple inheritance problem

Type Augmentation

- Groups domain elements into "Augmentation" groups
  - Augmentation groups are really Substitution groups, specific to a domain
- Objects in an IEPD can then extend base types by pulling in Augmentations from multiple domains
  - More like a Person with a bag of "Justice things" and a bag of "Intel things"
  - Less like a special "Justice Person" or "Intelligence Person"

NIEM Structure for PersonAugmentation



Specific Example: IEPD Schema

```
<complexType name="IEPDPersonType">
 <complexContent>
  <extension base="c:PersonType">
   <sequence>
    <group ref="j:PersonAugmentation"/>
    <group ref="intel:PersonAugmentation"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>

<element name="IEPDPerson“ type="iepd:IEPDPersonType"/>
```

Specific Example: IEPD Instance

```
<iepd:IEPDPerson>
  <c:PersonBuildText>Husky</c:PersonBuildText>
  <!-- More Common Person-related elements -->
  <j:PersonAugmentation>
    <j:PersonDrivingInsuranceStatusText>
            None at all
    </j:PersonDrivingInsuranceStatusText>
    <!-- More Justice Person-related elements -->
  </j:PersonAugmentation>

  <intel:PersonAugmentation>
    <intel:OccupationName>NIEM Trainer</intel:OccupationName>
    <!-- More Intel Person-related elements -->
  </intel:PersonAugmentation>

</iepd:IEPDPerson>
```

Augmentation Practical Exercise

- Suppose your exchange requires a Person object with the following properties:
  - Person Name
  - Hair Color
  - FBI ID Number
  - Person Function
- Which augmentation element(s) would be used to extend the base Person object?

0:00

Practical Exercise Solution

- j:PersonFBIID is contained in **j:PersonAugmentation**
- it:PersonFunctionText is contained in **it:PersonAugmentation**

Bonus: What would the extension schema look like in order to apply the above augmentations to the base Person object? *(keep in mind the requirements from the previous slide)*

Extension Schema

```
<xsd:complexType name="PersonType">
   <xsd:complexContent>
      <xsd:extension base="c:PersonType">
         <xsd:sequence>
            <xsd:element ref="j:PersonAugmentation"/>
           <xsd:element ref="it:PersonAugmentation"/>
           </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Person" type="loc:PersonType">
```

creative commons
COMMONS DEED

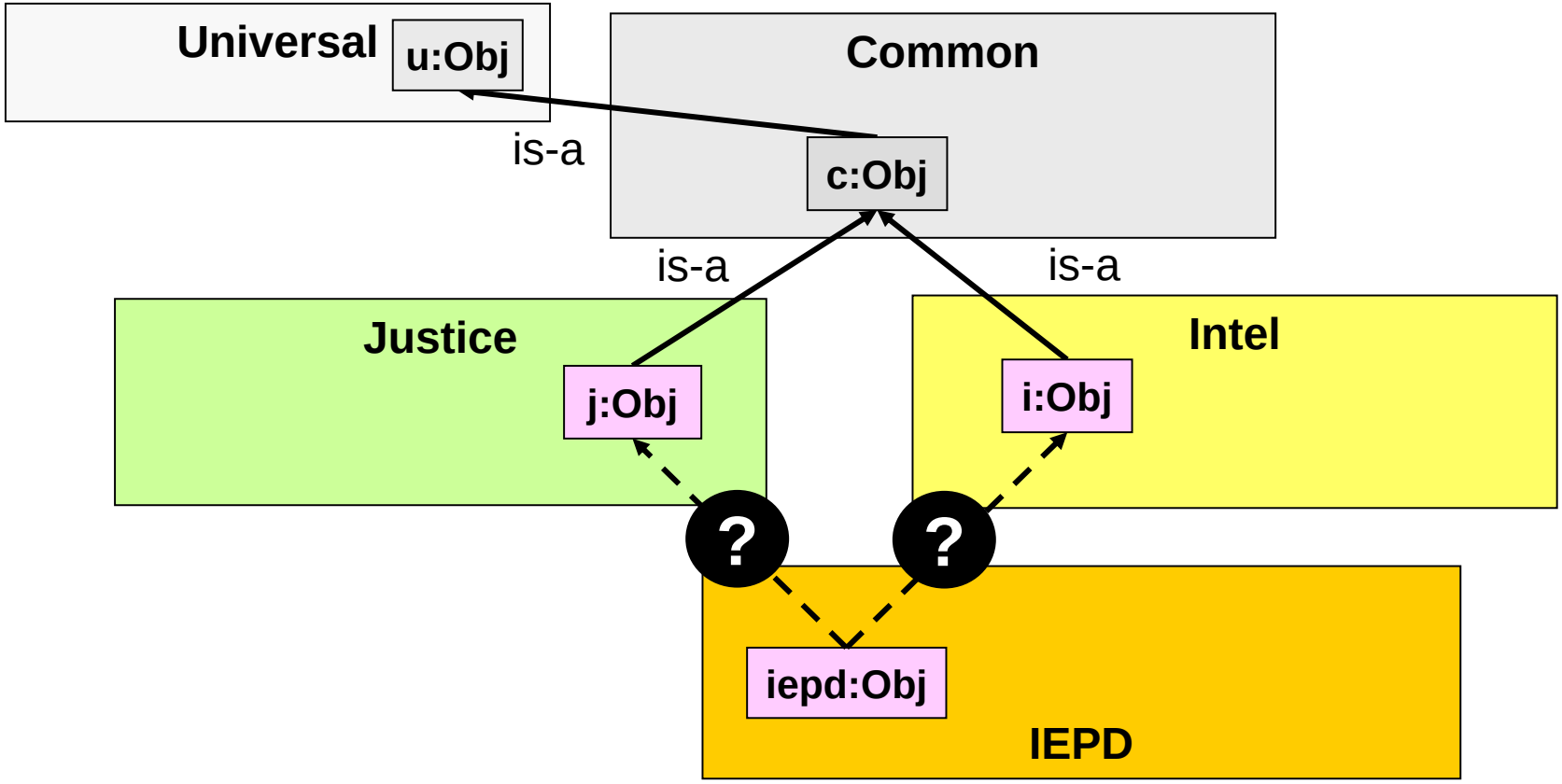
This work is licensed under the Creative Commons Attribution-ShareAlike 2.5 License.
To view a copy of this license
a) visit http://creativecommons.org/licenses/by-sa/2.5/; or,
b) send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA."

Backup Slides

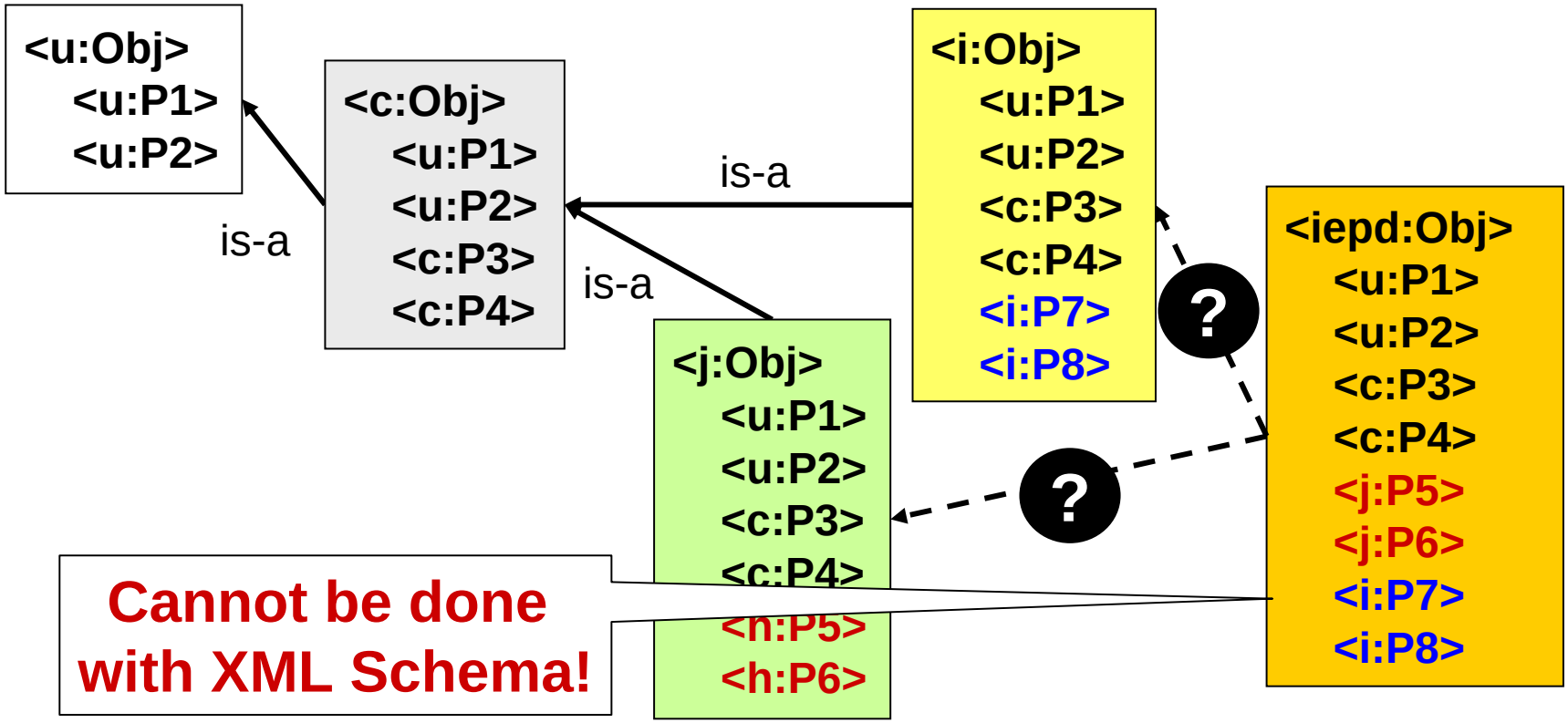
The Problem With Specialization

- Specialization creates a new object as a new concept, which isn't always accurate:
  - No such thing as an "Intel Person"
- Can't combine properties from multiple domains into one base object
- Can't specialize from multiple domains
- Hard to reuse domain-specific extensions because extensions are defined at the element level, not the domain level

From whom do we specialize?



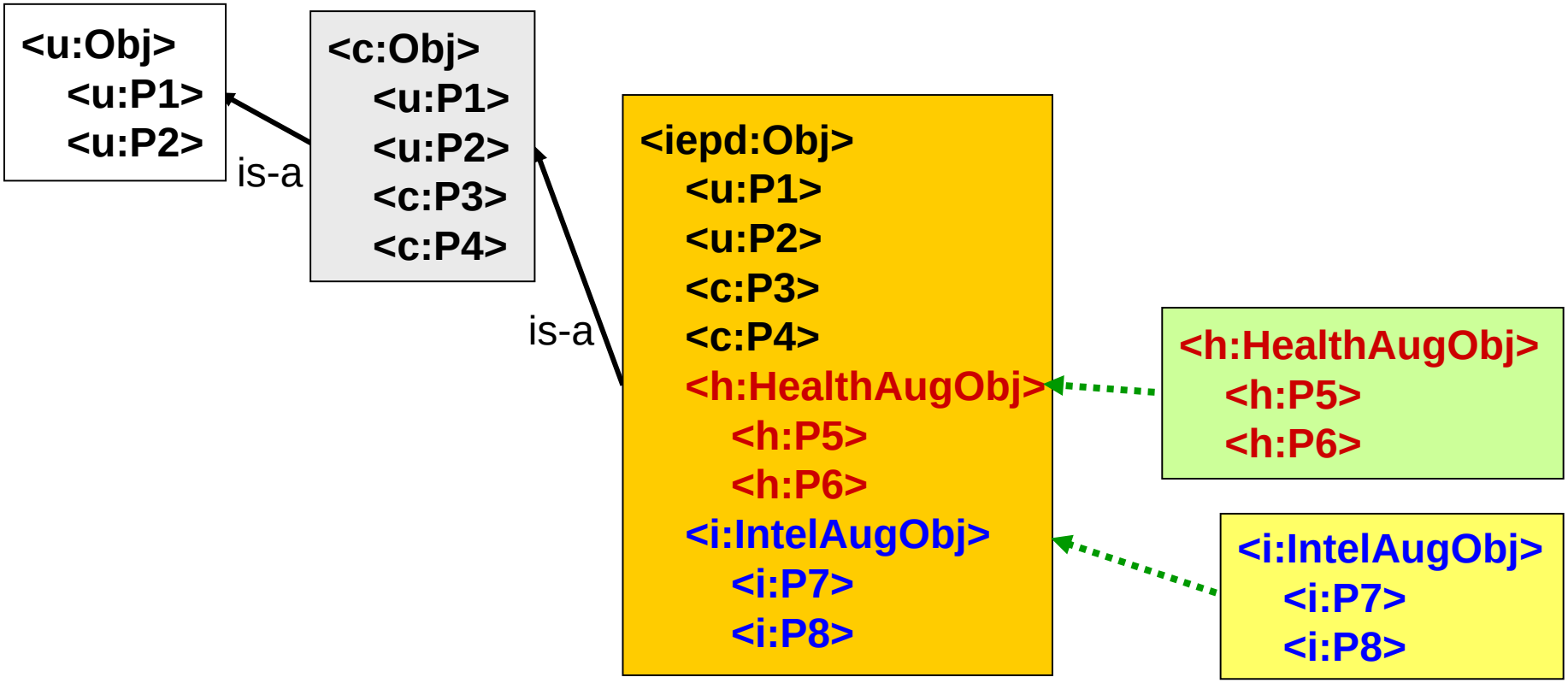
Can't Specialize From Both Domains



A Solution: Type Augmentation

- Groups domain elements into "Augmentation" groups
- Augmentation groups are really Substitution groups, specific to a domain
- Objects in an IEPD can then extend base types by pulling in Augmentations from multiple domains
- More like a Person with a bag of "Justice things" and a bag of "Intel things"
- Less like a special "Justice/Intel Person"

Adding "Augmentations" from Both Domains



Generic Example: IEPD Schema

```
<complexType name="IEPDDerivedType">
  <complexContent>
    <extension base="base:BaseType">
      <sequence>
        <element ref="domain:DomainAugmentationContainer"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element
  name="IEPDDerivedContainer"
  type="iepd:IEPDDerivedType"
  substitutionGroup="base:BaseContainer"/>
```

Generic Example: IEPD Instance

```
<iepd:IEPDDerivedContainer>
  <base:Property1/>
  <base:Property2/>
  <base:Property3/>
  <domain:DomainAugmentationContainer>
    <domain:Property4/>
    <domain:Property5/>
    <domain:Property6/>
  </domain:DomainAugmentationContainer>
</iepd:IEPDDerivedContainer>
```