Brick: Present and Future

Dr. Gabe Fierro

gtfierro@mines.edu

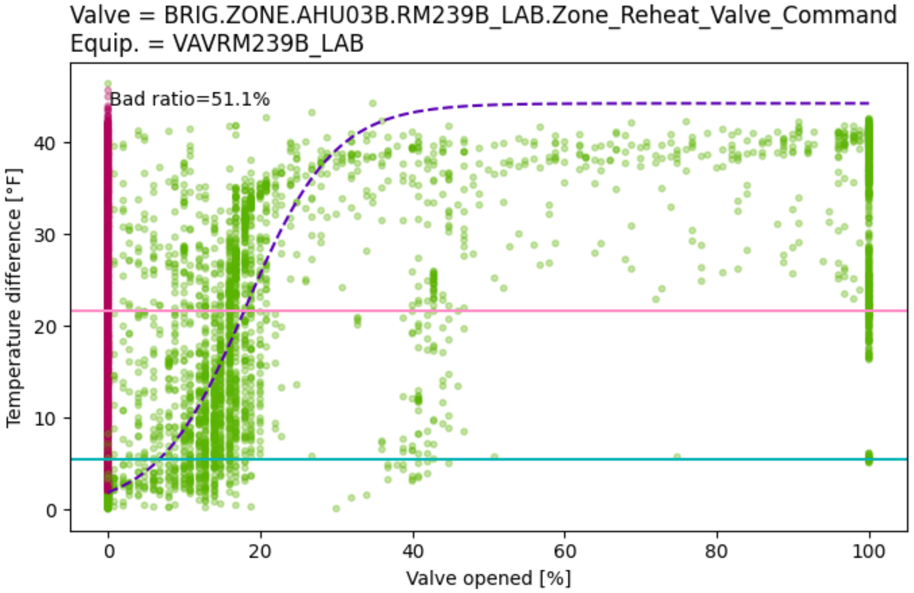
Colorado School of Mines / National Renewable Energy Laboratory

brickschema.org

COLORADOSCHOOLOFMINES
EARTH • ENERGY • ENVIRONMENT

Increasing Amounts of Building Data Available

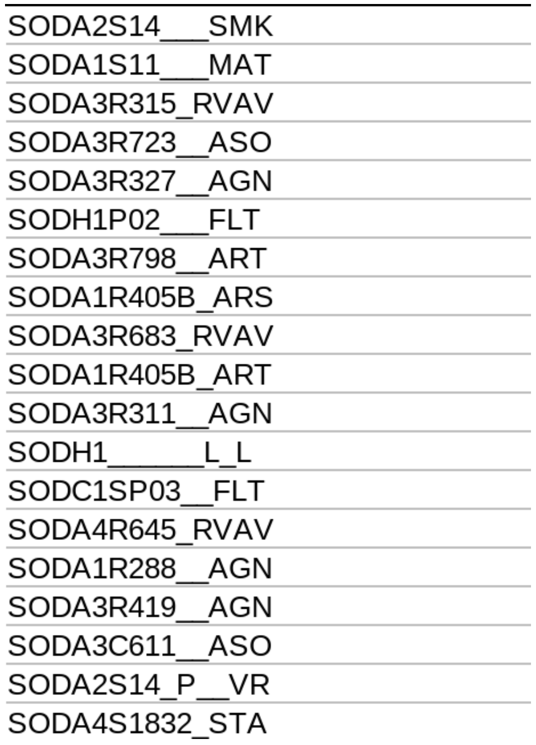
- Enable new kinds of data-driven processes:
  - Automated fault detection and diagnosis
  - Digital twins
  - Energy efficient control schemes
  - Predictive maintenance
  - etc.



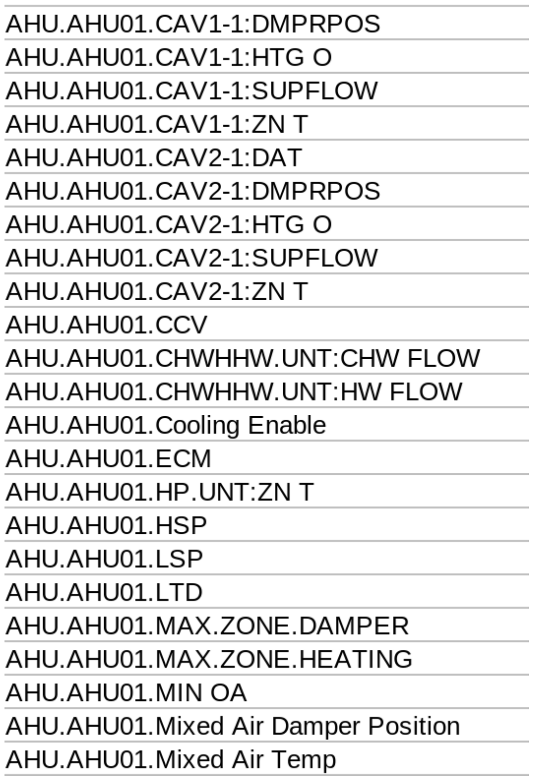
- However, such data is **hard to access, use**

- Data scientists spend ~40% time discovering, understanding data even in well-curated datasets

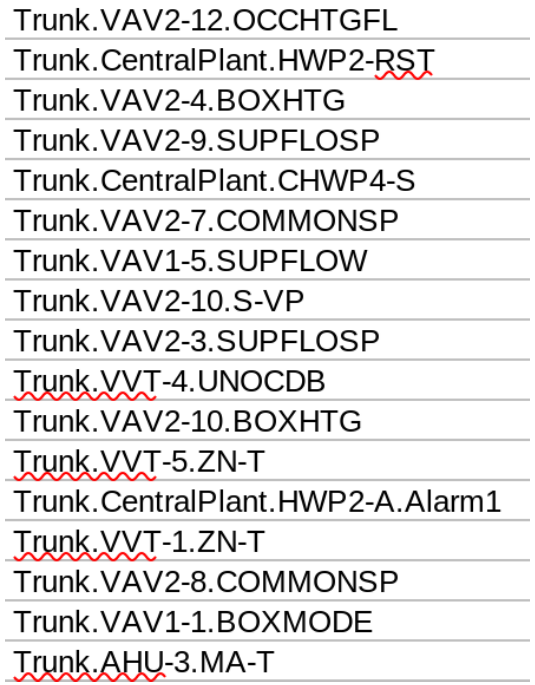
State of Building Metadata



SODA2S14___SMK
SODA1S11___MAT
SODA3R315_RVAV
SODA3R723__ASO
SODA3R327__AGN
SODH1P02___FLT
SODA3R798__ART
SODA1R405B_ARS
SODA3R683_RVAV
SODA1R405B_ART
SODA3R311__AGN
SODH1______L_L
SODC1SP03__FLT
SODA4R645_RVAV
SODA1R288__AGN
SODA3R419__AGN
SODA3C611__ASO
SODA2S14_P__VR
SODA4S1832_STA



AHU.AHU01.CAV1-1:DMPRPOS
AHU.AHU01.CAV1-1:HTG O
AHU.AHU01.CAV1-1:SUPFLOW
AHU.AHU01.CAV1-1:ZN T
AHU.AHU01.CAV2-1:DAT
AHU.AHU01.CAV2-1:DMPRPOS
AHU.AHU01.CAV2-1:HTG O
AHU.AHU01.CAV2-1:SUPFLOW
AHU.AHU01.CAV2-1:ZN T
AHU.AHU01.CCV
AHU.AHU01.CHWHHW.UNT:CHW FLOW
AHU.AHU01.CHWHHW.UNT:HW FLOW
AHU.AHU01.Cooling Enable
AHU.AHU01.ECM
AHU.AHU01.HP.UNT:ZN T
AHU.AHU01.HSP
AHU.AHU01.LSP
AHU.AHU01.LTD
AHU.AHU01.MAX.ZONE.DAMPER
AHU.AHU01.MAX.ZONE.HEATING
AHU.AHU01.MIN OA
AHU.AHU01.Mixed Air Damper Position
AHU.AHU01.Mixed Air Temp



Trunk.VAV2-12.OCCHTGFL
Trunk.CentralPlant.HWP2-RST
Trunk.VAV2-4.BOXHTG
Trunk.VAV2-9.SUPFLOSP
Trunk.CentralPlant.CHWP4-S
Trunk.VAV2-7.COMMONSP
Trunk.VAV1-5.SUPFLOW
Trunk.VAV2-10.S-VP
Trunk.VAV2-3.SUPFLOSP
Trunk.VVT-4.UNOCDB
Trunk.VAV2-10.BOXHTG
Trunk.VVT-5.ZN-T
Trunk.CentralPlant.HWP2-A.Alarm1
Trunk.VVT-1.ZN-T
Trunk.VAV2-8.COMMONSP
Trunk.VAV1-1.BOXMODE
Trunk.AHU-3.MA-T

- 3 different buildings/BMS/subsystems → 3 (or more) different labeling/naming schemes

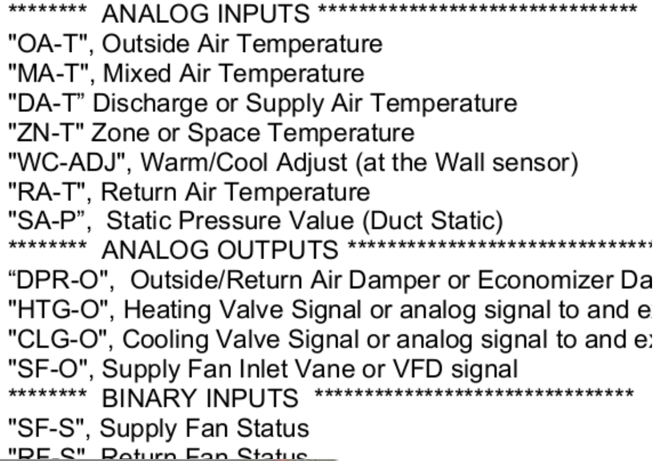
Brick

https://brickschema.org

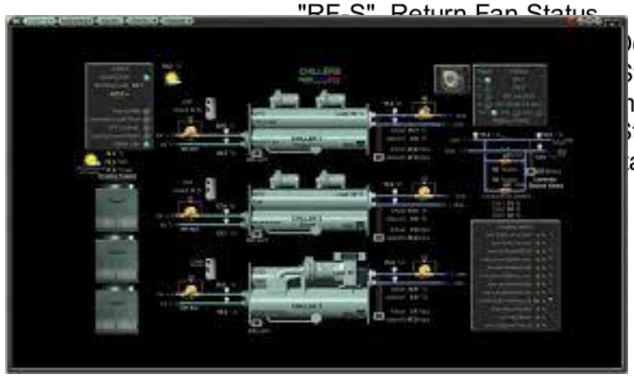
Framing the Metadata Challenge

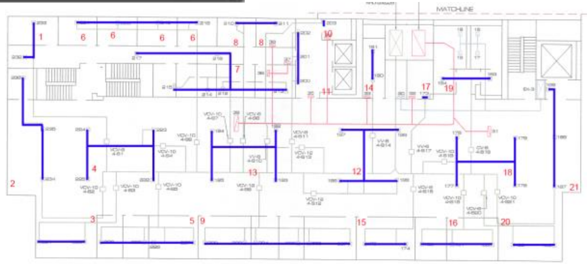
- Extreme heterogeneity
  - Proprietary, vendor-specific data “silos”
  - Every building is a custom, one-off design
  - Different BMS, equipment vendors, etc

- No common data representation:
  - Binders, BMS graphics, marked up PDF scans of blueprints, out-of-date BIM
  - Descriptions dominated by <u>informal</u> and <u>ad-hoc</u> labels
  - Convention is fine for humans, but not for machines
  - Difficult to develop interoperable software



```
********  ANALOG INPUTS ********************************
"OA-T", Outside Air Temperature
"MA-T", Mixed Air Temperature
"DA-T" Discharge or Supply Air Temperature
"ZN-T" Zone or Space Temperature
"WC-ADJ", Warm/Cool Adjust (at the Wall sensor)
"RA-T", Return Air Temperature
"SA-P",  Static Pressure Value (Duct Static)
********  ANALOG OUTPUTS ******************************
“DPR-O",  Outside/Return Air Damper or Economizer Da
"HTG-O", Heating Valve Signal or analog signal to and e
"CLG-O", Cooling Valve Signal or analog signal to and e
"SF-O", Supply Fan Inlet Vane or VFD signal
********  BINARY INPUTS  *******************************
"SF-S", Supply Fan Status
"RF-S", Return Fan Status
                              etector Status (supervisory
                              tatus (aka. Freeze Stat)
                              np Status
                              tatus
                              atus
```



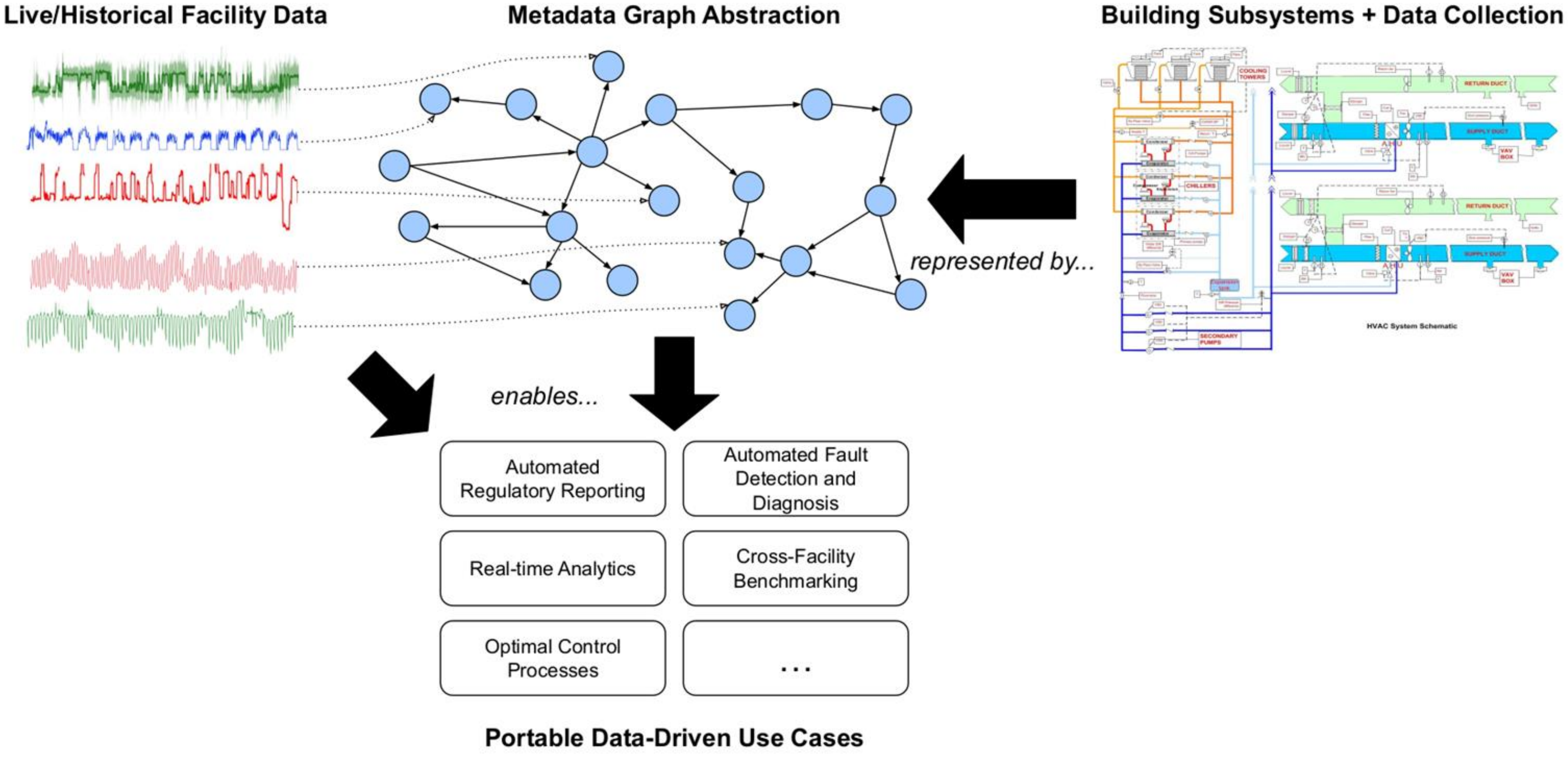


Brick's Goal: Make Working with Building Data Easier

- Most building data resides in **opaque data silos**
  - Unclear, inconsistent, hard-to-interpret labels
  - *(if you have access to it at all)*

- Existing metadata standards focus on other perspectives of the building
  - Design, construction
  - Asset management
  - Commissioning, Auditing

- Need a metadata representation designed for **data-driven building software**
  - Unlock potential of building data
  - Unify data across subsystem, vendor silos
  - Enable "write once, use on any building" smart building applications
  - *Preserve existing investments*



Live/Historical Facility Data

Metadata Graph Abstraction

Building Subsystems + Data Collection

represented by...

enables...

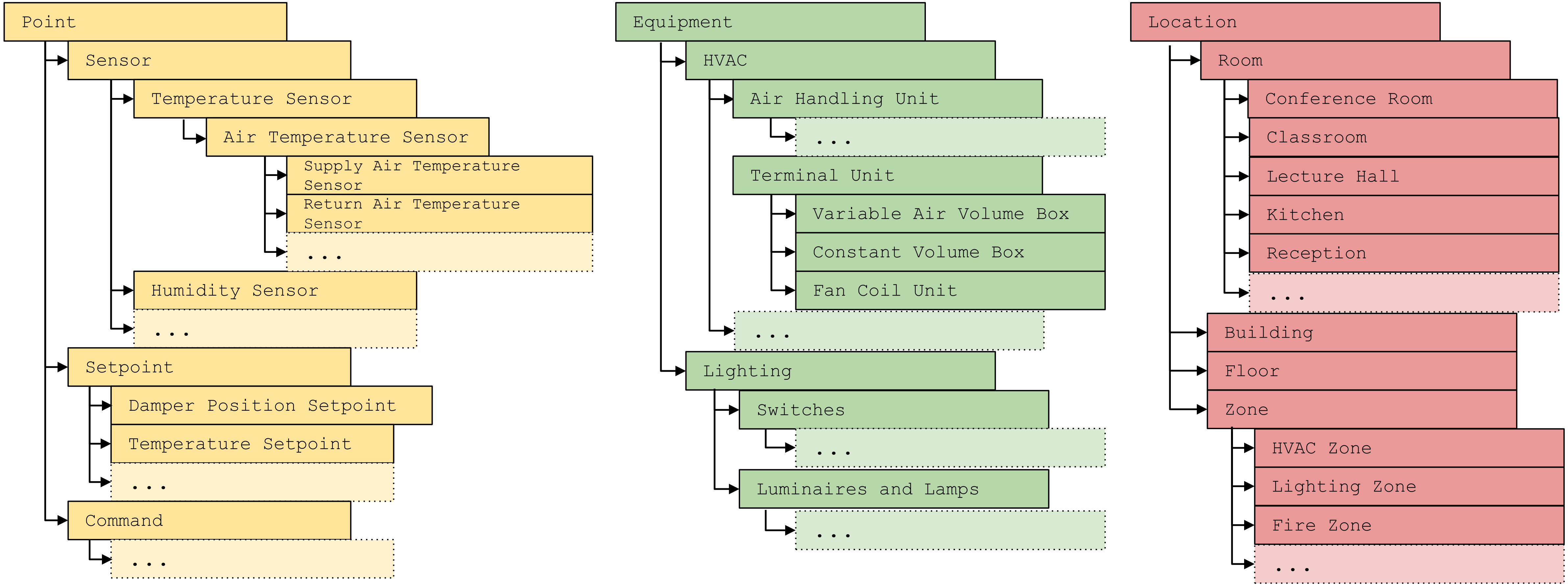
- Automated Regulatory Reporting
- Automated Fault Detection and Diagnosis
- Real-time Analytics
- Cross-Facility Benchmarking
- Optimal Control Processes
- ...

Portable Data-Driven Use Cases

Brick

https://brickschema.org

Brick Classes: “Points”, “Equipment”, and “Location”

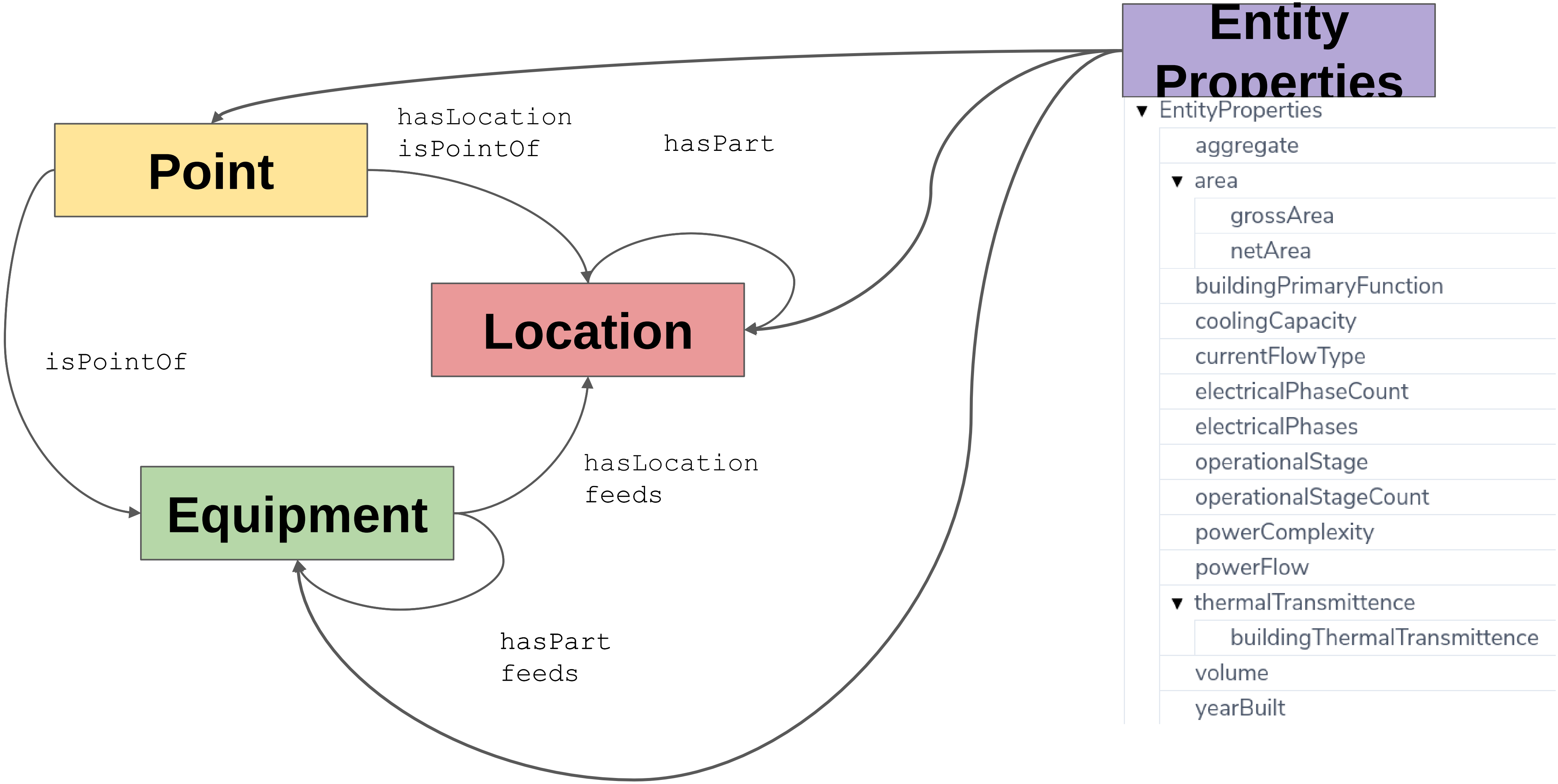


- Point
  - Sensor
    - Temperature Sensor
      - Air Temperature Sensor
        - Supply Air Temperature Sensor
        - Return Air Temperature Sensor
        - . . .
    - Humidity Sensor
    - . . .
  - Setpoint
    - Damper Position Setpoint
    - Temperature Setpoint
    - . . .
  - Command
    - . . .

- Equipment
  - HVAC
    - Air Handling Unit
      - . . .
    - Terminal Unit
      - Variable Air Volume Box
      - Constant Volume Box
      - Fan Coil Unit
    - . . .
  - Lighting
    - Switches
      - . . .
    - Luminaires and Lamps
      - . . .

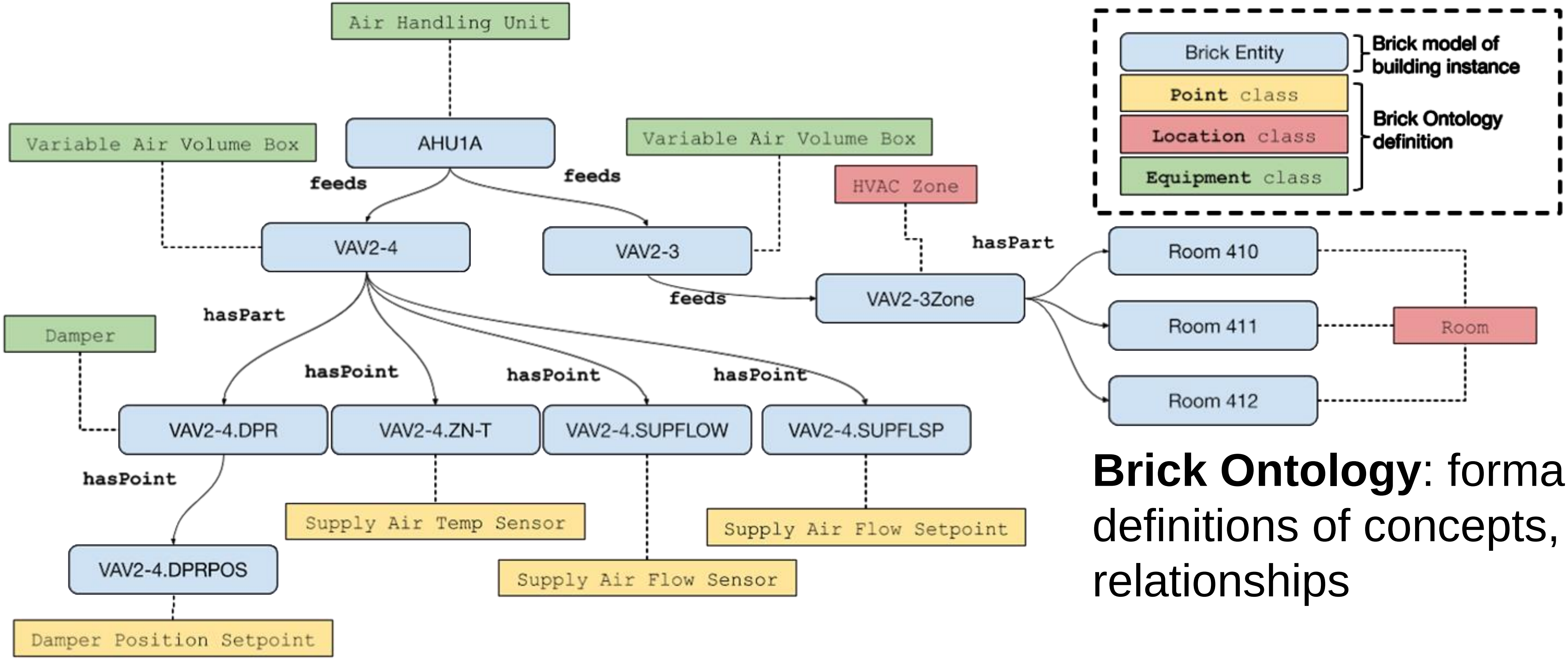
- Location
  - Room
    - Conference Room
    - Classroom
    - Lecture Hall
    - Kitchen
    - Reception
    - . . .
  - Building
  - Floor
  - Zone
    - HVAC Zone
    - Lighting Zone
    - Fire Zone
    - . . .

Brick Relationships



- ▼ EntityProperties
  - aggregate
  - ▼ area
    - grossArea
    - netArea
  - buildingPrimaryFunction
  - coolingCapacity
  - currentFlowType
  - electricalPhaseCount
  - electricalPhases
  - operationalStage
  - operationalStageCount
  - powerComplexity
  - powerFlow
  - ▼ thermalTransmittence
    - buildingThermalTransmittence
  - volume
  - yearBuilt

https://brickschema.org



Brick Ontology: formal definitions of concepts, relationships

Brick Model: the graph representing a particular building

Brick Adoption

Brick Data Platforms,
s

Johnson Controls

IBM®

CSIRO

Onboard

MAPPED

Bractlet

CLOCKWORKS ANALYTICS

Brick-enabled
R a

NREL NATIONAL RENEWABLE ENERGY LABORATORY

Carnegie Mellon University

BERKELEY LAB Lawrence Berkeley National Laboratory

Canada NRC·CNRC

Berkeley UNIVERSITY OF CALIFORNIA

SDU University of Southern Denmark

UC San Diego

Pacific Northwest NATIONAL LABORATORY

BCA

機電工程署 EMSD

Impacting Metadata
Standards and Effort

ASHRAE®

Project Haystack

Google

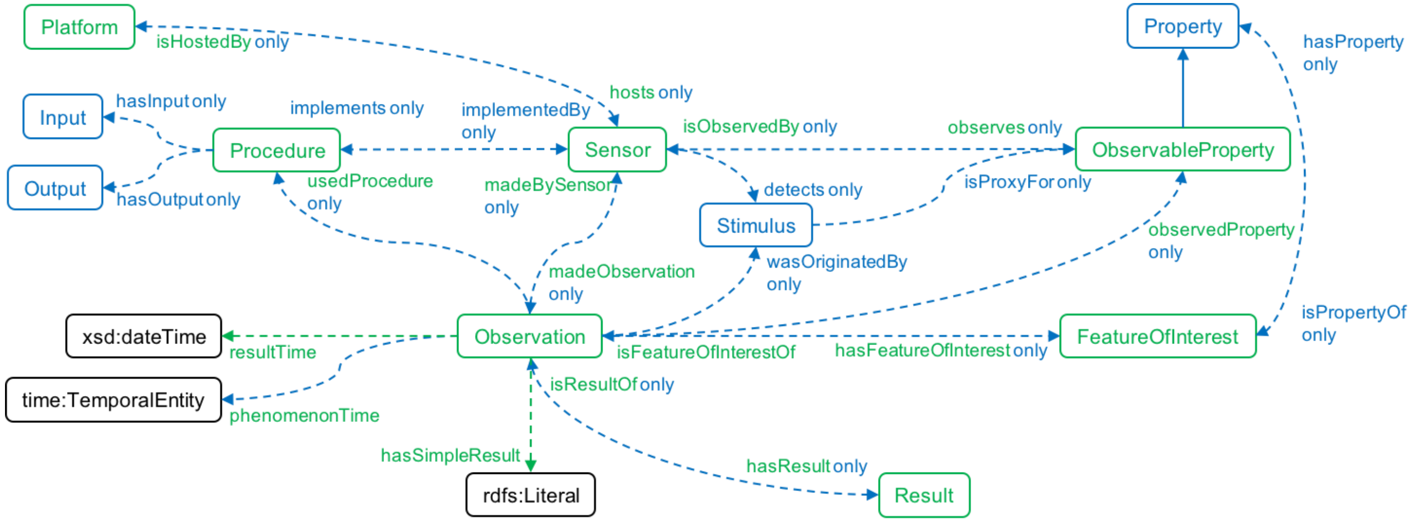
W3C®

Microsoft

Interesting Elements of Brick’s Design

- **Get your telemetry out of my RDF model!**

- Existing telemetry databases/systems already have semantics:
  - Or have semantics placed onto them
  - Rarely a need to make this explicit in the model



Awkward, complex to query

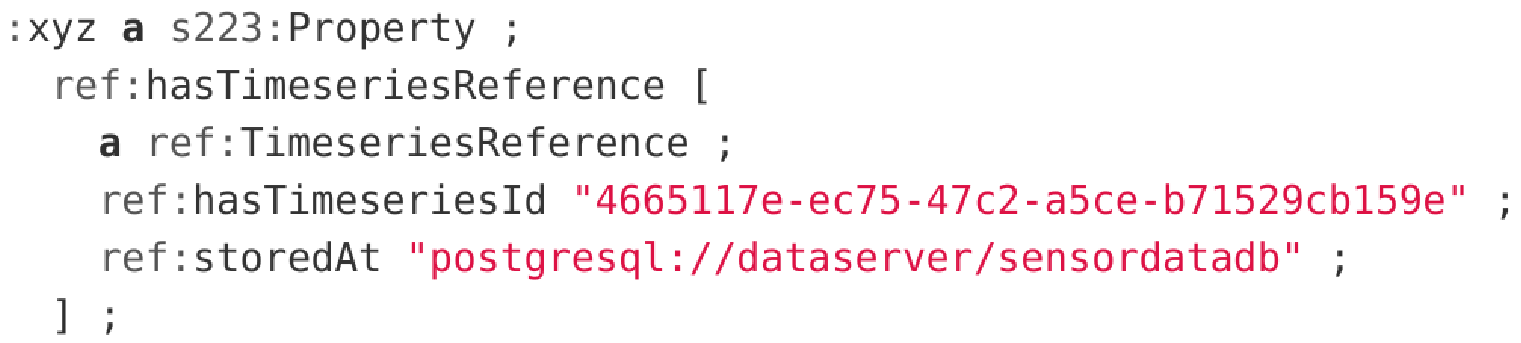
https://brickschema.org

Interesting Elements of Brick’s Design

- **Get your telemetry out of my RDF model!**
- Existing telemetry databases/systems already have semantics:
  - Or have semantics placed onto them
  - Rarely a need to make this explicit in the model
- Brick’s approach:
  - Put the “foreign key” or the access parameters into the model
  - Software has to go ”get” the data using its own client
  - Avoid rolling our own IDL – this is a recipe for disaster!
    - Remember SOAP? WSDL? COM? DCOM? CORBA?
  - Use existing standards + “paths” if needed
    - Xpath, jsonpath, etc….

External Reference Types

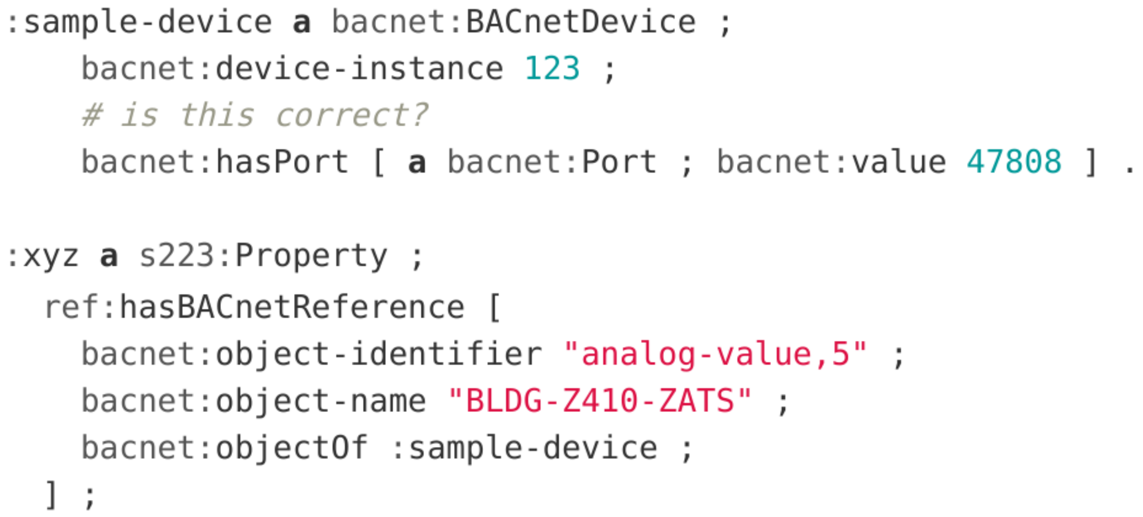
- **hasTimeseriesReference:**
  - **storedAt:** database connection string
  - **timeseriesId:** primary key in timeseries table
  - **dataTable:** name of (SQL) table containing data
  - **dataColumn, timeColumn, valueColumn:** names of (SQL) fields



```
:xyz a s223:Property ;
  ref:hasTimeseriesReference [
    a ref:TimeseriesReference ;
    ref:hasTimeseriesId "4665117e-ec75-47c2-a5ce-b71529cb159e" ;
    ref:storedAt "postgresql://dataserver/sensordatadb" ;
  ] ;
```

External Reference Types

- **hasBACnetReference:**
  - **object-identifier, object-name**
  - **objectOf:** points to the BACnet device
  - **read-property:** which property of the BACnet object should be read (defaults to present value)
  - or, **BACnetURI**: URI containing most/all of the above information

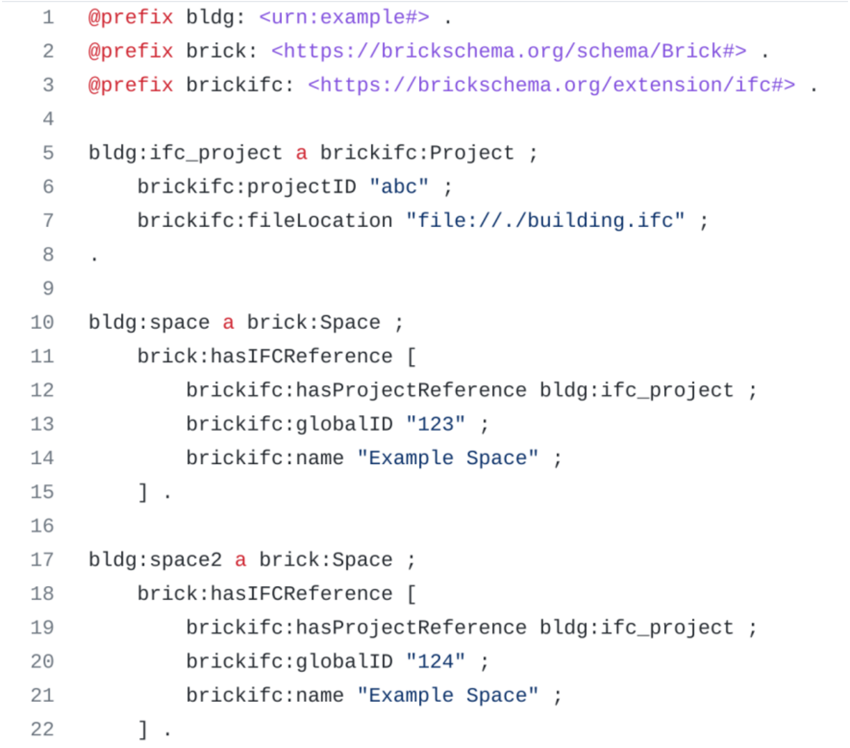


```
:sample-device a bacnet:BACnetDevice ;
    bacnet:device-instance 123 ;
    # is this correct?
    bacnet:hasPort [ a bacnet:Port ; bacnet:value 47808 ] .

:xyz a s223:Property ;
  ref:hasBACnetReference [
    bacnet:object-identifier "analog-value,5" ;
    bacnet:object-name "BLDG-Z410-ZATS" ;
    bacnet:objectOf :sample-device ;
  ] ;
```

External Reference Types

- **hasIFCReference:**
  - **hasProjectReference:** points to description of IFC project (e.g. file location)
  - **globalID:** entity identifier in the IFC project
  - **name:** label in the IFC project

- Not "data" exactly, but still useful



```
@prefix bldg: <urn:example#> .
@prefix brick: <https://brickschema.org/schema/Brick#> .
@prefix brickifc: <https://brickschema.org/extension/ifc#> .

bldg:ifc_project a brickifc:Project ;
    brickifc:projectID "abc" ;
    brickifc:fileLocation "file://./building.ifc" ;
.

bldg:space a brick:Space ;
    brick:hasIFCReference [
        brickifc:hasProjectReference bldg:ifc_project ;
        brickifc:globalID "123" ;
        brickifc:name "Example Space" ;
    ] .

bldg:space2 a brick:Space ;
    brick:hasIFCReference [
        brickifc:hasProjectReference bldg:ifc_project ;
        brickifc:globalID "124" ;
        brickifc:name "Example Space" ;
    ] .
```

Interesting Elements of Brick’s Design

- Classes vs Properties:
  - **`:x a brick:Air_Temperature_Sensor`**

    vs

    ```
    :x a brick:Sensor ;
       brick:hasSubstance brick:Air ;
       brick:hasQuantity  brick:Temperature ;
     .
    ```

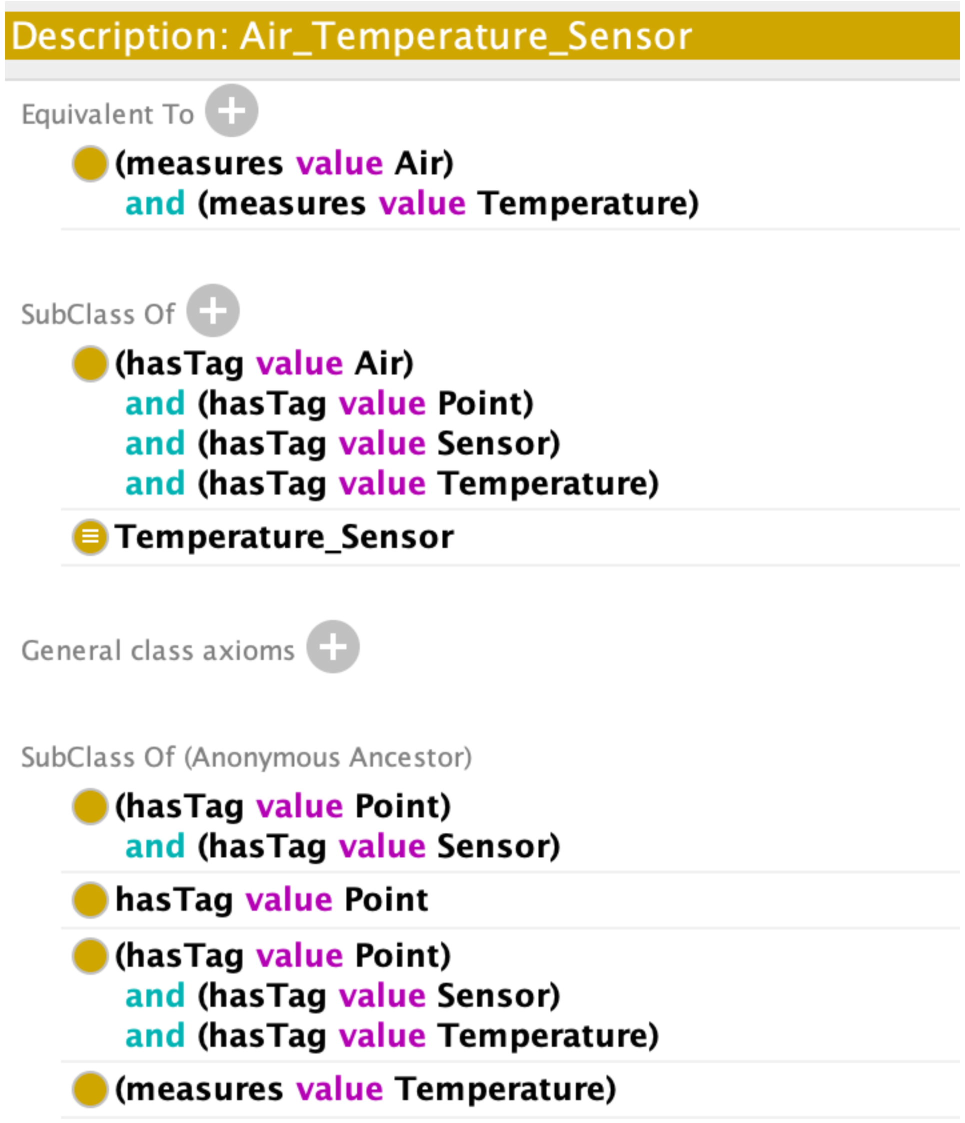
- **Classes** are more natural when *authoring* a model
  - Easy to grasp for non-ontologists
  - “Type explosion” --- is this a problem?
  - Implicit properties --- actually a problem
- **Properties** are more natural when *querying* a model

Interesting Elements of Brick’s Design

- Brick handles **both!**

- v1.2 (OWL 2 RL):
  - Bi-directional population: class ↔ properties
  - Infer tags as well!
  - Awkward (more on this later)
  - Slow! Needed a new reasoner:
    - https://github.com/gtfierro/reasonable

  - Reasoning polluted the graph with tons of blank nodes that confused users



Description: Air_Temperature_Sensor

Equivalent To

- (measures value Air)
  and (measures value Temperature)

SubClass Of

- (hasTag value Air)
  and (hasTag value Point)
  and (hasTag value Sensor)
  and (hasTag value Temperature)
- Temperature_Sensor

General class axioms

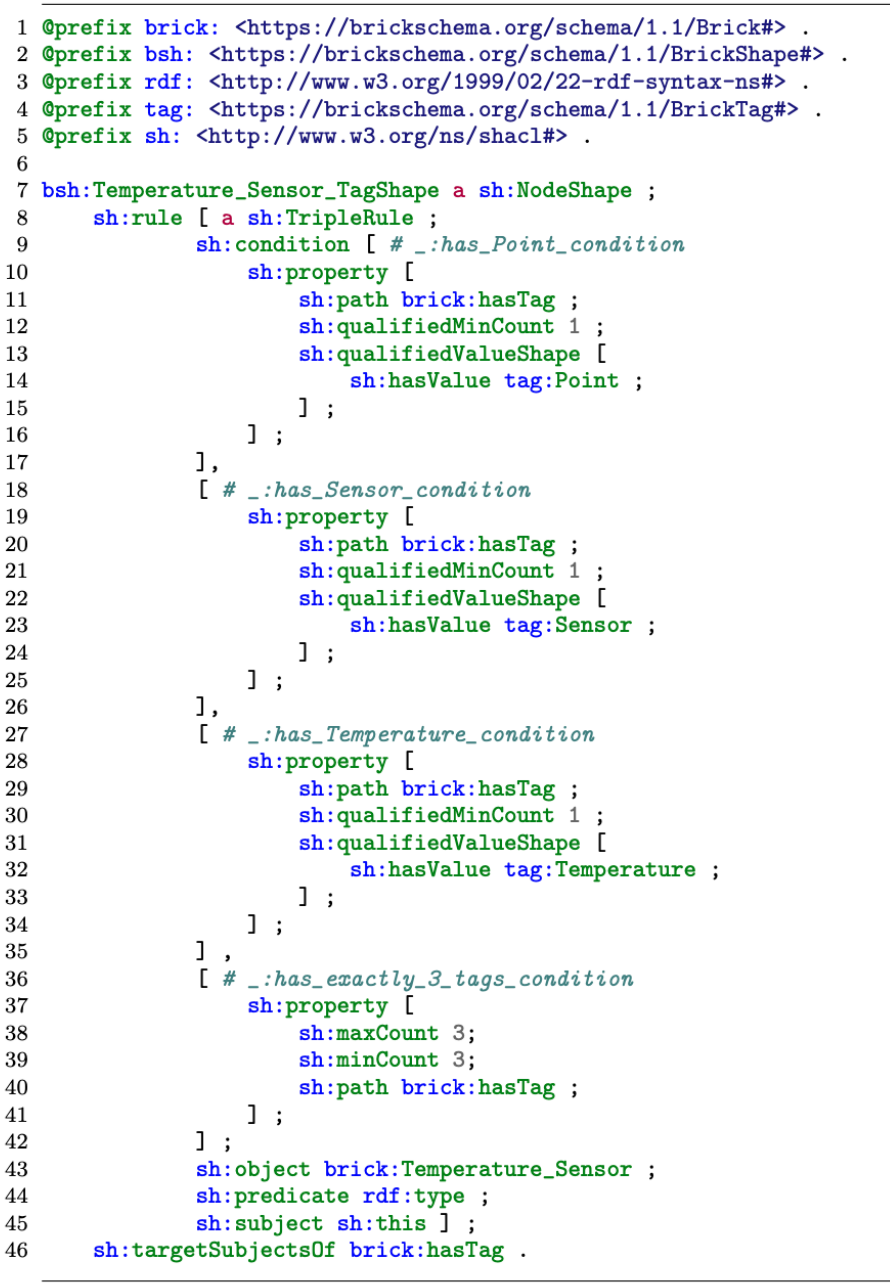
SubClass Of (Anonymous Ancestor)

- (hasTag value Point)
  and (hasTag value Sensor)
- hasTag value Point
- (hasTag value Point)
  and (hasTag value Sensor)
  and (hasTag value Temperature)
- (measures value Temperature)

Interesting Elements of Brick’s Design

- Brick handles **both!**

- v1.3 (SHACL + SHACL-AF):
  - Bi-directional population: class ↔ properties
  - Infer tags as well! tags ↔ class
  - Much easier to express and *validate* these behaviors



```
@prefix brick: <https://brickschema.org/schema/1.1/Brick#> .
@prefix bsh: <https://brickschema.org/schema/1.1/BrickShape#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix tag: <https://brickschema.org/schema/1.1/BrickTag#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .

bsh:Temperature_Sensor_TagShape a sh:NodeShape ;
    sh:rule [ a sh:TripleRule ;
            sh:condition [ # _:has_Point_condition
                sh:property [
                    sh:path brick:hasTag ;
                    sh:qualifiedMinCount 1 ;
                    sh:qualifiedValueShape [
                        sh:hasValue tag:Point ;
                    ] ;
                ] ;
            ],
            [ # _:has_Sensor_condition
                sh:property [
                    sh:path brick:hasTag ;
                    sh:qualifiedMinCount 1 ;
                    sh:qualifiedValueShape [
                        sh:hasValue tag:Sensor ;
                    ] ;
                ] ;
            ],
            [ # _:has_Temperature_condition
                sh:property [
                    sh:path brick:hasTag ;
                    sh:qualifiedMinCount 1 ;
                    sh:qualifiedValueShape [
                        sh:hasValue tag:Temperature ;
                    ] ;
                ] ;
            ] ,
            [ # _:has_exactly_3_tags_condition
                sh:property [
                    sh:maxCount 3;
                    sh:minCount 3;
                    sh:path brick:hasTag ;
                ] ;
            ] ;
            sh:object brick:Temperature_Sensor ;
            sh:predicate rdf:type ;
            sh:subject sh:this ] ;
    sh:targetSubjectsOf brick:hasTag .
```

Brick
https://brickschema.org

SHACL(-AF) vs OWL (2 RL)

- Brick (and 223P) are moving towards SHACL, SHACL-AF and ditching OWL
- Why?
    1. the *open-world assumption* is inappropriate for CPS settings
    2. emerging use cases more focused on validation than DL inferencing
    3. validation in OWL is limited, difficult to use

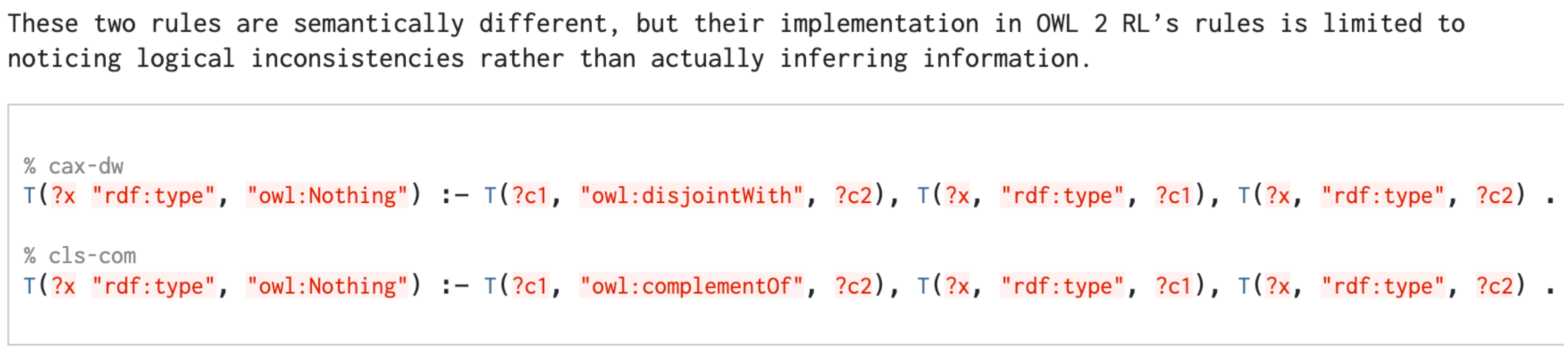
CWA > OWA

- OWA makes sense on “the web”:
  - Don’t know who is making statements about you

- In CPS deployments, there *is* a bound on what is physically/logically present
  - Digital records intended to be comprehensive, authoritative references
  - Want to reason about what is *not* present, as well as what *is*

OWL Issue: Limited Negation

- Example: **Mutually exclusive information:**
  - E.g., entities may not be an instance of both an Equipment and a Location class
  - E.g., entities can measure Air and CO2, but not Air and Water simultaneously



These two rules are semantically different, but their implementation in OWL 2 RL's rules is limited to noticing logical inconsistencies rather than actually inferring information.

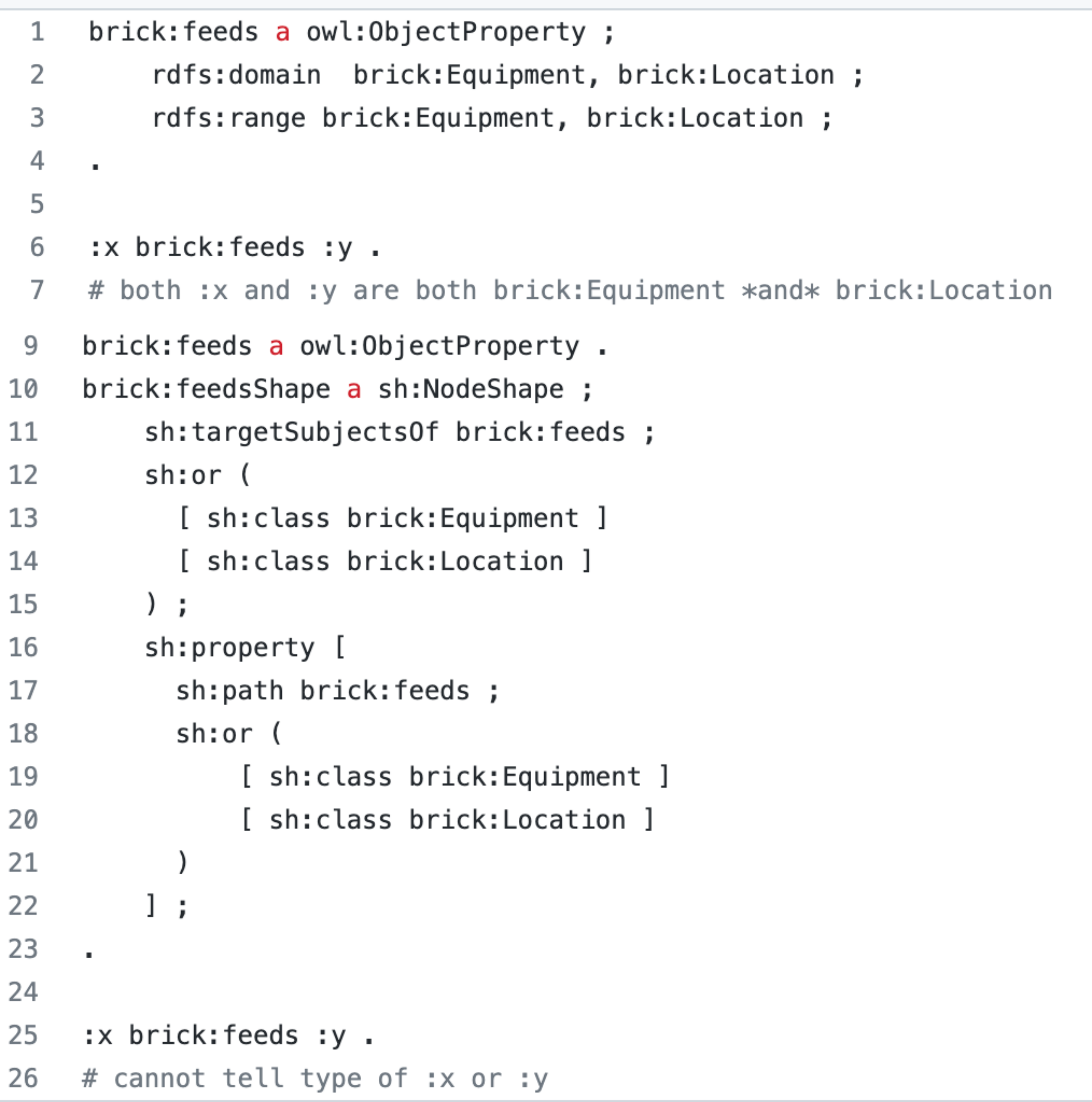
```
% cax-dw
T(?x "rdf:type", "owl:Nothing") :- T(?c1, "owl:disjointWith", ?c2), T(?x, "rdf:type", ?c1), T(?x, "rdf:type", ?c2) .

% cls-com
T(?x "rdf:type", "owl:Nothing") :- T(?c1, "owl:complementOf", ?c2), T(?x, "rdf:type", ?c1), T(?x, "rdf:type", ?c2) .
```

OWL Issue: **And**, not **Or**

- Multiple domains/ranges for same property can aid in use:
    - Connect **Equipment** to **Spaces**, **Equipment** to **Equipment**, **Spaces** to **Equipment**

- OWL 2 RL semantics for `rdfs:domain/rdfs:range` give us *intersection*
    - We actually want *union*

- Could define multiple relationships, but this is more a hack than anything



OWL

```
brick:feeds a owl:ObjectProperty ;
  rdfs:domain  brick:Equipment, brick:Location ;
  rdfs:range brick:Equipment, brick:Location ;
.

:x brick:feeds :y .
# both :x and :y are both brick:Equipment *and* brick:Location
```

SHACL

```
brick:feeds a owl:ObjectProperty .
brick:feedsShape a sh:NodeShape ;
  sh:targetSubjectsOf brick:feeds ;
  sh:or (
   [ sh:class brick:Equipment ]
   [ sh:class brick:Location ]
  ) ;
  sh:property [
   sh:path brick:feeds ;
   sh:or (
     [ sh:class brick:Equipment ]
     [ sh:class brick:Location ]
   )
  ] ;
.

:x brick:feeds :y .
# cannot tell type of :x or :y
```

Brick

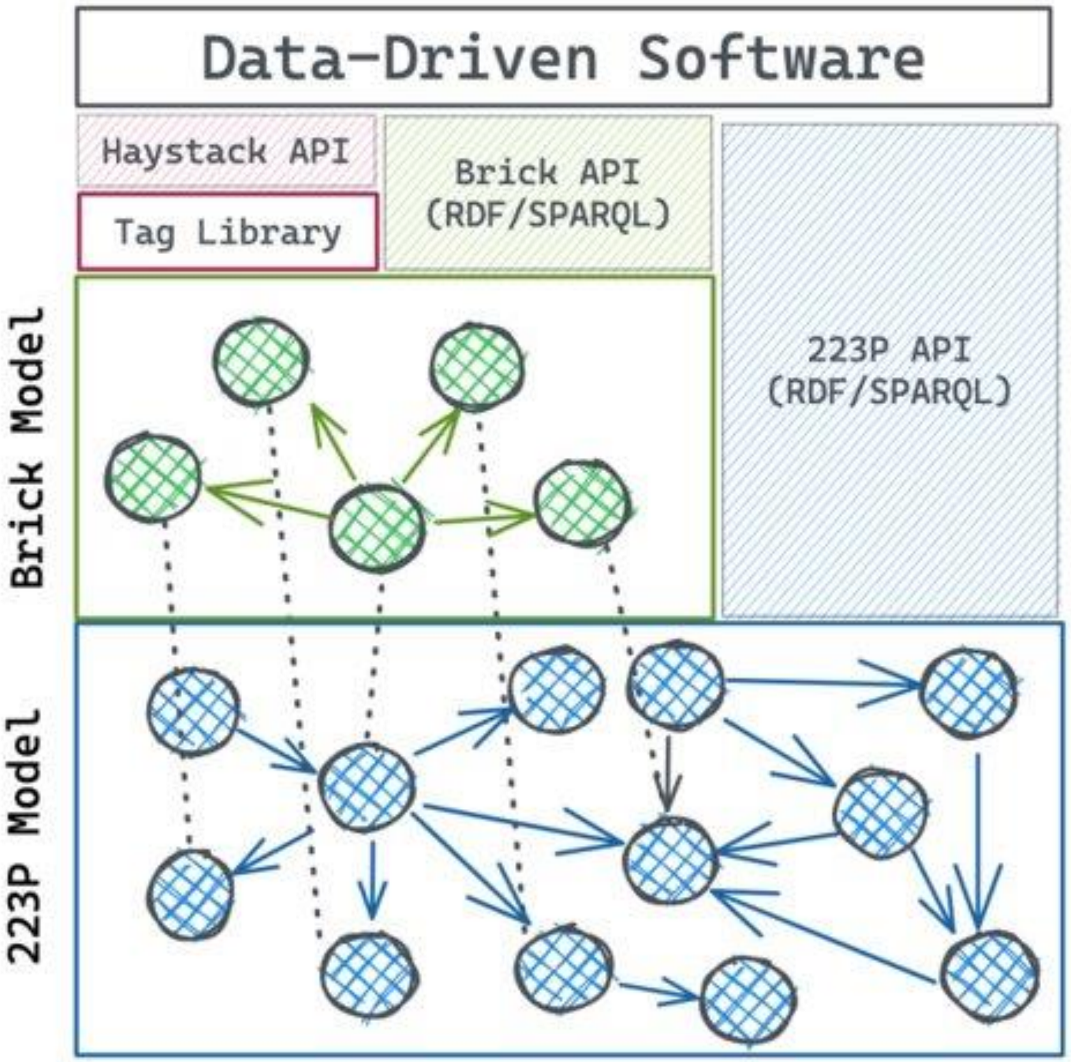
https://brickschema.org

Other OWL Issues

- Inference to materialize `rdfs:subClassOf` transitive closure:
    - Hard to know what the “most specific” types of any entity are!
- OWL 2 RL rules spit out `owl:Nothing` when logical inconsistency is reached
    - No hint as to why or which rule…
- Cannot say “when X is *not true*”:
    - `air temp setpoint:` a setpoint
    - **`max`** `air temp setpoint:` a “parameter”; disjoint from setpoints

Brick, Haystack, 223P and others

- Goal: interoperability between **Brick**, **Project Haystack** and **223(P)**
  - Technical solution in development
  - Ongoing, active development
- Approach:
  - Different levels of abstraction
  - 223P: fine-grained, detailed
  - Brick: high-level, application-facing, verifiable, familiar naming
  - Haystack: high-level, application-facing
- Keep an eye out for future announcements



Brick
https://brickschema.org

Brick Community: How to Get Involved

- Read documentation, resources, downloads, reference models
  - https://brickschema.org/
  - https://docs.brickschema.org

- Subscribe to community forum and mailing list
  - https://groups.google.com/forum/#!forum/brickschema

- Download and contribute to open-source ontology and tool development
  - https://github.com/BrickSchema/
  - CONTRIBUTING.md in https://github.com/BrickSchema/Brick/

- Join the working groups (calendar available)
  - https://brickschema.org/blog/working-groups/

https://home.gtf.fyi
gtfierro@mines.edu