Introducción a SHACL

Mauricio Espinoza Mejía

Créditos: **Robin Keskisärkkä**
Jose Emilio Labra Gayo

Contenido

- Qué es validación
- Cómo encaja SHACL en la tecnología de lenguajes basados en lógica
- Comprensión básica de las características principales de SHACL

¿Por qué describir y validar modelos?

- Para productores
  - Los desarrolladores pueden comprender los contenidos que van a producir.
  - Pueden asegurarse de producir la estructura esperada
  - Generar interfaces
  - ….
- Para consumidores
  - Entender los contenidos
  - Verificar la estructura antes de procesarla
  - Generación y optimización de consultas

Tecnologías similares

Tecnología
Bases de Datos Relacionales
XML
JSON
RDF

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=...>
  <xs:element name="note">
   CREATE TABLE employees (
     id int NOT NULL,
     department int NOT NULL,
     CONSTRAINT emp_pk PRIMARY KEY (id),
     CONSTRAINT emp_dept_fk
        FOREIGN KEY department
        REFERENCES departments
   );
</xs:schema>
```

Nuestro objetivo es llenar ese vacío

Validación en RDFs, OWL

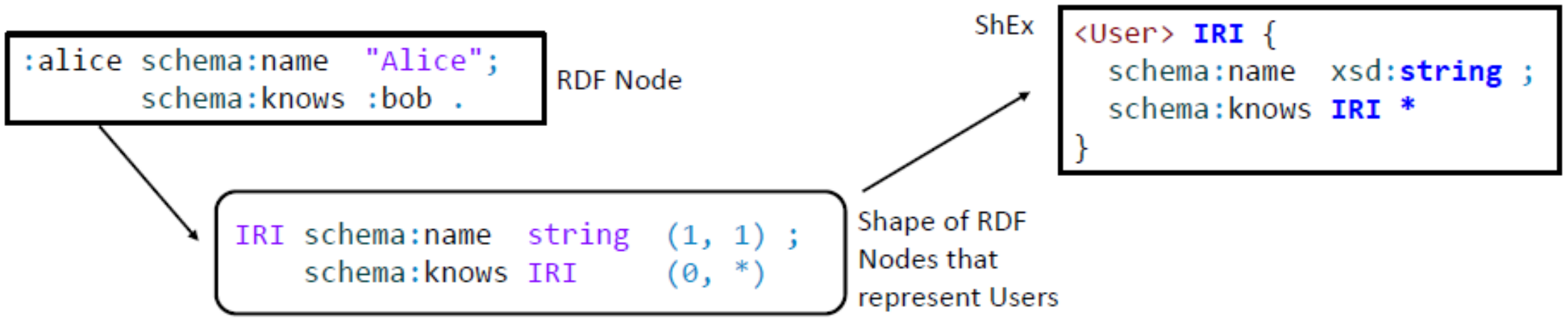
- Ontología: Persona ⊑ ∃tienePadre.Persona
- ABox: Persona (Mauricio)

  Persona (Jorge)

  tienePadre (Mauricio, Jorge)
- ¿Esto "valida"? ¡No hay información sobre el padre de Jorge!
- Podemos inferir que ∃tienePadre.Persona(Jorge), es decir, tiene un padre
- ¡OWL no se puede utilizar para la validación!
- OWL y RDFS son buenos para agregar axiomas faltantes, sin detectar que faltan

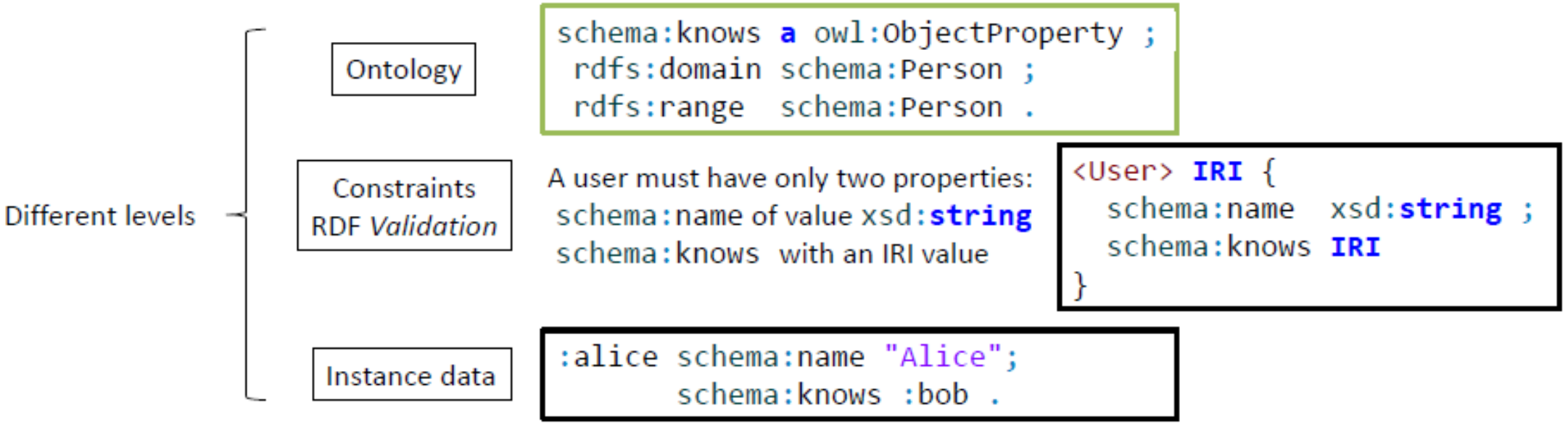
Entendiendo el problema

- RDF está compuesto por nodos y arcos entre nodos
- Podemos describir / comprobar
  - forma del propio nodo (restricción de nodo)
  - número de posibles arcos entrantes / salientes de un nodo
  - posibles valores asociados con esos arcos



Entendiendo el problema

- Validación RDF ≠ definición de ontología ≠ datos de instancia
  - Las ontologías suelen centrarse en entidades del mundo real.
  - La validación de RDF se centra en las características del grafo RDF (nivel inferior)



Entendiendo el problema

- Flexibilidad RDF
  - Uso mixto de objetos y literales
  - schema:creator puede ser una cadena o schema:Person en los mismos datos

```
:angie schema:creator "Keith Richards" ,
       [ a schema:Person ;
           schema:singleName "Mick" ;
           schema:lastName "Jagger"
       ] .
```

Entendiendo el problema

- Propiedades repetidas
  - A veces, la misma propiedad se utiliza para diferentes propósitos en los mismos datos
  - Ejemplo: un registro de libro debe tener 2 códigos con estructura diferente

```
:book schema:productID "isbn:123-456-789";
      schema:productID "code456" .
```

ShEx and SHACL

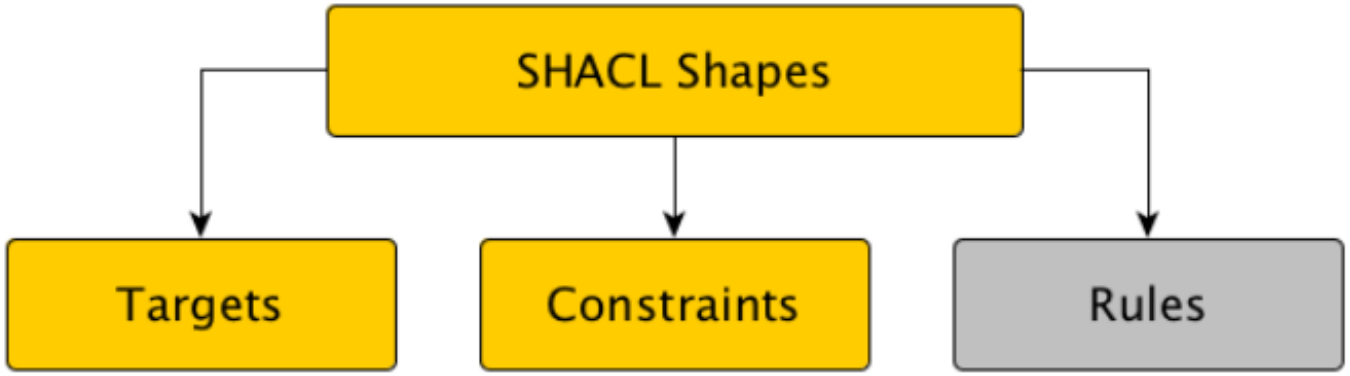
- Taller de validación RDF 2013
  - Conclusiones del taller:
    - Existe la necesidad de un lenguaje conciso de mayor nivel para la validación RDF
    - ShEx propuesto inicialmente por Eric Prud'hommeaux

- 2014 W3C Data Shapes WG autorizado
- 2015 SHACL como entregable del WG
- W3C recommendation: https://www.w3.org/TR/shacl/(July2017)
  - Inspirado por SPIN, OSLC & pedazos of ShEx
  - 2 partes:
    - SHACL-Core, SHACL-SPARQL

¿Qué es SHACL?

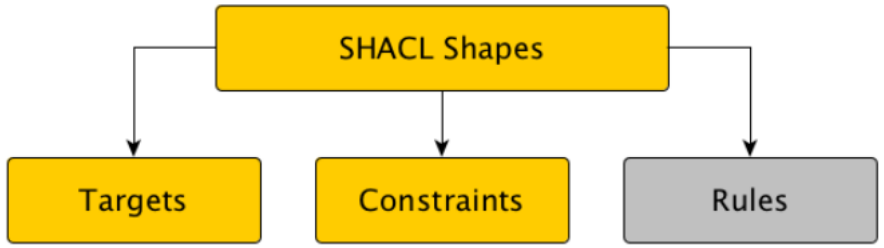
- Shapes Constraint Language
- Permite validar los datos RDF contra *shapes* (formas) ...

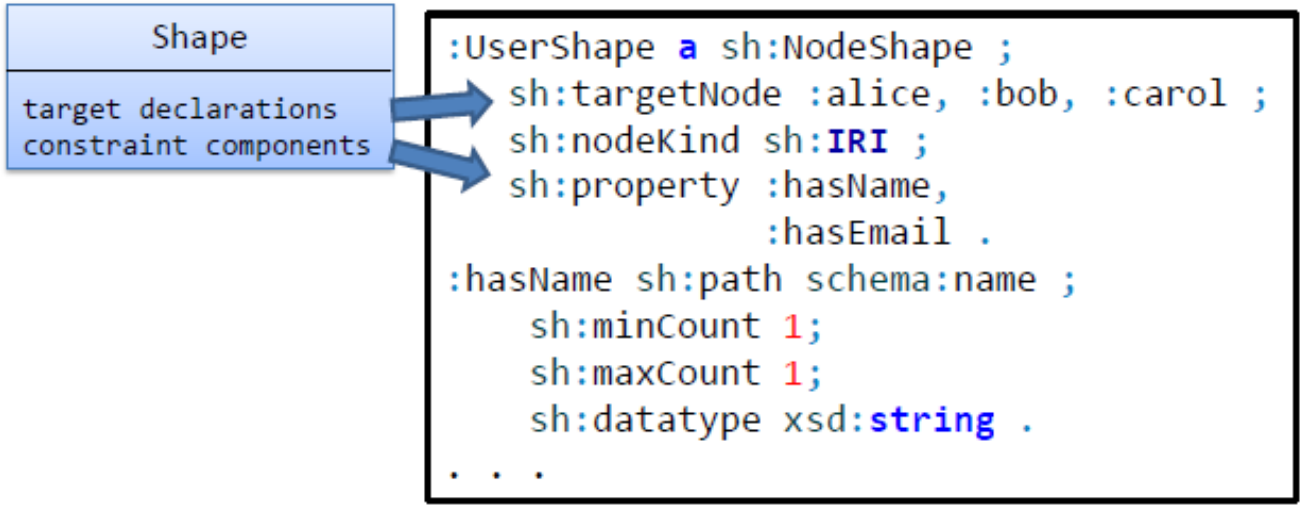
  y más (generación de interfaz de usuario, inferencia, extensiones SPARQL, etc.)
- Usa formato Turtle
  - Los paréntesis denotan listas o colecciones



¿Qué es SHACL?

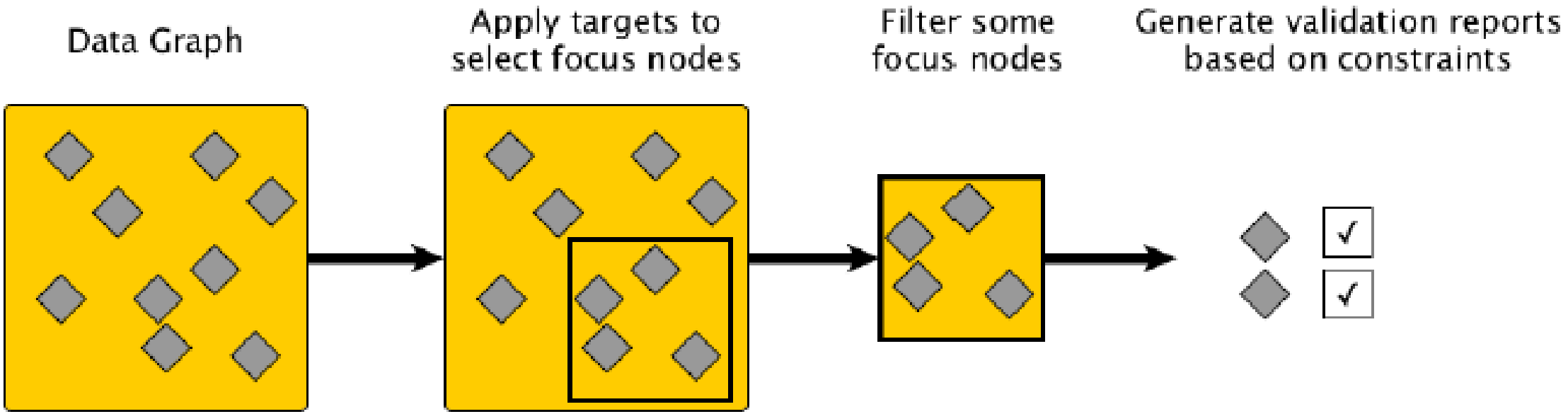
- Shape: colección de targets y componentes de constraints
- Targets: especificar qué nodos en el grafo de datos deben ajustarse a una forma
- Constraints: determinar cómo validar un nodo





Proceso de validación

Data graph: un grafo RDF que contiene datos para ser validados
Shape graph: un grafo RDF que contiene shapes



Ejemplo

```
prefix :       <http://example.org/>
prefix sh:     <http://www.w3.org/ns/shacl#>
prefix xsd:    <http://www.w3.org/2001/XMLSchema#>
prefix schema: <http://schema.org/>

:UserShape a sh:NodeShape ;
   sh:targetNode :alice, :bob, :carol ;
   sh:nodeKind sh:IRI ;
   sh:property :hasName,
               :hasEmail .
:hasName sh:path schema:name ;
    sh:minCount 1;
    sh:maxCount 1;
    sh:datatype xsd:string .
:hasEmail sh:path schema:email ;
   sh:minCount 1;
   sh:maxCount 1;
   sh:nodeKind sh:IRI .
```

Shapes graph


```
:alice schema:name "Alice Cooper" ;
       schema:email <mailto:alice@mail.org> .

:bob   schema:firstName "Bob" ;
       schema:email <mailto:bob@mail.org> .

:carol schema:name "Carol" ;
       schema:email "carol@mail.org" .
```

Data graph

Ejemplo

```
prefix :       <http://example.org/>
prefix sh:     <http://www.w3.org/ns/shacl#>
prefix xsd:    <http://www.w3.org/2001/XMLSchema#>
prefix schema: <http://schema.org/>

:UserShape a sh:NodeShape ;
   sh:targetNode :alice, :bob, :carol ;
   sh:nodeKind sh:IRI ;
   sh:property [
    sh:path schema:name ;
    sh:minCount 1; sh:maxCount 1;
    sh:datatype xsd:string ;
  ] ;
  sh:property [
   sh:path schema:email ;
   sh:minCount 1; sh:maxCount 1;
   sh:nodeKind sh:IRI ;
  ] .
```

Shapes graph

```
:alice schema:name "Alice Cooper" ;
       schema:email <mailto:alice@mail.org> .

:bob   schema:firstName "Bob" ;
       schema:email <mailto:bob@mail.org> .

:carol schema:name "Carol" ;
       schema:email "carol@mail.org" .
```

Data graph

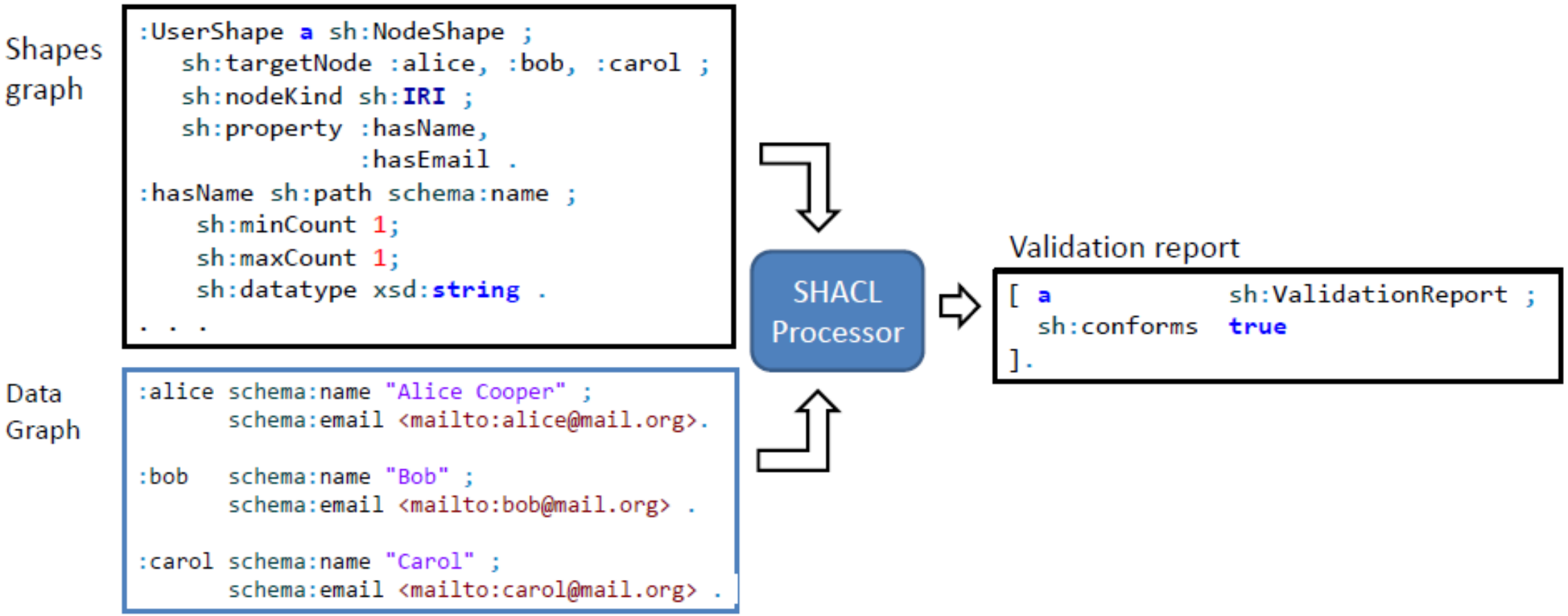
Reporte validación

- El resultado del proceso de validación es una lista de errores de infracción.
- Sin errores → RDF se ajusta al shape graph

```
[ a              sh:ValidationReport ;
  sh:conforms    true
].
```

```
[ a              sh:ValidationReport ;
  sh:conforms    false ;
  sh:result      [
   a               sh:ValidationResult ;
   sh:focusNode    :bob ;
   sh:message
     "MinCount violation. Expected 1, obtained: 0" ;
   sh:resultPath schema:name ;
   sh:resultSeverity sh:Violation ;
   sh:sourceConstraintComponent
     sh:MinCountConstraintComponent ;
   sh:sourceShape  :hasName
] ;
...
```

Procesador SHACL



Importación Shapes graphs

- Los procesadores SHACL siguen las declaraciones de importaciones OWL
  - Extiende el shape graph actual con shapes importadas

```
:UserShape a sh:NodeShape ;
   sh:targetNode :alice, :bob, :carol ;
   sh:nodeKind sh:IRI ;
   sh:property :hasName .
:hasName sh:path schema:name ;
    sh:minCount 1;
    sh:maxCount 1;
    sh:datatype xsd:string .
```

```
<> owl:imports <http://example.org/UserShapes> .

:TeacherShape a sh:NodeShape;
  sh:targetClass :Teacher ;
  sh:node :UserShape ;
  sh:property [
   sh:path :teaches ;
   sh:minCount 1;
   sh:datatype xsd:string;
 ] .
```

Resumen

- Las ontologías no son buenas para la validación.
  - Las ontologías expresan hechos sobre el dominio.
  - Restricciones, modelos de datos, etc., expresan hechos sobre los datos

- Se han explorado varios enfoques diferentes
- Uno de ellos, SHACL, se ha convertido en una recomendación del W3C
- Construido alrededor de restricciones que deben verificarse

Tutorial

Questions?