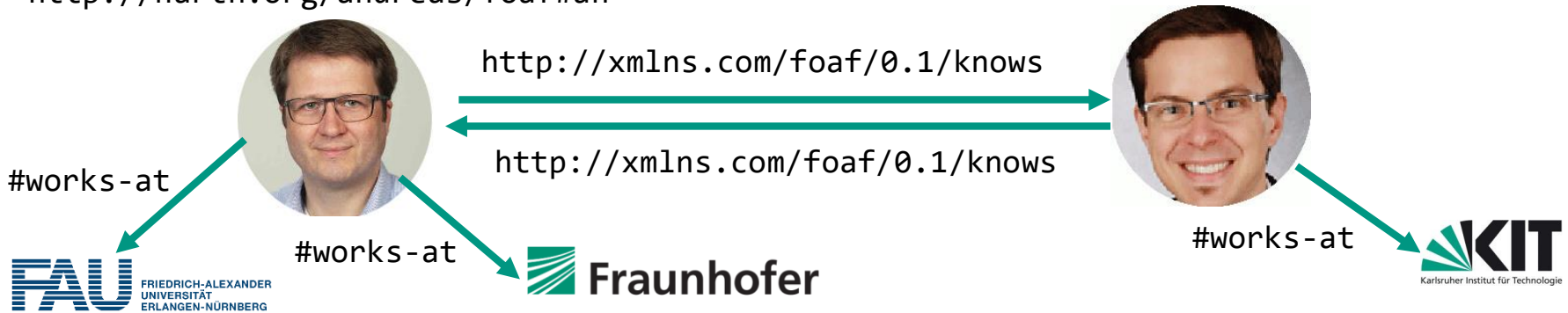
Tutorial on Distributed Knowledge Graphs for the Web of Things, Part IV: Agents and the Cognitive Loop

Tobias Käfer (KIT) and Andreas Harth (FAU)

Tutorial @ 10th International Conference on the Internet of Things (IoT), 2020



Agenda

- **Web Architecture and Linked Data**
- User Agents and Cognitive Architectures
- Query Processing User Agents
- Link Traversal Query Processing User Agents
- Summary

Execution Environment

- How to represent queries?
- How to represent behaviour?
- Where should queries (and behaviour) be executed?
- Not in the cloud!
- Need a (small and elegant) execution environment...

William of Ockham, 1287-1347

The simplest system architecture ~~explanation~~ is usually the correct one.

Keep It Simple

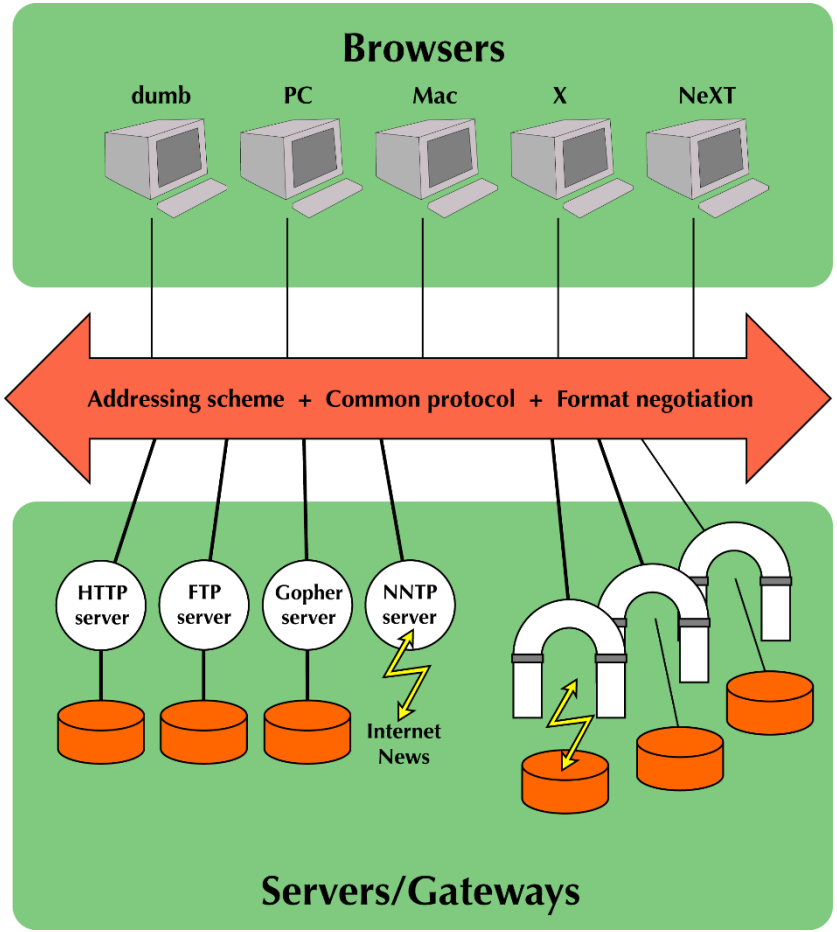
- The web is a very simple (but scalable) hypertext system
  - Tim Berners-Lee's paper to a hypertext conference was only accepted as a poster
- RDF is a very simple knowledge representation language
- RDFS provides only very few modelling primitives
- Schema.org provides a fixed vocabulary

- Operational agent-oriented programming…
  - Yoav Shoham, Agent-Oriented Programming. Artificial Intelligence, 1993
- …instead of Situation Calculus
  - J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In: Machine Intelligence, 4:463–502. Edinburgh University Press, 1969

- If you want, layer more complex things on top later

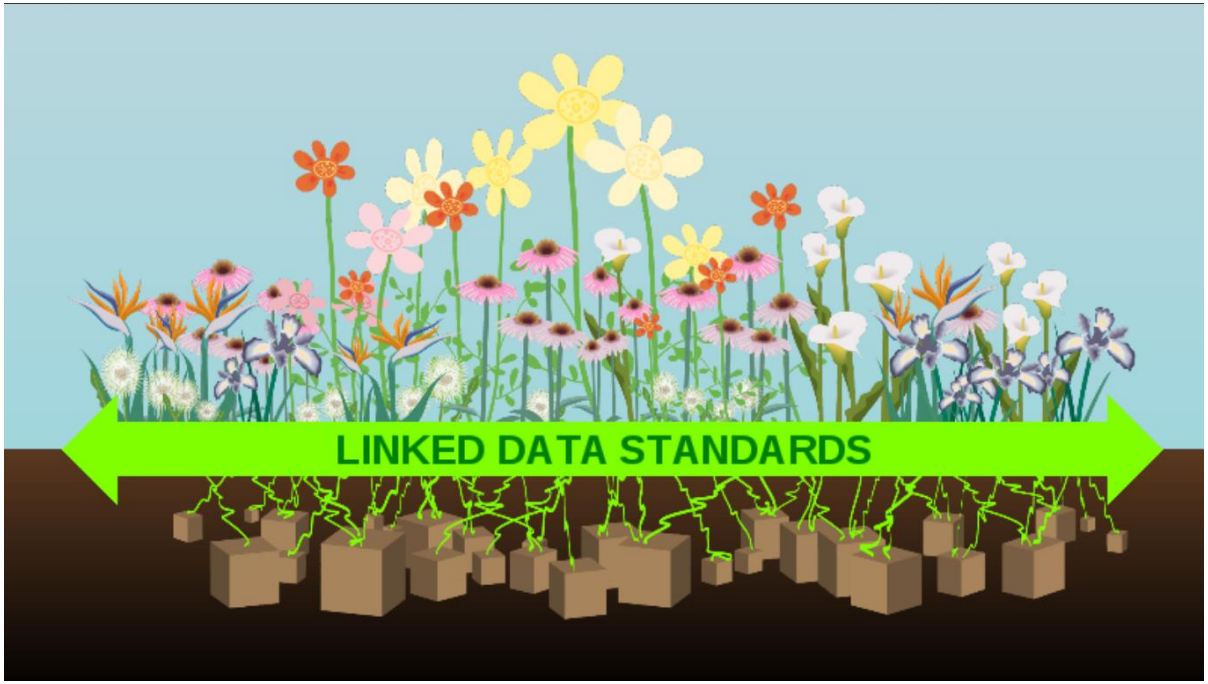
Web Architecture

- URI: RFC 1630 (1994), now RFC 3986
- HTTP: RFC 1945 (1996), now RFC 7230, 7231, 7232, 7234, 7235



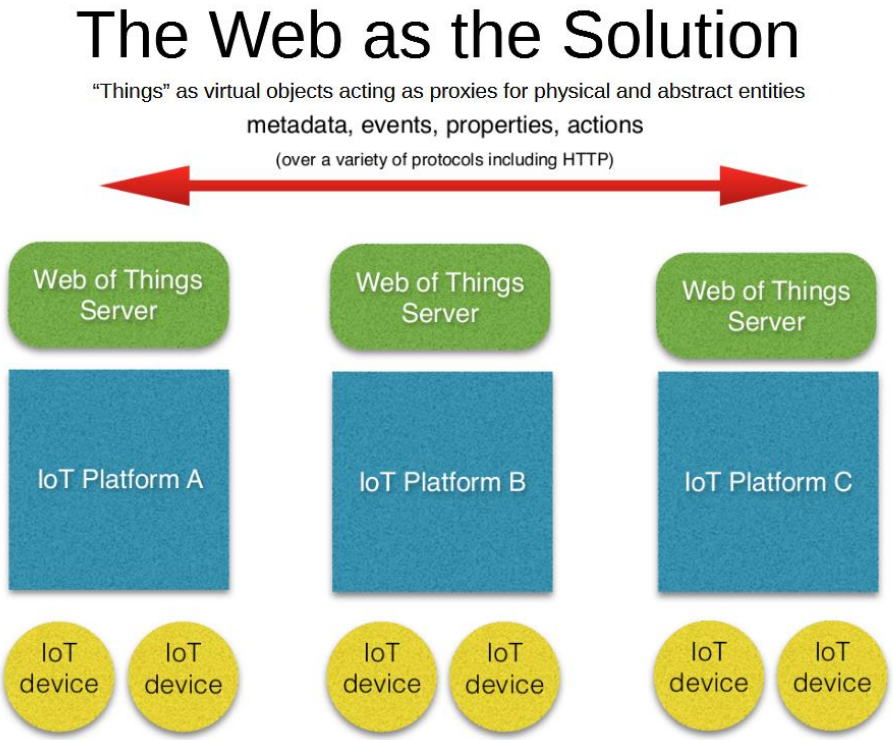
https://www.w3.org/DesignIssues/diagrams/history/Architecture_crop.png
Redrawn from an image from 1990

Linked Data Architecture (2009)



https://www.w3.org/2009/Talks/0204-ted-tbl/#(7)

Web of Things Architecture (2015)



https://www.w3.org/2015/05/wot-framework.pdf

Adapted from https://www.w3.org/DesignIssues/LinkedData.html

Linked Data Principles: Two Perspectives

Data Consumer (User Agent)

1. Assume URIs as names for things. ✓
2. User agents look up HTTP URIs. ✓
3. User agents process RDF/RDFS documents containing useful information and provide the ability to evaluate SPARQL queries. ✘
4. User agents can discover more things via accessing links to other URIs. ✘

Data Publisher (Server)

1. Coin URIs to name things. ✓
2. Use a HTTP server to provide access to documents. ✓
3. Upon receiving a request for a URI, the server returns useful information (about the URI in the request) in RDF and RDF Schema. ✓
4. The “useful information” the server returns in the RDF document includes links to other URIs (on other servers). ✓

Agenda

- Web Architecture and Linked Data
- **User Agents and Cognitive Architectures**
- Query Processing User Agents
- Link Traversal Query Processing User Agents
- Summary

Examples of Linked Data User Agents

- Temperature recorder (for temperature of room, building, area…)
- Temperature display (for temperature of room, building, area…)
- Thermostat (for room temperature)
  - Setting temperature depending on who is in the room
- Heat alarm (for temperature of room, building, area…)
- Distance alarm (for theft detection)
- … (your ideas?)

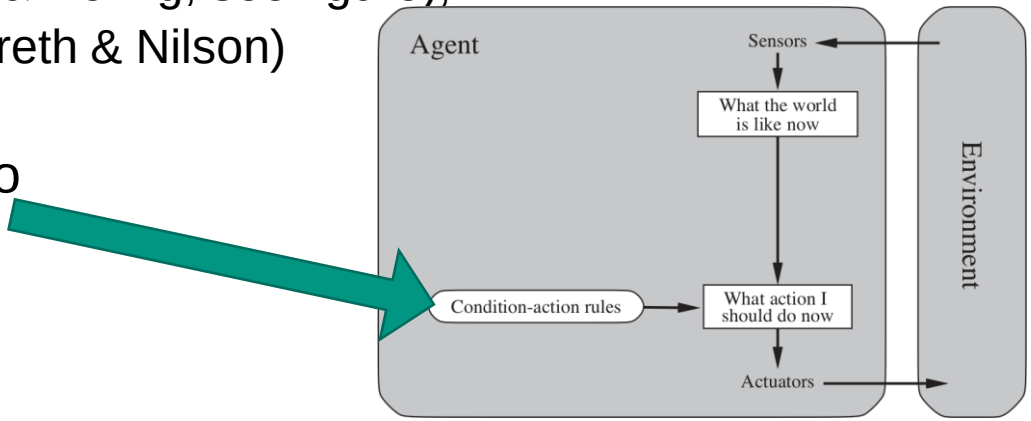
- WoT „Recipes“ or IFTTT „Applets“ provide similar functionality

Cognitive Architectures

- SOAR (initially: State, Operator, Apply, Result),
- ACT-R (Adaptive Control of Though – Rational)
- Goal: to create „intelligent agents“
- In the tutorial, we start with user agents that are
  - „simple reflex agents“ (Russel & Norvig, see figure),
  - aka „tropistic agents“ (Genesereth & Nilson)

- We explain how to use rules to control the agent‘s behaviour

Russel and Norvig, Artificial Intelligence – A Modern Approach, Third Edition, 2010



Towards Simple Agents On The Web

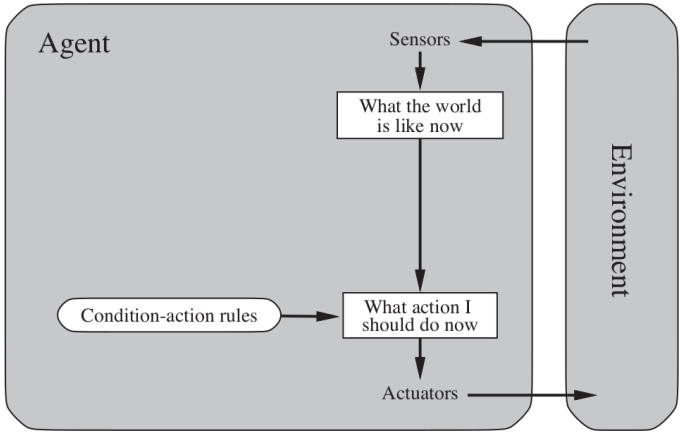
- Can we start with a simple “Hello World” scenario for agents on the web?

- Agents: Query processing on live data
- Server: Based on a (read-only) Linked Data interface to sensors

- Agents: Then, add condition-(read)action rules to specify link traversal

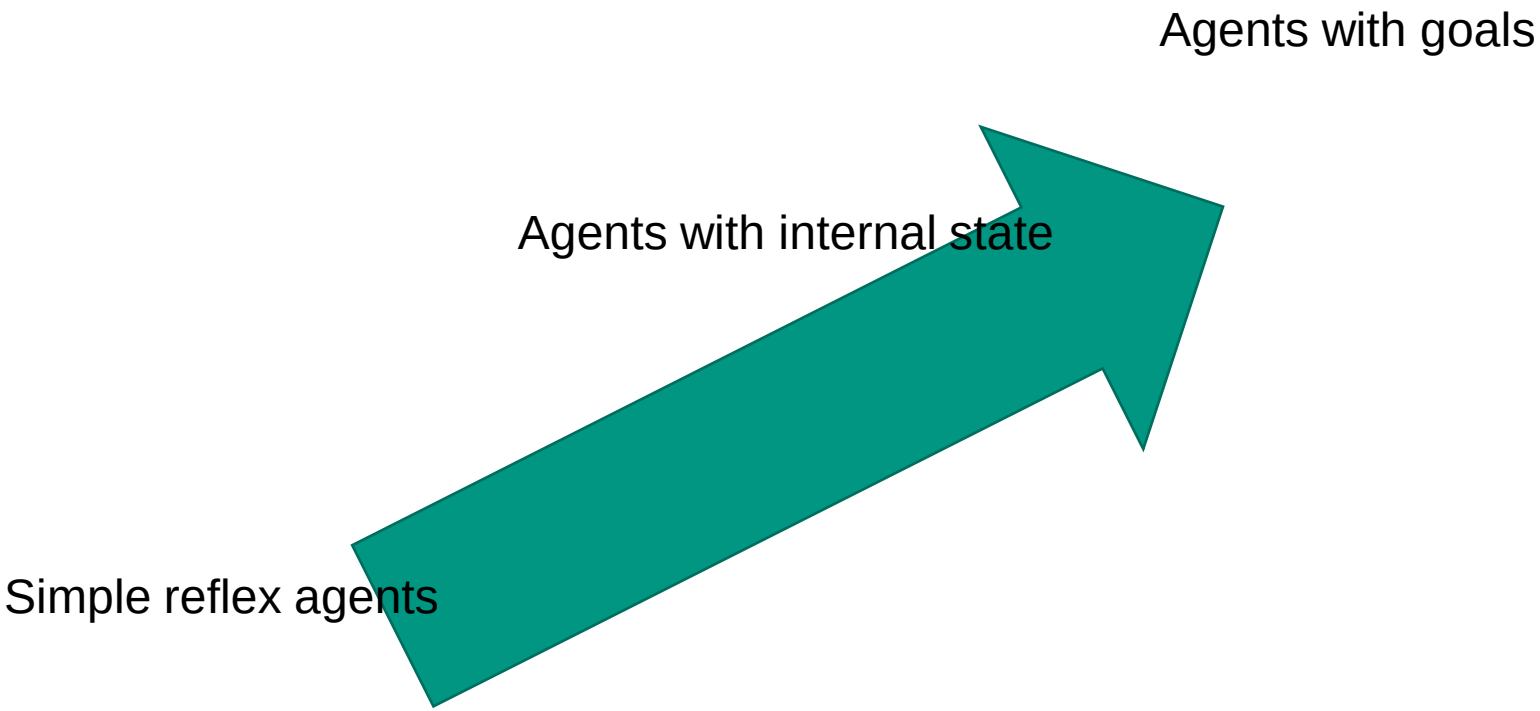
- Server: Next, provide a Read-Write interface to sensors and actuators
- Agents: And add condition-(read-write) action rules

Simple Reflex Agent



Russel and Norvig, Artificial Intelligence – A Modern Approach, Third Edition, 2010

Increasing Agent Complexity



Simple Reflex User Agents Layer Cake

User Agent

Read/Write Linked Data User Agents

Link-Following User Agents

Query Processing User Agents

Server/Environment

Adding Unsafe HTTP Methods Read-Write Linked Data

URI + HTTP + RDF (read-only) Linked Data

Agenda

- Web Architecture and Linked Data
- User Agents and Cognitive Architectures
- **Query Processing User Agents**
- Link Traversal Query Processing User Agents
- Summary

SPARQL FROM and FROM NAMED

Definition (Named Graph, RDF Dataset) *Let $\mathcal{G}$ be the set of RDF graphs and $\mathcal{U}$ be the set of URIs. A pair $\langle g, u \rangle \in \mathcal{G} \times \mathcal{U}$ is called a named graph. An RDF dataset consists of a (possibly empty) set of named graphs (with distinct names) and a default graph $g \in \mathcal{G}$ without a name.*

We assume the graph names are also addresses to locations with RDF documents.

Dave Beckett‘s roqet

- SPARQL processors operate on RDF datasets
- Many SPARQL processors operate on local RDF datasets
- A SPARQL „database“, to which you first import your data and then pose queries

- Instead, we can use a query processing user agent
- E.g., roqet (http://librdf.org/rasqal/roqet.html) (for years)
- roqet dereferences the URIs of the graphs in the RDF dataset during query time
- Possible to access current data and evaluate a query over the current data

Algorithm: Query Processing User Agent

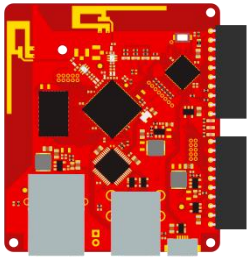
- Input: Query (as SPARQL algebra expression)
- Output: Query results

- Construct RDF dataset for local processing from URIs in Query
- Evaluate SPARQL algebra expression over local RDF dataset
- Return query results

Algorithm: Agent Loop

```
1  input: integer delay
2  output: number of iterations
3  t := 0
4  while termination condition/criteria not true:
5              access data, evalute queries...
6              t := t + 1
7              output results (datasets, query results, request/response information...)
8              wait delay milliseconds
9  return t
```

SPARQL Query



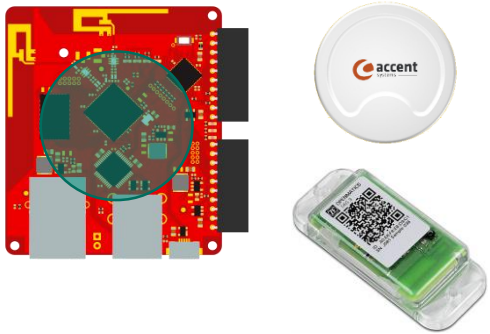
- Return true if beacon 0 is a certain distance away from the Tessel (approximating distance via RSSI value):

```
PREFIX ssn: <http://www.w3.org/ns/ssn/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

ASK
FROM <http://tessel2.lan/ble>
WHERE {
   <http://tessel2.lan/beacons/0#id> ssn:hasProperty ?prop .
   ?prop ssn:hasValue ?y .
   FILTER ((xsd:integer(?y)) > -60)
}
```

Agenda

- Web Architecture and Linked Data
- User Agents and Cognitive Architectures
- Query Processing User Agents
- **Link Traversal Query Processing User Agents**
- Summary

4: Links to Other URIs (Recap)



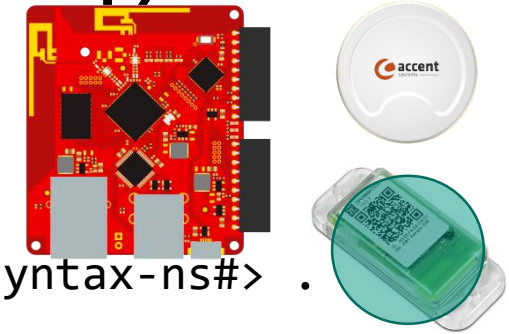
- A User Agent performing a HTTP GET on `http://tessel.lan/` leads to:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <http://example.org/woto#> .

<#id> rdf:type :Platform ; rdfs:comment "Andreas' Tessel2" ;
      :hosts <beacons/#id> , <leds/#id> .
```

- A HTTP GET on `http://tessel.lan/beacons/` leads to:

```
@prefix : <http://example.org/woto#> .

<#id> :hosts <0#id> , <1#id> , <2#id> , <3#id> .
```

3: Provide Useful Information in RDF (Recap)



- A User Agent performing a HTTP GET on `http://tessel.lan/beacons/3#id` leads to:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://example.org/woto#> .

</beacons/0#id> rdf:type :Sensor ;
                :hasProperty <#prop1> , <#prop2> .
<#prop1> rdf:type :ObservableProperty ;
       :measure :RSSI ; :value "-52" ; :unit "dbm" .
<#prop2> rdf:type :ObservableProperty ;
       :measure :temperature ; :value "23" ; :unit "Celcius" .
```

Link Traversal User Agent Loop

1. The user agent starts its interaction based on a specified seed URI.
2. The user agent performs HTTP requests[1] on URIs and parses the response message.
3. Based on the response, the user agent has the choice as to which URIs to dereference next.
4. The user agent decides which link(s) to follow and initiates a new request/new requests.

[1]We restrict HTTP requests to GET requests for now.

- Follow one link to arrive at a path through the web of information resources (e.g., depth-first search)
- Follow all links to arrive at a tree of information resources (e.g., breadth-first search)

How to Specify Which Links to Traverse?

Alternatives:

- Function that maps current state to additional requests
- Implicitely encoded in the algorithm/system
- Encode the function using rules

Request Rule Abstract Syntax

Definition (Request Template Pattern) *A request template pattern is a HTTP request, in which the start line S consists of a tuple* $\langle M, t, V \rangle$*, where* M *is the HTTP method,* $t \in \mathcal{U} \cup \mathcal{V}$ *is the request target, which can be a URI or variable, and* V *is the HTTP version.*

Definition (Request Rule) *Let* q *be a graph pattern and* r *a request template. A request rule is a pair* $\langle q, r \rangle$ *and has the form* `{ q } => { r } .` *All variables in* r *must also occur in* q *(the rule is called safe).*

Example Traversal Rules

```
@prefix http: <http://www.w3.org/2011/http#> .
@prefix httpm: <http://www.w3.org/2011/http-methods#> .
@prefix : <http://example.org/woto#> .

{ [] http:mthd httpm:GET ;                          # access index (entry) page
   http:requestURI <http://tessel.lan/> . }

{
  ?x :hosts ?y .                                    # match :hosts triples
} => {
  [] http:mthd httpm:GET ;                          # access linked documents
     http:requestURI ?y .
} .
```

Run with Linked Data-Fu, http://linked-data-fu.github.io/, Steffen Stadtmüller, Sebastian Speiser, Andreas Harth, Rudi Studer. "Data-Fu: A Language and an Interpreter for Interaction with Read/Write Linked Data". WWW 2013, Rio de Janerio, Brasil.

Link Traversal Query Processors

- Hartig et al. ISWC 2009
  - Index nested loops joins
  - Iterator model (bolted on top of ARQ/Jena)
- Harth et al. WWW 2010
  - Repeated evaluation
  - Improve source index over time
- Ladwig and Tran ISWC 2010
  - Architecture for link-traversal query processor (push, SHJ)
  - But no recursion
- Fionda, Gutierrez and Pirro WWW 2012 (also: NautiLOD)
  - No notion Information Resource Graph
  - Function calls on results (no REST state manipulation)
  - But: recursion on graph traversal paths
- Hartig and Perez 2016: LDQL
  - Language to specify link traversal
  - Keep link traversal specification and query specification separate (as in Harth et al. WWW 2010)

Link Traversal Query Processing User Agent

- Input: Query (as SPARQL algebra expression), Link traversal specification
- Output: Query results

- Construct RDF dataset for local processing from Link traversal specification
- Evaluate SPARQL algebra expression over local RDF dataset
- Return query results

- Loop: do repeated query evaluation

Agenda

- Web Architecture and Linked Data
- User Agents and Cognitive Architectures
- Query Processing User Agents
- Link Traversal Query Processing User Agents
- **Summary**

Summary

- The agent metaphor is attractive for deployment on the (Semantic) Web, also in scenarios around Internet of Things and Industry 4.0
- Agents can operate in a decentralised environment without the need for centralised components
- We have presented how single machine agents can operate in a Linked Data environment
- We use SPARQL for specifying queries, and a rule-based language for specifying link traversal

- With the foundations in place, we can move on to more sophisticated types of agents