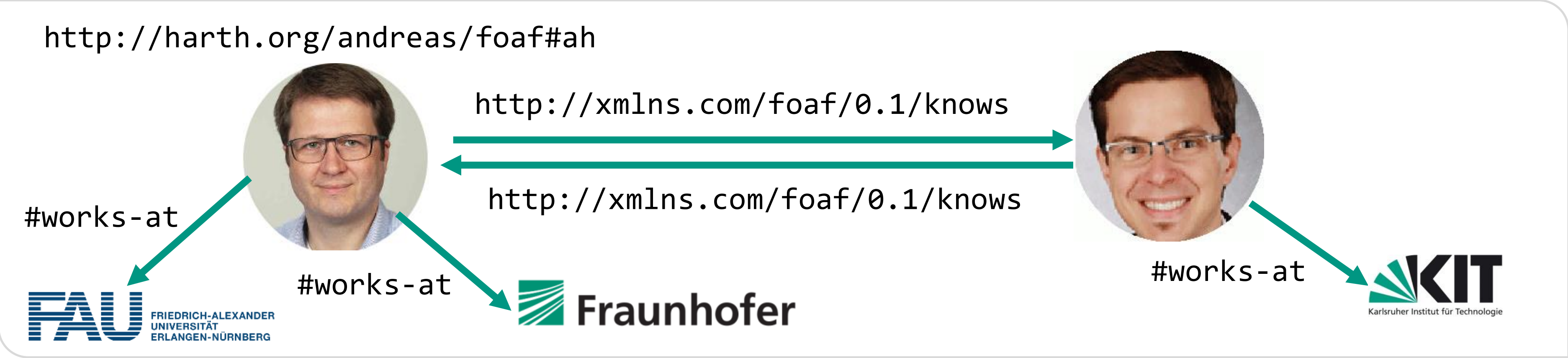
# Tutorial on Distributed Knowledge Graphs for the Web of Things, Part VII: Executing RW Linked Data Agents

**Tobias Käfer (KIT) and Andreas Harth (FAU)**

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



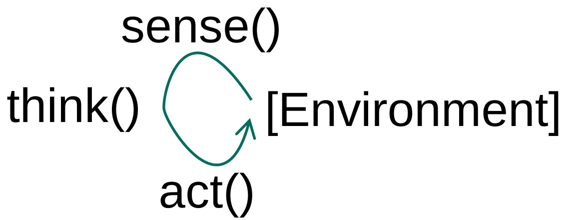

Karlsruhe Institute of Technology (KIT); Fraunhofer IIS-SCS; Friedrich-Alexander University Erlangen-Nuremberg (FAU)

**http://go.wiwi.kit.edu/togwot**

# How Far Away Are We From AI Agents on the Web of Things?

- Cognitive loop:
  - while true:
    - sense()
    - think()
    - act()
- Read-Write Linked Data gives us:
  - sense() and act() to interact with distributed sources
  - Knowledge Graphs to describe data

| Russell / Norvig‘s Agent Layer Cake [1] | Ingredients |
|---|---|
| Agents with goals | Capability descriptions |
| Agents with internal state | State Maintenance |
| Simple reflex agents | Execution semantics |
| (Describe Perception) | Data model |
| (Perception/action means) | Interaction |




[1] Russell and Norvig: “Artificial Intelligence – A Modern Approach”. Prentice Hall (1995).

# Technologies Covered So Far

- Read-Write Linked Data processing with IoT Devices
- Web of Things

| Level | Foundational approaches / categories | | | | | | |
|---|---|---|---|---|---|---|---|
| Capability description | Input, Output, Precondition, Effect (for automated composition) | Affordance (for manual composition) | | | | | … |
| Composition description | Rules | BPEL* | Pi cal-culus | Petri Nets | (Temporal) logic | Unformalised Implementation | … |
| Execution Semantics | ASM | LTS | Situation Calculus | Unformalised Implementation | | | … |
| Data model | Graph (RDF) | Nested (JSON, XML) | | | | | … |
| Interaction | REST | Arbitrary functions | Event/push | Blackboard | | | … |

*Semantics of BPEL have been given eg. in Petri Nets and ASMs, but Petri Nets are also used to describe compositions

# Technologies Covered So Far

- **Read-Write Linked Data** processing with IoT Devices
- Web of Things

| Level | Foundational approaches / categories | | | | | | |
|---|---|---|---|---|---|---|---|
| Capability description | Input, Output, Precondition, Effect (for automated composition) | | | Affordance (for manual composition) | | | … |
| Composition description | Rules | BPEL | Pi cal-culus | Petri Nets | (Temporal) logic | Unformalised Implementation | … |
| Execution Semantics | ASM | LTS | Situation Calculus | Unformalised Implementation | | | … |
| Data model | Graph (RDF) | | | Nested (JSON, XML) | | | … |
| Interaction | REST | Arbitrary functions | Event/push | Blackboard | | | … |

# Technologies Covered So Far

- Read-Write Linked Data processing with IoT Devices
- **Web of Things**

| **Level** | **Foundational approaches / categories** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Capability description | Input, Output, Precondition, Effect (for automated composition) | Affordance (for manual composition) | | | | | | … |
| Composition description | Rules | BPEL | Pi cal-culus | Petri Nets | (Temporal) logic | Unformalised Implementation | | … |
| Execution Semantics | ASM | LTS | Situation Calculus | Unformalised Implementation | | | | … |
| Data model | Graph (RDF) | Nested (JSON, XML) | | | | | | … |
| Interaction | REST | Arbitrary functions | Event/push | Blackboard | | | | … |

# Technologies Covered So Far

- **IoT Platforms**

| Level | Foundational approaches / categories | | | | | | |
|---|---|---|---|---|---|---|---|
| Capability description | Input, Output, Precondition, Effect (for automated composition) | Affordance (for manual composition) | … | | | | |
| Composition description | Rules | BPEL* | Pi cal-culus | Petri Nets | (Temporal) logic | Unformalised Implementation | … |
| Execution Semantics | ASM | LTS | Situation Calculus | Unformalised Implementation | … | | |
| Data model | Graph (RDF) | Nested (JSON, XML) | … | | | | |
| Interaction | REST | Arbitrary functions | Event/push | Blackboard | … | | |

# OWL-S and WSMO

- Semantic Web Service Description Language Stacks

| Level | Foundational approaches / categories | | | | | | |
|---|---|---|---|---|---|---|---|
| Capability description | Input, Output, Precondition, Effect (for automated composition) | | | Affordance (for manual composition) | | | … |
| Composition description | Rules | BPEL* | Pi cal-culus | Petri Nets | (Temporal) logic | Unformalised Implementation | … |
| Execution Semantics | ASM | LTS | Situation Calculus | Unformalised Implementation | | | … |
| Data model | Graph (RDF) | | | Nested (JSON, XML) | | | … |
| Interaction | REST | Arbitrary functions | Event/push | Blackboard | | | … |

*Semantics of BPEL have been given eg. in Petri Nets and ASMs, but Petri Nets are also used to describe compositions

# Hydra and Schema.org Potential Actions

- Affordance descriptions
- cf. Web of Things Actions/Properties/Events

| Level | Foundational approaches / categories | | | | | | |
|---|---|---|---|---|---|---|---|
| Capability description | Input, Output, Precondition, Effect (for automated composition) | Affordance (for manual composition) | | | | | … |
| Composition description | Rules | BPEL* | Pi cal-culus | Petri Nets | (Temporal) logic | Unformalised Implementation | … |
| Execution Semantics | ASM | LTS | Situation Calculus | Unformalised Implementation | | | … |
| Data model | Graph (RDF) | Nested (JSON, XML) | | | | | … |
| Interaction | REST | Arbitrary functions | Event/push | Blackboard | | | … |

# RESTdesc [1]

- Aim: Automated service composition and composition execution in the presence of hyperlinks in HTTP responses
- Composition problem:
  - Initial knowledge

    ```
    <#r> :isOn false .
    ```
  - API descriptions:

```
{ preconditions }=>{ HTTP-request.postconditions }.
```

    - Precondition, Postcondition: ~ BGP; Postcondition ~ HTTP response's body
    - HTTP-Request: (Method, URI + optional parameters)
      - Optional: eg. body: URIs or literals
  - Goal specification

    ```
    { <#r> :isOn true }=>{<#r> :isOn true } .
    ```
  - Background knowledge, eg. ontologies

- Sample API description:

```
@prefix : <http://example.org/>.
@prefix http: <http://www.w3.org/2011/http#>.

{
  <#r> :isOn false .
}
=>
{
  _:request http:methodName "PUT";
            http:requestURI /relay/1 ;
            http:body "<#r> :isOn true ."
            http:resp [ http:body ?b1 ].
  <#r> :isOn true .
}.
```

[1] Verborgh, Steiner, Van Deursen, Coppens, Vallés, Van de Walle: "Functional descriptions as the bridge between hypermedia APIs and the Semantic Web". In Proc. 3rd International Workshop on RESTful Design (WS-REST) (2012)

# RESTdesc Algorithm [1]

Proof before an HTTP request
Set of API descriptions in rules
Goal
World/Application state
Background knowledge

1) Start an N3 reasoner to generate a pre-proof for $(R, g, H, B)$.
   a) If the reasoner is not able to generate a proof, halt with failure.
   b) Else scan the pre-proof for applications of rules of $R$, set the number of these applications to $n_{pre}$
2) Check $n_{pre}$:
   a) If $n_{pre} = 0$, halt with success.
   b) Else continue with 3).
3) Out of the pre-proof, select a sufficiently specified HTTP request description which is part of the application of a rule $r \in R$.
4) Execute the described HTTP request and parse the (possibly empty) server response to a set of ground formulas $G$.
5) Invoke the reasoner with the new API composition problem $(R, g, H \cup G, B)$ to produce a post-proof.
6) Determine $n_{post}$:
   a) If the reasoner was not able to generate a proof, set $n_{post} \coloneqq n_{pre}$.
   b) Else scan the proof for the number of inference steps which are using rules from *R* and set this number of steps to $n_{post}$.
7) Compare $n_{post}$ with $n_{pre}$:
   a) If $n_{post} \geq n_{pre}$, go back to 1) with the new API composition problem $(R \setminus \{r\}, g, H, B)$.
   b) If $n_{post} < n_{pre}$, the post-proof can be used as the next pre-proof. Set $n_{pre} \coloneqq n_{post}$ and continue with 2)

[1] Verborgh, Arndt, Van Hoecke, De Roo, Mels, Steiner, Gabarró: "The pragmatic proof: Hypermedia API composition and execution". *Theory and Practice of Logic Programming*, *17*(1), (2017)

# Classifying RESTdesc

More than mere affordances, but not full IOPE

| Level | Foundational approaches / categories | | | | | |
|---|---|---|---|---|---|---|
| Capability description | Input, Output, Precondition, Effect (for automated composition) | Affordance (for manual composition) | … | | | |
| Composition description | Rules | BPEL* | Pi cal-culus | Petri Nets | (Temporal) logic | Unformalised Implementation | … |
| Execution Semantics | ASM | LTS | Situation Calculus | Unformalised Implementation | … | | |
| Data model | Graph (RDF) | Nested (JSON, XML) | … | | | |
| Interaction | REST | Arbitrary functions | Event/push | Blackboard | … | | |

*Semantics of BPEL have been given eg. in Petri Nets and ASMs, but Petri Nets are also used to describe compositions

# ASM4LD [0]

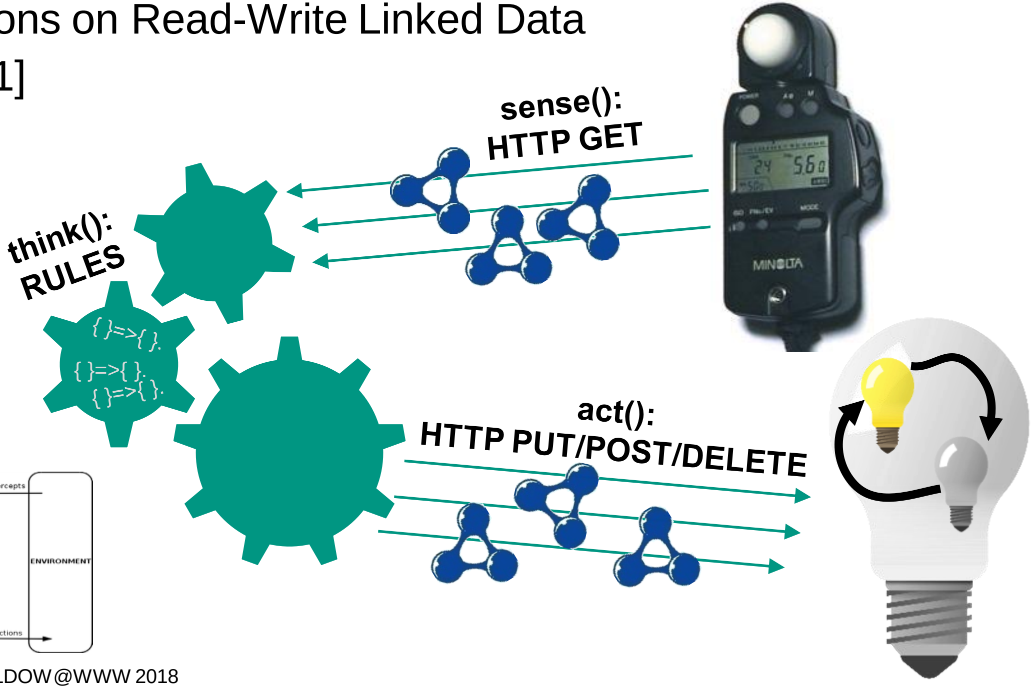
- A Model of Computation for Read-Write Linked Data [0]
- Operational Semantics for the Linked Data-Fu Language [1]

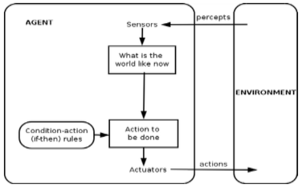
| Level | Foundational approaches / categories | | | | | | |
|---|---|---|---|---|---|---|---|
| Capability description | Input, Output, Precondition, Effect (for automated composition) | Affordance (for manual composition) | … | | | | |
| Composition description | Rules | BPEL* | Pi cal-culus | Petri Nets | (Temporal) logic | Unformalised Implementation | … |
| Execution Semantics | ASM | LTS | Situation Calculus | Unformalised Implementation | … | | |
| Data model | Graph (RDF) | Nested (JSON, XML) | … | | | | |
| Interaction | REST | Arbitrary functions | Event/push | Blackboard | … | | |

[0] Käfer & Harth: Rule-based Programming of User Agents for Linked Data. LDOW@WWW, 2018
[1] Stadtmüller, Speiser, Harth, Studer: Data-Fu: a language and an interpreter for interaction with read/write linked data. WWW 2013

# ASM4LD [0]

- Aim: Execution of agent specifications on Read-Write Linked Data
- Inspired by Simple Reflex Agents [1]
- Based on:
  - Abstract State Machines [2]
  - Model-theoretics semantics of RDF
  - Message semantics of HTTP
- Cognitive loop
  + Fixpoint loop for reasoning
  + Fixpoint loop for link following



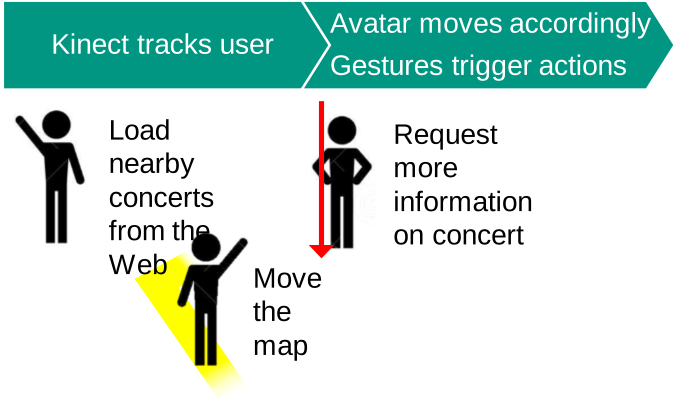


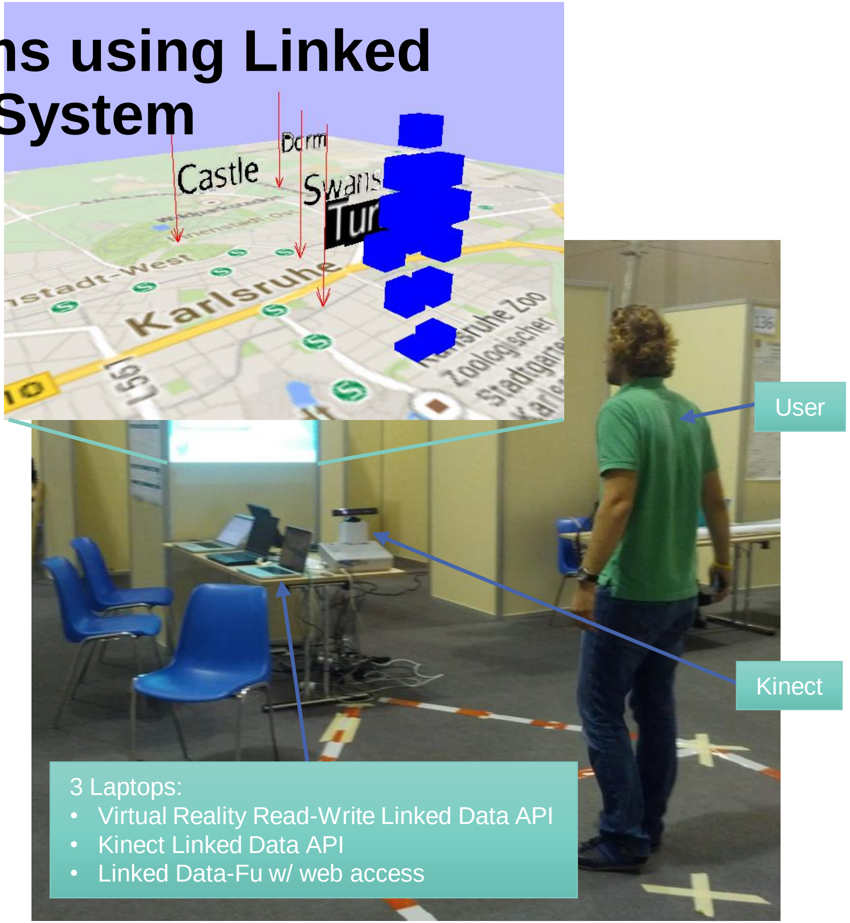
[0] Käfer & Harth: Rule-based Programming of User Agents for Linked Data. LDOW@WWW 2018
[1] Russell & Norvig: Artificial Intelligence – A Modern Approach. Prentice Hall (1996)
[2] Gurevich:. "Evolving algebras 1993: Lipari guide." *Specification and validation methods* (1995)

```
Require: assertions ▷ Graph
Require: rules ▷ Derivation and request rules
var data, oldData: set<triple>
var fixpointReached: Boolean
var unsafeRequests: set<request>
while true do ▷ Loop of the ASM steps
    unsafeRequests.clear()
    data.clear()
    data.add(assertions)
    repeat ▷ Loop for determining the fixpoint and the update set
        fixpointReached <- true
        for rule : rules do
            if rule.matches(data) then
                oldData = data.copy()
                if rule.type==derivation then
                    data.add(rule.match(data).data)
                else ▷ So the rule must be an interaction rule
                    if rule.match(data).request.type==GET then
                        data.add(rule.match(data).request.execute())
                    else
                        unsafeRequests.add(rule.match(data).request)
                    end if
                end if
                if ! data.copy().remove(oldData).isEmpty() then
                    fixpointReached <- false
                end if
            end if
        end for
    until fixpointReached
    for request : unsafeRequests do ▷ Enacting the update set
        request.execute()
    end for
end while
```

# Algorithm to combine fixpoint calulation for forward-chaining reasoning, and link following, with the cognitive loop

# Integration of Distributed Systems using Linked Data: Example: a Virtual Reality System




- We encoded in Linked Data-Fu rules:
  - Movement of the avatar according to Kinect data
  - Detection of user gestures
  - Movement of the map according to gestures
  - Loading of concert data from the web
  - Data integration between VR RWLD API, concert LD API, Kinect LD API
- Execution at Kinect sensor refresh rate (30Hz)

Keppmann, Käfer, Stadtmüller, Schubotz, Harth: "High Performance Linked Data Processing for Virtual Reality Environments". P&D ISWC 2014.

# Video: Manufacturing Control (2020)