

技術勉強会 2023/02/10 オンライン

ナレッジグラフの基礎

江上周作

産業技術総合研究所
人工知能研究センター
データ知識融合研究チーム

ナレッジグラフ（知識グラフ）とは



- 実世界の知識を蓄積し，伝達することを目的とした「データのグラフ」であり，そのノードは関心のあるエンティティを表し，エッジはこれらのエンティティの間の様々な関係を表す
- その「データのグラフ」は，グラフベースのデータモデルに準拠し、エッジラベル付き有向グラフ、異種グラフ、プロパティグラフなどがある
 - 引用：A. Hogan, et al. *Knowledge Graphs*, ACM Computing Survey, vol.54, no.4, pp. 1–37, ACM, 2021
- ナレッジグラフは、情報を取得してオントロジーに統合し、推論器を適用して新しい知識を導き出す
 - 引用：Ehrlinger et al. *Towards a Definition of Knowledge Graphs*, SEMANTiCS (Posters, Demos, SuCCESS), vol.48, pp.1–4, CEUR, 2016

ナレッジグラフの記述方法

Webでナレッジグラフを共有するための標準形式

RDF (Resource Description Framework) とは

すべての情報を

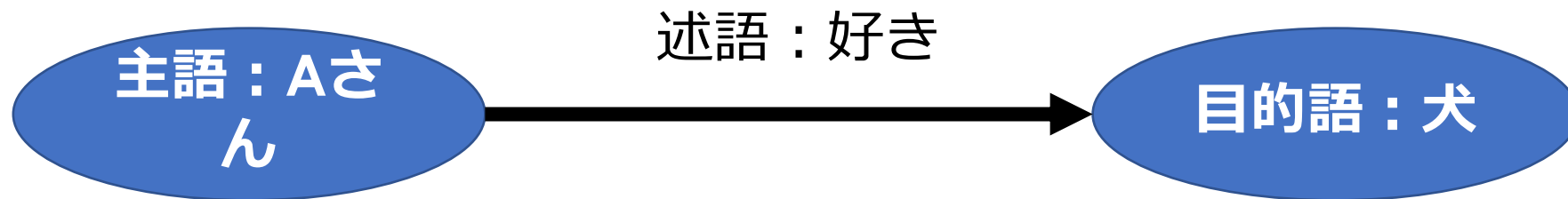
- ・ 主語 (リソース)
- ・ 述語 (プロパティ)
- ・ 目的語 (リソースorリテラル)

の三組み (トリプルという) で記述する方法を提供

その実体は、ラベル付き有向グラフ

(※)

情報 「Aさんは犬が好き」

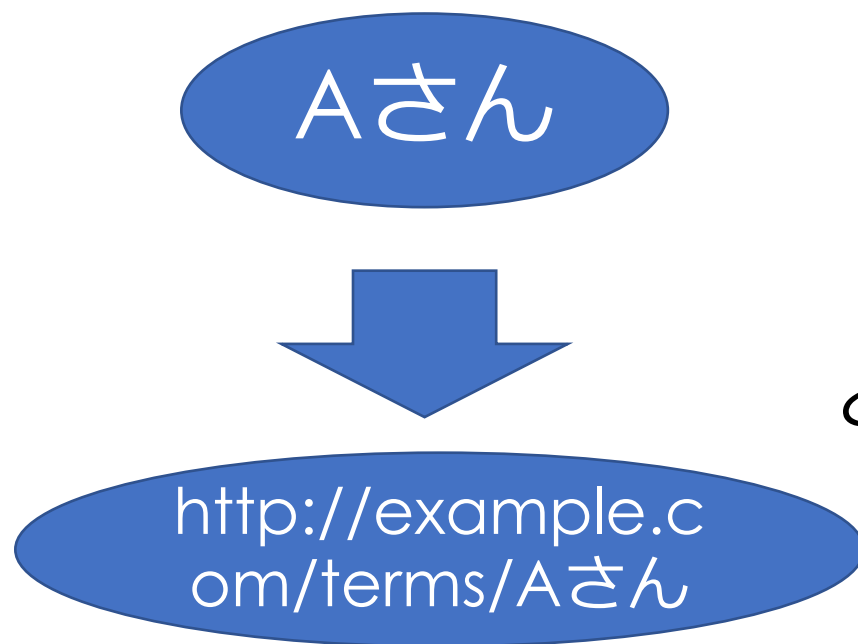


このようにグラフ形式で図示したものをRDFグラフと呼ぶ

URIで識別可能な全ての事物

- ・ **人物，書籍，イベントなどの実世界の物事**
- ・ **趣味，嗜好，信頼性などの事柄など...**

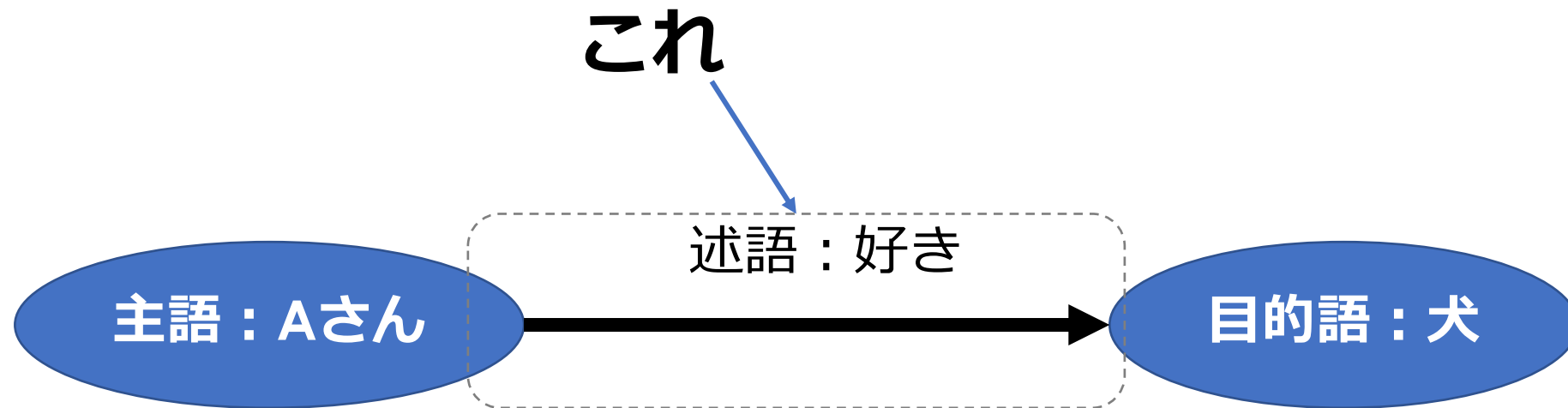
つまり、事物に**一意なURI**を割り当ててしまえば、
RDFでそれはリソースとして認識される



「Aさん」という事物に
「`http://example.com/terms/Aさん`」
というURIを与えよう

これがリソース

※本当はブラウザでアクセスできることが望ましいが
とりあえず今は重複がなければこれでよい（後で紹介）



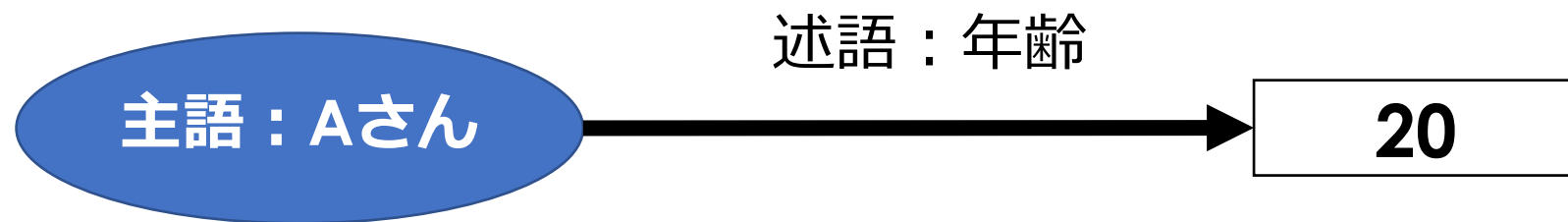
プロパティにも一意なURIを割り当てる

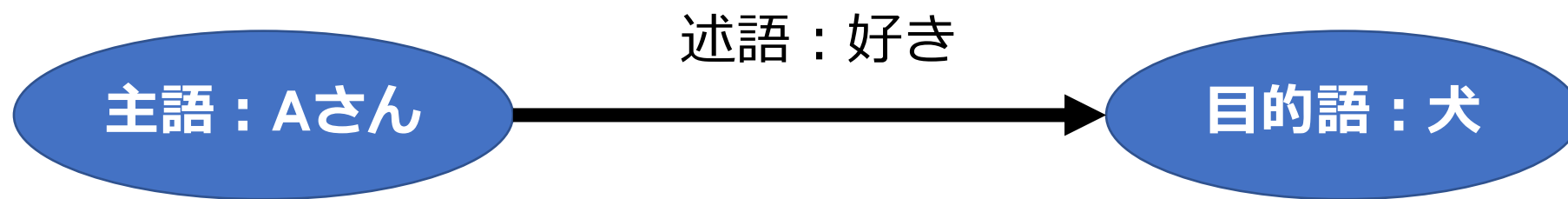
<http://example.com/terms/好き>

文字列や**数値**のこと。

プロパティを持たない。

URIを割り当てない。





↑これを

XML, Turtle, JSON-LD等で書くことができる。

記述方法が異なるが、結局RDFグラフとして図示する
と全部同じになることが特徴！（= 情報の統一化）

RDFのシリアライゼーションフォーマット

- RDFはリソースをURIで識別し, 主語・述語・目的語のトリプル形式で関係を記述するデータモデルである
 - リソースを説明するメタデータが付与される

ラベル付き有向グラフの形でリソースを表現できることはわかったが、具体的にコンピュータでどのような形式で処理をするの？

- N-triples, Turtle, RDF/XML, RDF/JSON, RDFa, JSON-LD など様々なシリアライゼーション(変換)フォーマットが存在

相互に変換可能

<https://www.easyrdf.org/converter>

- <URI> をスペースで分けて並べる最も原始的な記法。
- トリプルはピリオドで区切る。
- 拡張子は.ntとすることが多い
- CSVからの変換が楽

<http://ja.dbpedia.org/resource/電気通信大学> <http://www.w3.org/2000/01/rdf-schema#label> "電気通信大学" .

<http://ja.dbpedia.org/resource/電気通信大学> <http://dbpedia.org/ontology/country> <http://ja.dbpedia.org/resource/日本> .

XMLタグの入れ子構造でRDFグラフを記述している！

【名前空間】

以降で青文字部分を「rdf:」に省略できる（カプセル化）

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<rdf:RDF
```

```
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
```

```
  <rdf:Description rdf:about="http://example.com/terms/Aさん">
```

```
    <http://example.com/terms/好き>
```

名前空間定義していないからURIをフルスペルで書いてる

```
      <rdf:Description rdf:about="http://example.com/terms/犬" />
```

```
    </http://example.com/terms/好き>
```

```
  </rdf:Description>
```

緑文字はリソースを記述する決まり文句

```
</rdf:RDF>
```

ファイルの拡張子は「.rdf」とするのが慣習。

XMLエディタで開きたい場合は「.xml」

Terse RDF Triple Language (Turtle)



XML形式は幅広いシステムで利用可能だけど長い！

【名前空間】

以降で青文字部分を「ex:」に省略できるよ

@prefix ex: <http://example.com/terms/> .

<http://example.com/terms/Aさん>

ex:好き ex:犬, ex:猫 ;

ex:誕生日 "2012-01-01T00:00:00"^^xsd:dateTime .

ピリオド忘れずに！

省略しない場合は<>でURIをくくる

同じプロパティの値が複数ある場合はカンマで列挙可

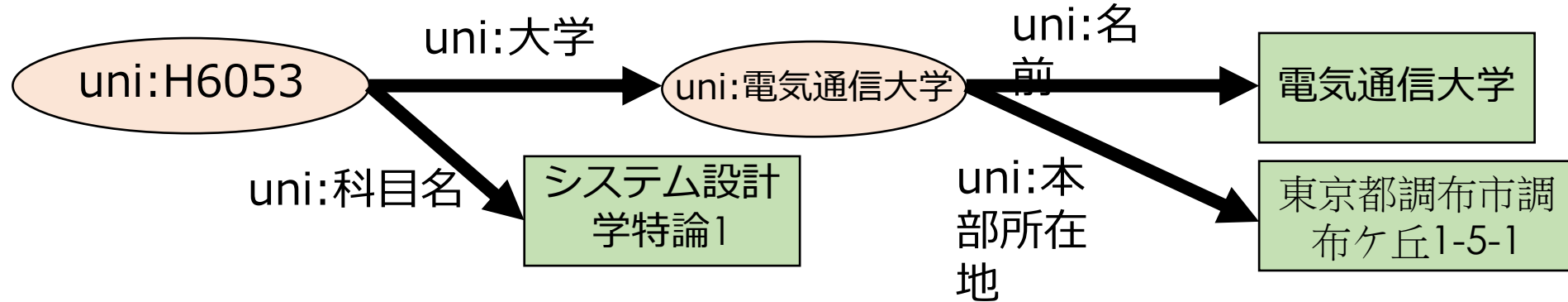
リテラル値のデータ型

**ファイルの拡張子は「.ttl」とするのが慣習。
対応しているシステムに限られるが記述が楽！**

- JSONでのデータ交換や、HTMLのscriptタグへの埋め込みに利用されている
- 拡張子は.jsonや.jsonld

```
{
  "@context": {
    "dbo": "http://dbpedia.org/ontology/",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#"
  },
  "@id": "http://ja.dbpedia.org/resource/電気通信大学",
  "dbo:country": {
    "@id": "http://ja.dbpedia.org/resource/日本"
  },
  "rdfs:label": "電気通信大学"
}
```

トリプルの連鎖の例



@prefix uni: <http://www.mydomain.org/uni-ns/> .

uni:H6053

uni:科目名 "システム設計学特論1" ;

uni:大学 uni:電気通信大学 .

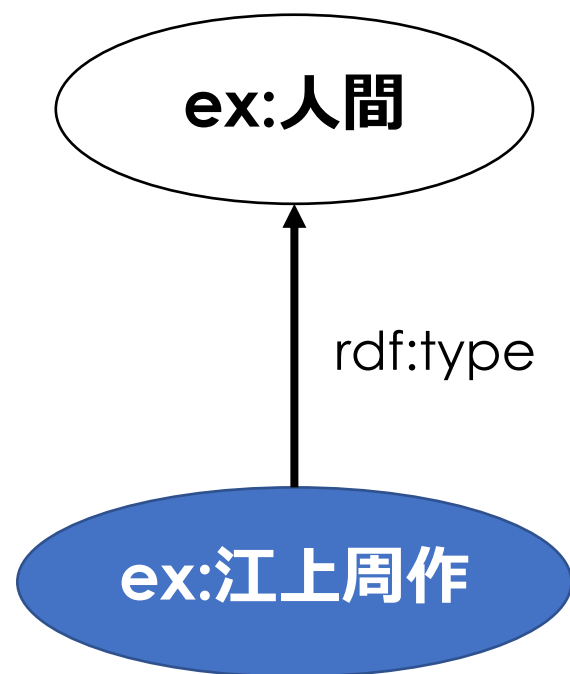
uni:電気通信大学

uni:名前 "電気通信大学" ;

uni:本部所在地 "東京都調布市調布ヶ丘1-5-1" .

○はリソース
□はリテラル

- RDF/XML文書またはRDF/XML文書が置かれたURLを入力すると、文書の妥当性の確認やトリプルの視覚化を行ってくれるサービス
- URL: <http://www.w3.org/RDF/Validator/>
- 日本語を含むトリプルを視覚化したい場合には、「Graph format」を「SVG-embedded」または「SVG-link」とする
 - SVG(Scalable Vector Graphics)は、XMLをベースとした2次元ベクターイメージ用の画像形式の一つ
- **より見やすい日本語サイト**
 - <https://www.kanzaki.com/works/2009/pub/graph-draw>

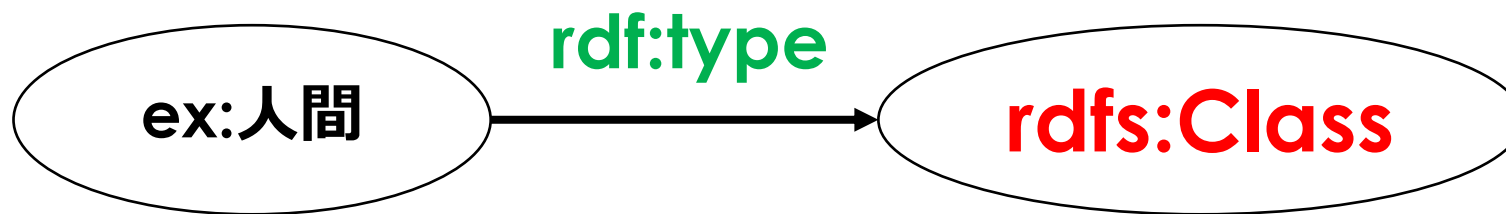


「人間」さんは実在しない。
概念, カテゴリ, グループのニュアンス。
こういうのを**クラス**という

「江上周作」は実在する。
人間を具体化させたものの一例。
人間クラスのインスタンスという。

「ex:人間」を普通のリソースではなく**クラス**として定義する！

書きたい情報「ex:人間は**クラスである**」



「rdfs」は「<http://www.w3.org/2000/01/rdf-schema#>」

RDFSはクラスを定義するための用語をRDFで提供している。
↑アクセスしてみるとRDFSのTurtleが見れる

クラスの指定には「rdf:type」を使う決まり

インスタンスの定義方法



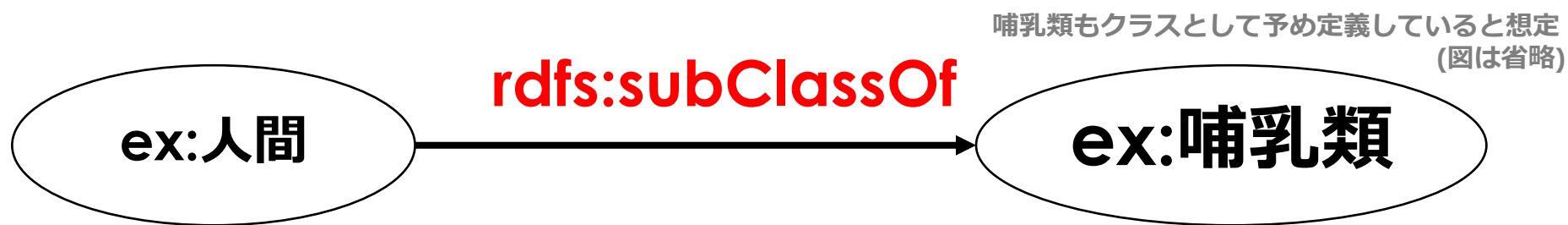
- クラスのインスタンスを定義するためには, `rdf:type`プロパティを用いる
 - リソースがクラスであることを定義するためにも, `rdf:type`プロパティを用いた
- クラスのインスタンスの集合のことを**クラスの外延**と呼ぶ
- 例: `ex:太郎` (インスタンス) は `ex:Person` (クラス) である



`ex:太郎` **`rdf:type`** `ex:Person` .

「ex:人間」を「ex:哺乳類」のサブクラスとして定義する

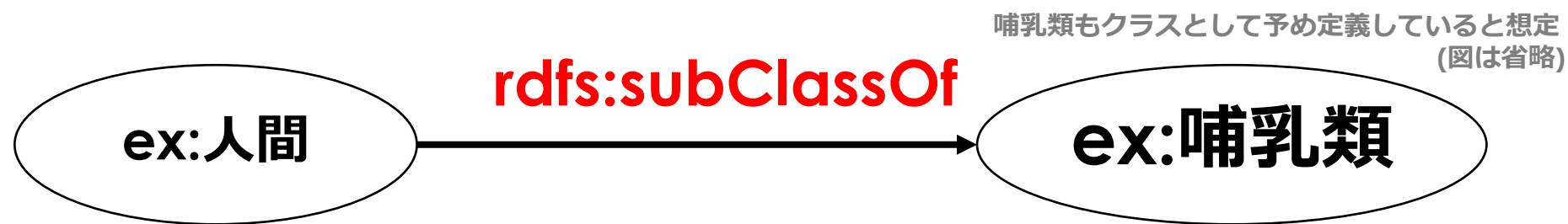
書きたい情報 「ex:人間はex:哺乳類のサブクラスである」



階層関係を意味するプロパティとして
rdfs:subClassOfが一般的に使用される。

「ex:人間」を「ex:哺乳類」のサブクラスとして定義する

書きたい情報 「ex:人間はex:哺乳類の**サブクラスである**」

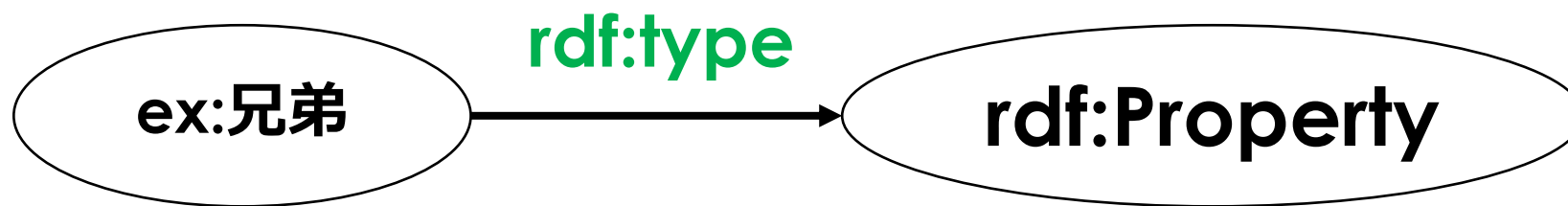


この定義をした上で

「ex:Aさん rdf:type ex:人間 .」と書くと,
「ex:Aさん rdf:type ex:哺乳類 .」を
記述しなくても当然導ける = 【推論可能】

「ex:兄弟」をプロパティとして定義する

書きたい情報 「ex:兄弟はプロパティである」



「ex:兄弟」を「ex:親族」のサブプロパティとして定義する！

書きたい情報「ex:兄弟はex:親族のサブプロパティである」



兄弟プロパティを使って「A ex:兄弟 B .」と書くと、
「A ex:親族 B .」が当然導ける = 【推論可能】