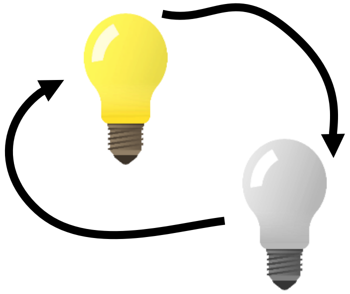
Executing Application Behaviour on the Web of Things – the Read-Write Linked Data Way

Tutorial Linked Data Techniques for the Web of Things – Part II
Andreas Harth and Tobias Käfer
8th International Internet of Things Conference, Santa Barbara (CA), USA, 2018

INSTITUTE FOR APPLIED INFORMATICS AND FORMAL DESCRIPTION METHODS, CHAIR FOR WEB SCIENCE

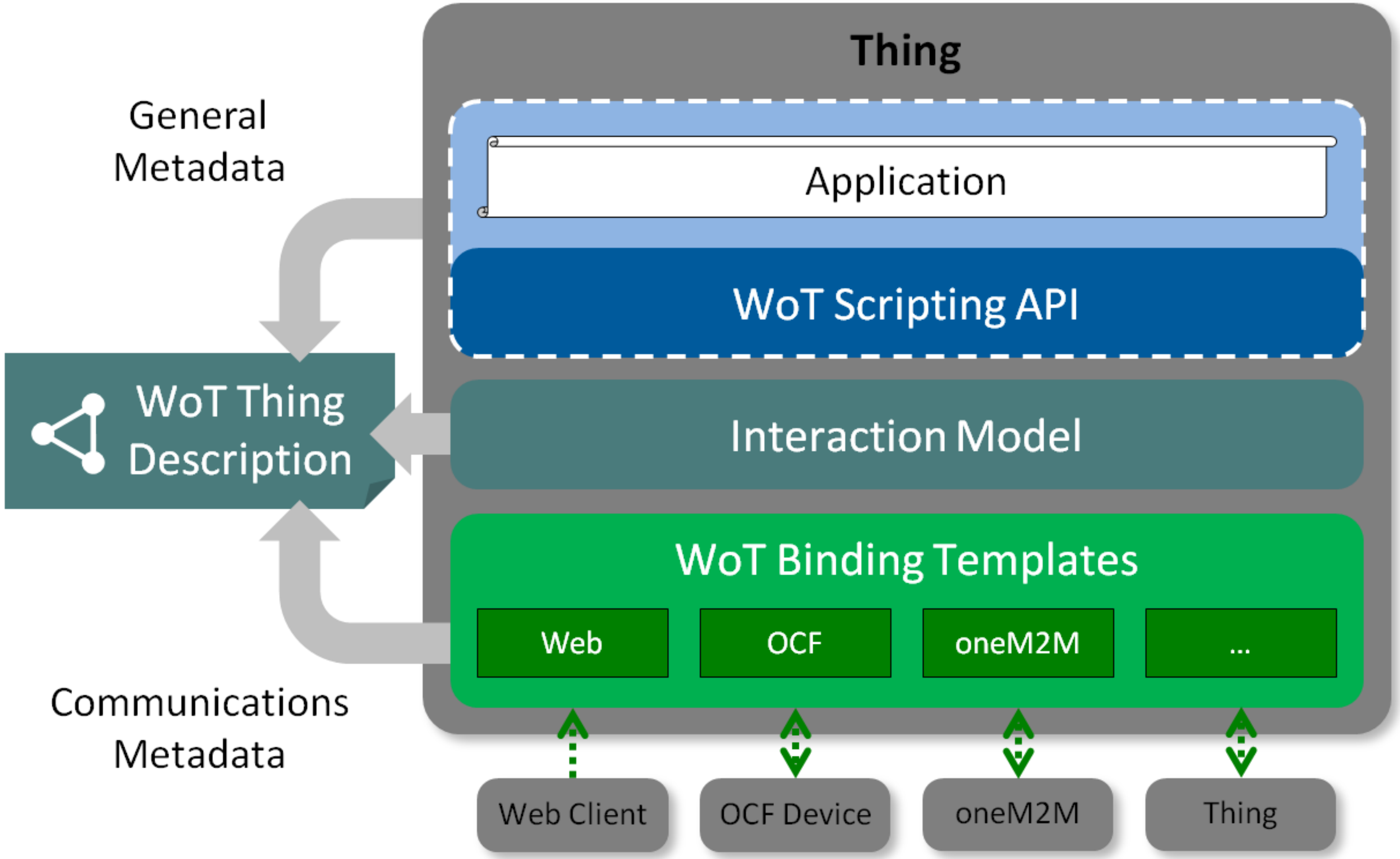
{ 

< 0.5 } => 

Agenda Part II

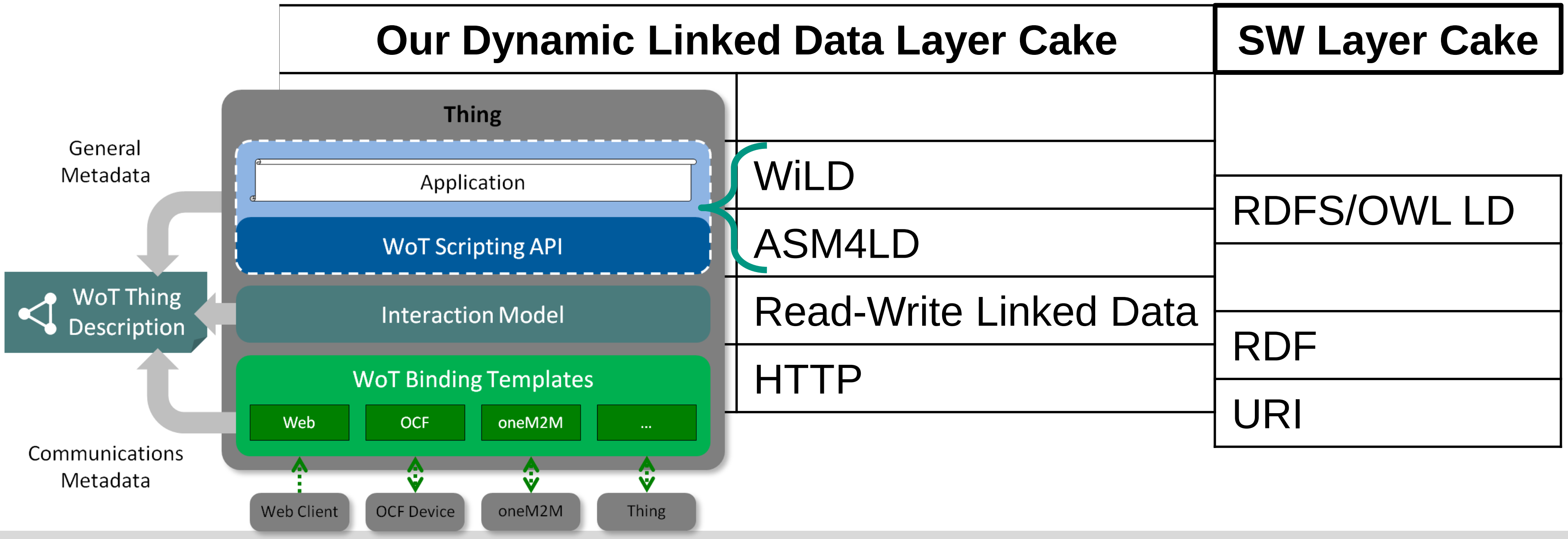
- Web of Things and Read-Write Linked Data
- Standards and practices around Read-Write Linked Data
- Building applications on Read-Write Linked Data
  - Rule-based
  - Workflow-based
- Related work
  - Building applications using web technologies and the cloud
  - Functional Descriptions around web technologies
- Conclusion

Web of Things Architecture [1]



[1] https://www.w3.org/TR/wot-architecture/

WoT, Semantic Web, Linked Data, and Agent Architectures



Institute for Applied Informatics and Formal Description Methods

Read-Write Linked Data and WoT Thing Descriptions

- Functional (wot Thing) Descriptions:
  - Interaction Patterns and interaction verbs
    - Property
      - readproperty, writeproperty, observeproperty
    - Action
      - invokeaction
    - Event
      - subscribeevent, unsubscribeevent
  - Interaction Models
    - Request-response
    - Publish-subscribe
    - message passing
  - Protocols
    - HTTP
    - MQTT
    - …

Protocol binding

There is a 1:1 mapping between the red interaction verbs and HTTP methods GET, PUT, POST

Read-Write Linked Data and the WoT Scripting API

Web of Things (WoT) Scripting API

…

2. Use Cases

- *This section is non-normative.*
- The following scripting use cases are supported in this specification:

2.1 Discovery

- ...

2.2 Consuming a Thing

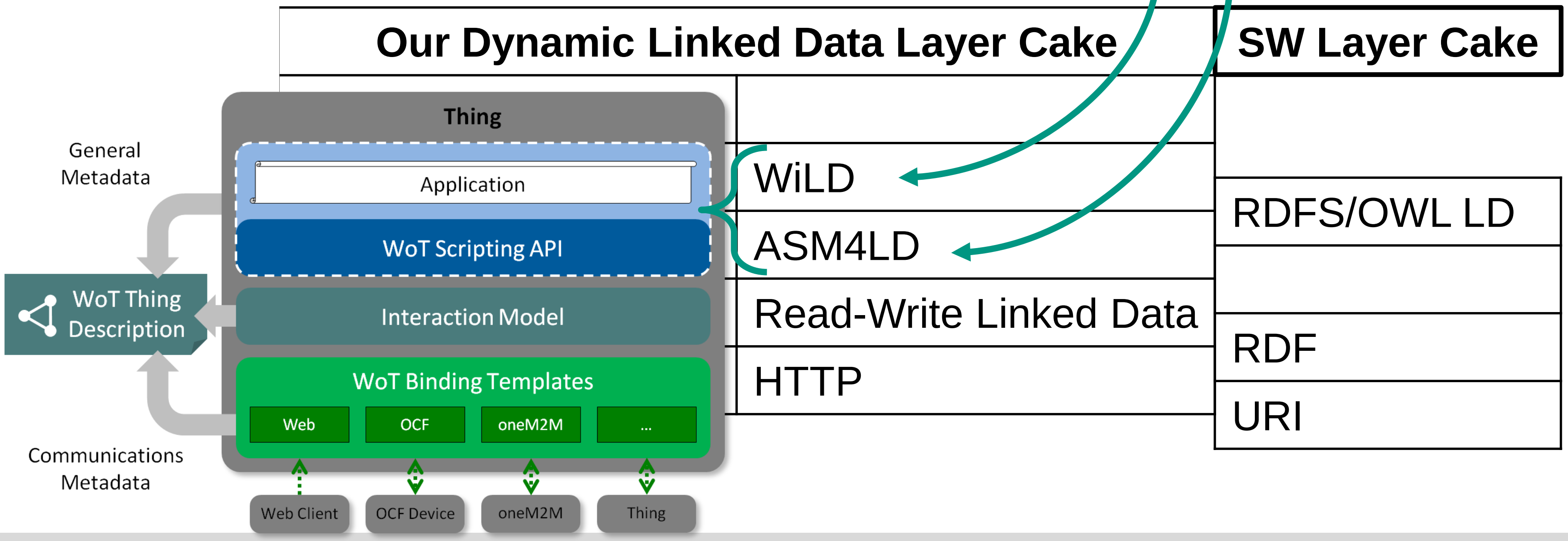
- …Fetch a TD and consume the thing (read the descriptions about the low level access APIs)…
- On a consumed Thing,
  - Read the value of a Property or set of properties.
  - Set the value of a Property or a set of properties.
  - Observe value changes of a Property.
  - Invoke an Action.
  - Observe Events emitted by the Thing.
  - Observe changes to the Thing Description of the Thing.
  - Get the Thing Description.
  - Get the list of linked resources based on the Thing Description.

+ Property value/action payload in RDF
≈ Read-Write Linked Data = HTTP access [RFC7230seqq]
+ RDF data

2.3 Exposing a Thing

Applications on the Layer Cakes?

- Declarative application specifications
  - Rule-based
  - Workflow-based



Institute for Applied Informatics and Formal Description Methods

Why?

- Read-Write Linked Data
  - Uniform interface of HTTP
  - Uniform data model of RDF
  - → Technologies for large-scale interoperability based on decentral information

- Rule-based specifications of applications
  - Declarative
  - Compatible with rule-based reasoning
  - Stateless

- Workflow-based specifications of applications
  - Where application state is required

So much on the big picture…

WEB STANDARDS AND PRACTICES AROUND READ-WRITE LINKED DATA

Tim Berners-Lee
Date: 2009-16-08, last change: $Date: 2013-08-16 18:45:00 $
Status: personal view only. Editing status: Not finished at all @@
Up to Design Issues

Read-Write Linked Data

> *There is an architecture in which a few existing or Web protocols are gathered together with some glue to make a world wide system in which applications (desktop or Web Application) can work on top of a layer of commodity read-write storage. The result is that storage becomes a commodity, independent of the application running on it.*

Introduction

The Linked Data article gave simple rules for putting data on the web so that it is linked. This article follows on from that to discuss allowing applications to write as well as read data.

Architectures

File write-back

The model is that all data is stored in a document (virtual or actual file) named with a URI. One way of changing the data is to overwrite the whole file with an HTTP PUT operation. Whereas typical Apache servers are not configured out of the box to accept PUT, when they are configured for WebDAV (The Web Distributed Authoring and Versioning specs) then

The Hypertext Transfer Protocol (HTTP) [RFC7230]

- Selected Properties of HTTP
  - Stateless
  - Request/response messages
  - Interaction with resources
  - Message: the current state of a resource
- Focus: requests that implement CRUD
  - Create, Read, Update, Delete, the basic operations of persistent storage [1]

CRUD Operation	HTTP Method
Read	GET
Update	PUT
Create	POST / PUT
Delete	DELETE

CRUD – HTTP Corresponcence

HTTP Method	Safe?	Idempotent?
GET	✓	✓
PUT		✓
POST		
DELETE		✓

Properties of HTTP Requests

  - POST
    - Append-to-collection vs. RPC
  - OPTIONS
    - Describe communication options
- NB: No events → polling

[1] James Martin: Managing the Data-Base Environment, Pearson (1983)

When Resource State is (Not) Sent/Received? – HTTP Message Semantics [RFC7231]

HTTP Request Method	HTTP Request, or Response Code	HTTP Message Semantics: The HTTP Message Body Contains…
GET	Request	Nothing
PUT	Request	State of the resource
POST	Request	Arbitrary data or state of resource
DELETE	Request	Nothing
any	Non-2xx	State of the request
GET	2xx	State of the resource
PUT	2xx	State of the resource or empty
POST	2xx	State of the request (refering to new resource)
DELETE	2xx	State of the request or empty

W3C Recommendation

W3C

Linked Data Platform 1.0

W3C Recommendation 26 February 2015

Abstract

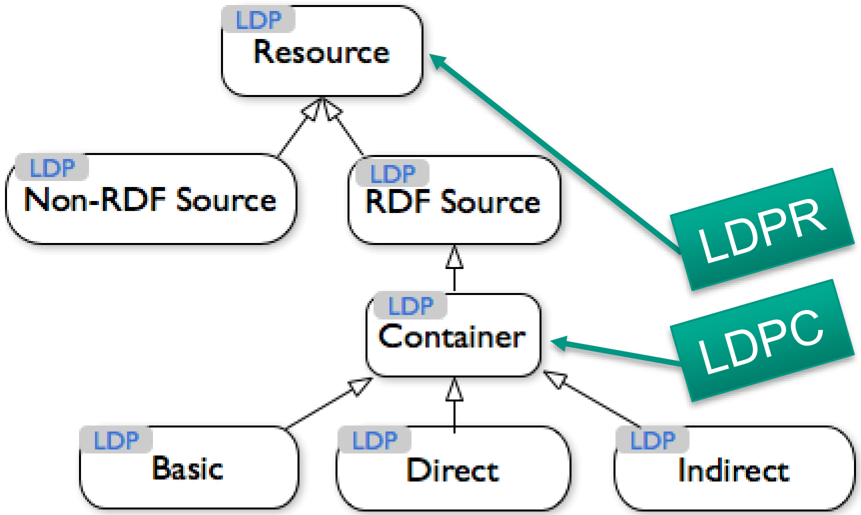
Linked Data Platform (LDP) defines a set of rules for HTTP operations on web resources, some based on RDF, to provide an architecture for read-write Linked Data on the web.

1. Introduction

This section is non-normative.

This specification describes the use of HTTP for accessing, updating, creating and deleting resources from servers that expose their resources as Linked Data. It provides clarifications and extensions of the rules of Linked Data [LINKED-DATA]:

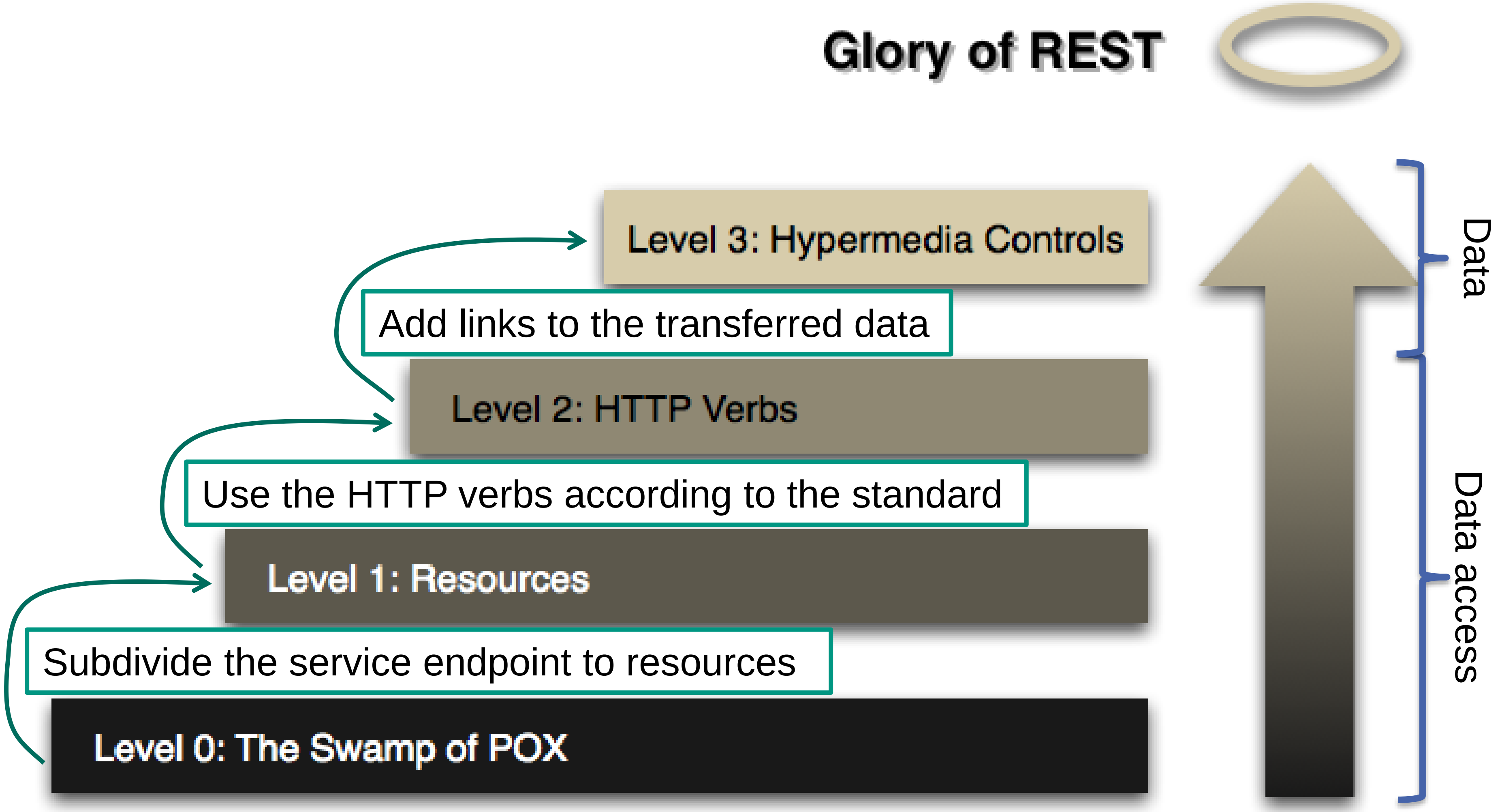
Linked Data Platform [1]



- Classification of resources →
- Clarifications for the use of the combination HTTP + RDF, eg.:
  - 4.2.8 HTTP OPTIONS and LDPR
    - 4.2.8.1 LDP servers MUST support the HTTP OPTIONS method.
    - 4.2.8.2 LDP servers MUST indicate their support for HTTP Methods by responding to a HTTP OPTIONS request on the LDPR's URL with the
HTTP Method tokens in the HTTP response header Allow.
  - 5.2.3 HTTP POST and LDPC
    - 5.2.3.1 LDP clients SHOULD create member resources by submitting a representation as the entity body of the HTTP POST to a known LDPC. If the resource was created successfully, LDP servers MUST respond with status code 201 (Created) and the Location header set to the new resource's URL. **Clients shall not expect any representation in the response entity body on a 201 (Created) response**.
- Cf. ATOM publishing protocol [RFC5023]: interactions with collections

[1] Speicher, Arwe, Malhotra: "Linked Data Platform 1.0" W3C Recommendation (2015)

REST (Representational State Transfer) [1] and the Richardson Maturity Model (RMM) for Services [2]



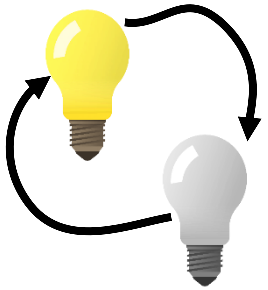
[1] Fielding: "Architectural Styles and the Design of Network-based Software Architectures". PhD Thesis, UC Irvine, USA (2000)
[2] Fowler: "Richardson Maturity Model" (2010) available from http://martinfowler.com/articles/richardsonMaturityModel.html

Käfer and Harth: “Rule-based Programming of User Agents for Linked Data”. Proc. 11th Workshop on Linked Data on the Web (LDOW) at the 27th Web Conference, 2018.

SPECIFYING APPLICATIONS FOR READ-WRITE LINKED DATA USING RULES

Example

- A light sensor available as Linked Data
- A LED available as Read-Write Linked Data

{  < 0.5}=> 

- Turn on the LED if the light sensor‘s value drops below a certain threshold

Developing ASM4LD

- HTTP GET / Readproperty
- To retrieve the CURRENT state of one device
- To retrieve the CURRENT state of multiple devices / systems
  → Retrieve the world state in RDF

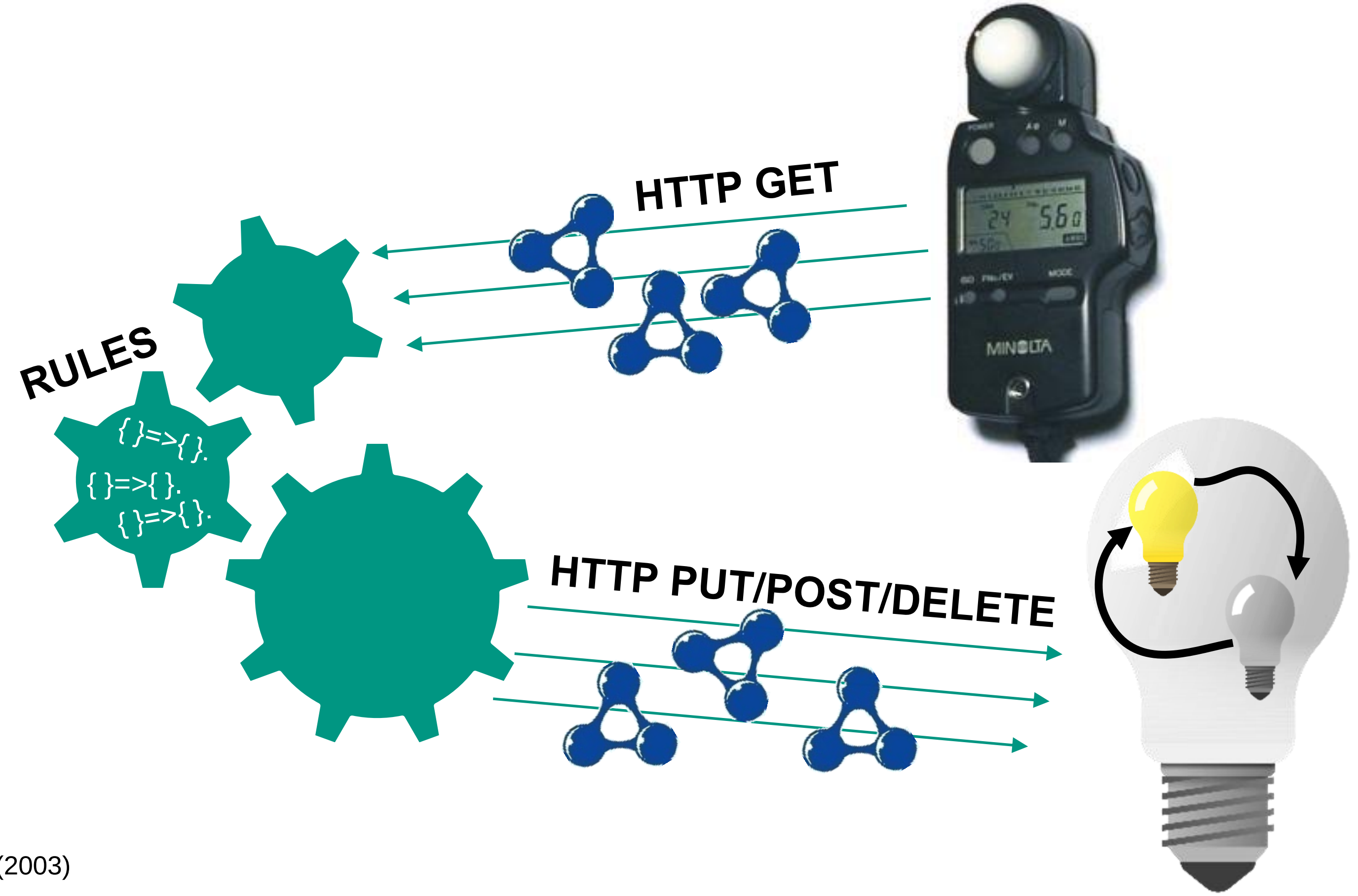
- ASK Queries on the world state → Conditions for actions

- HTTP PUT / Writeproperty
  → Actions
  → Set the state of components

ASM4LD-based User Agents for Read-Write Linked Data

- Aim: Execution of agent specifications on Read-Write Linked Data
- Approach:
  - Directly operate on the world state
  - Inspired by Simple Reflex Agents [1]
- In a nutshell:

```
while(true):
    sense()
```
  - HTTP-GET
```
    think()
```
  - Queries in rule bodies
```
    act()
```
  - HTTP-PUT/POST/DELETE



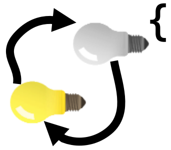
[1] Russell & Norvig: Artificial Intelligence – A Modern Approach. Prentice Hall (2003)

ASM4LD – a Model of Computation for Read-Write Linked Data [LDOW2018]

- Abstract state machines [2]
- Model-theoretic semantics of RDF Graphs
- Semantics of RDF Datasets
- Semantics of HTTP

ASM4LD [1]

- Supports Read-Write Linked Data:
  - URIs to identify things
  - HTTP for interaction
  - RDF for describing data
  - Writing to Linked Data
- Embraces reasoning
- Embraces link following
- Turing complete
- Engine: http://linked-data-fu.github.io/



{ < 0.5}=>

```
{ [] a http:Request ;
    http:hasMethod httpM:GET ;
    http:requestURI </ambient/light> . }
{ [] a http:Request ;
    http:hasMethod httpM:GET ;
    http:requestURI </led/1> . }

{ </ambient/light> rdf:value ?val .
  ?val math:lessThan 0.5 .
  </led/1#led> :isOn false . }
=>
{ [] a http:Request ;
    http:hasMethod httpM:PUT ;
    http:requestURI </leds/1> ;
    http:body
      { </led/1#led> :isOn true . } . } .
```

SENSE: Retrieve the world state

THINK: Conditionally…

ACT: …manipulate the world state

[LDOW2018] Käfer and Harth: Rule-based programming of user agents for Linked Data. Proc. 11th LDOW@TheWebConf (2018)
[2] Gurevich:. "Evolving algebras 1993: Lipari guide." *Specification and validation methods* (1995)

ASM4LD DEMO

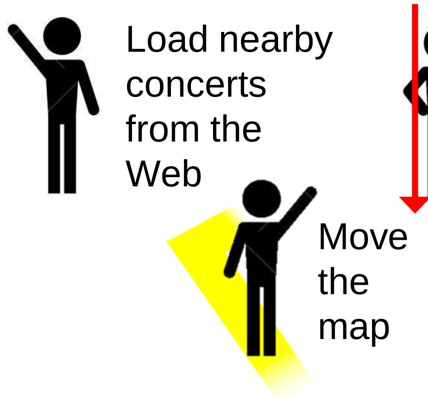
A Binary Counter in ASM4LD for 2 LEDs of a Tessel 2 [1]

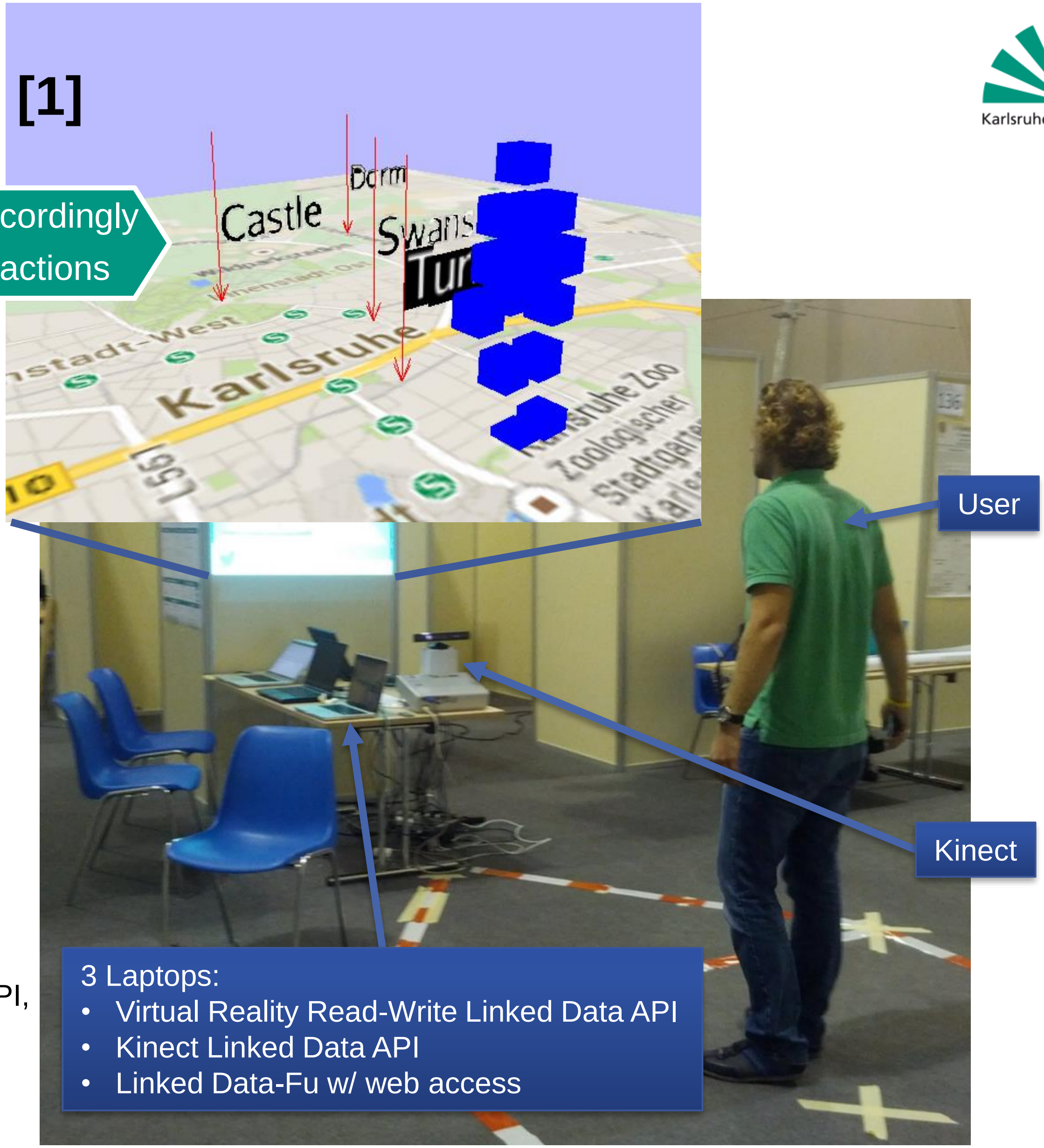
```
@prefix http: <http://www.w3.org/2011/http#>.
@prefix http_m: <http://www.w3.org/2011/http-methods#>.
@prefix saref: <https://w3id.org/saref#> .

# For the URIs to the LEDS
@prefix leds: <http://t2-rest-leds.lan/leds/> .

# Data retrieval:
{ _:h http:mthd http_m:GET ; http:requestURI leds: . }
{ ?x <http://www.w3.org/ns/sosa/hosts> ?y . } => { [] http:mthd http_m:GET; http:requestURI ?y . } .

# The logic:
{ leds:2#led saref:hasState saref:Off . }
=>
{ _:h http:mthd http_m:PUT ; http:requestURI leds:2 ; http:body { leds:2#led saref:hasState saref:On . } . } .

{ leds:2#led saref:hasState saref:On . }
=>
{ _:h http:mthd http_m:PUT ; http:requestURI leds:2 ; http:body { leds:2#led saref:hasState saref:Off . } . } .

{ leds:2#led saref:hasState saref:On .
  leds:3#led saref:hasState saref:Off . }
=>
{ _:h http:mthd http_m:PUT ; http:requestURI leds:3 ; http:body { leds:2#led saref:hasState saref:On . } . } .

{ leds:2#led saref:hasState saref:On .
  leds:3#led saref:hasState saref:On . }
=>
{ _:h http:mthd http_m:PUT ; http:requestURI leds:3 ; http:body { leds:2#led saref:hasState saref:Off . } . } .
```

[1] https://github.com/kaefer3000/t2-rest-leds

Linked Data-Fu Example: Distributed VR System Composition [1]

Kinect tracks user → Avatar moves accordingly / Gestures trigger actions

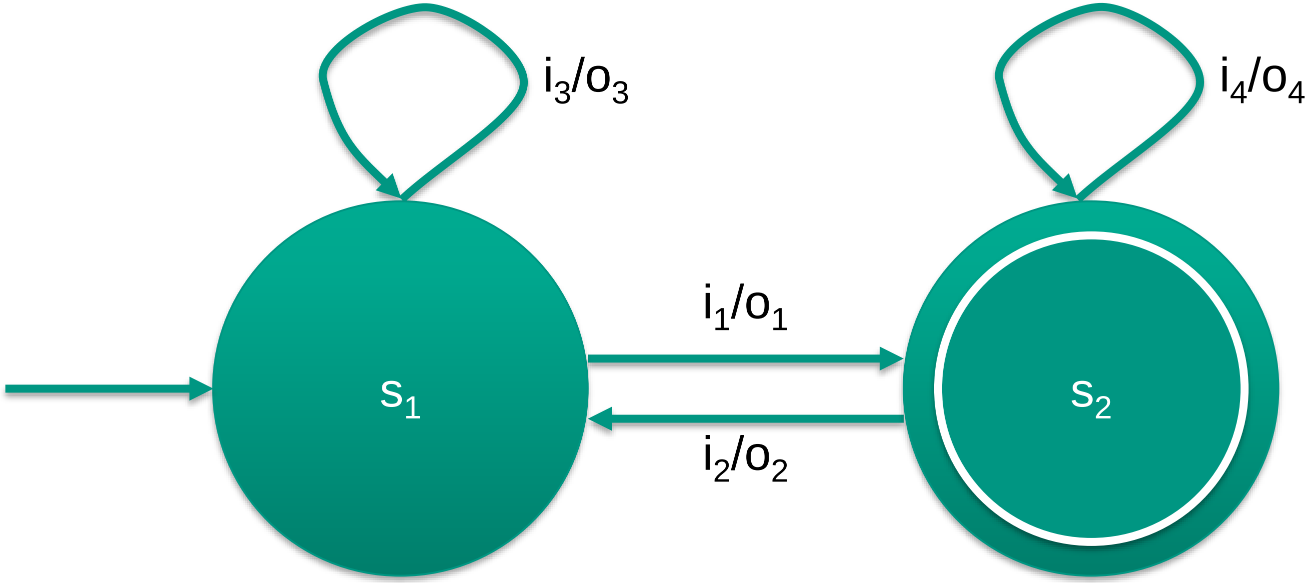




- We encoded in Linked Data-Fu rules:
  - Movement of the avatar according to Kinect data
  - Detection of user gestures
  - Movement of the map according to gestures
  - Loading of concert data from the web
  - Data integration between VR RWLD API, concert LD API, Kinect LD API
- Execution at Kinect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

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

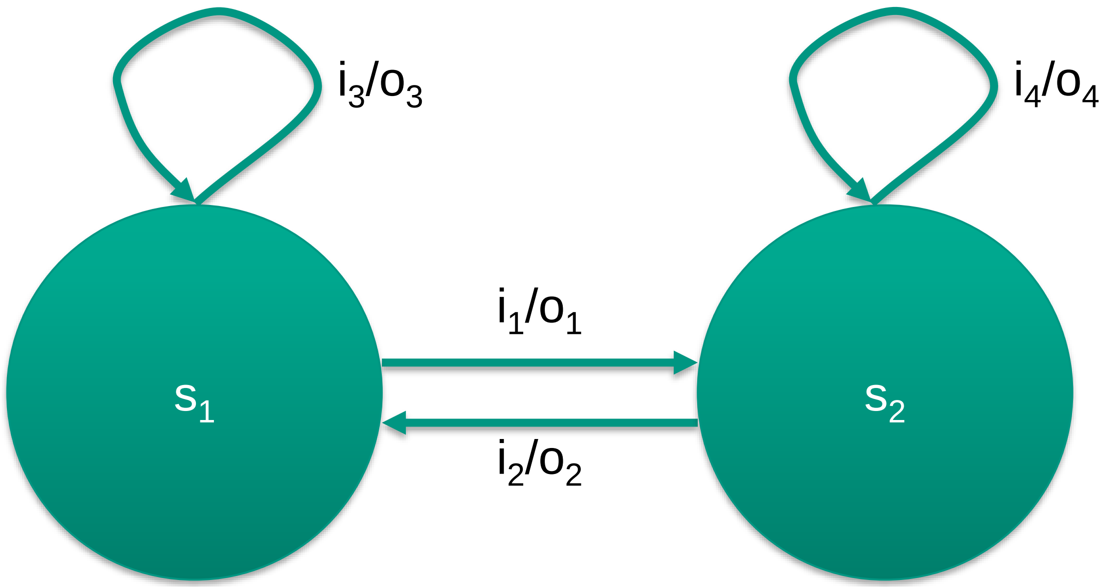
Finite State Machines (Mealy Automata) and Transition Systems



A State Machine

i_n: input
o_n: output

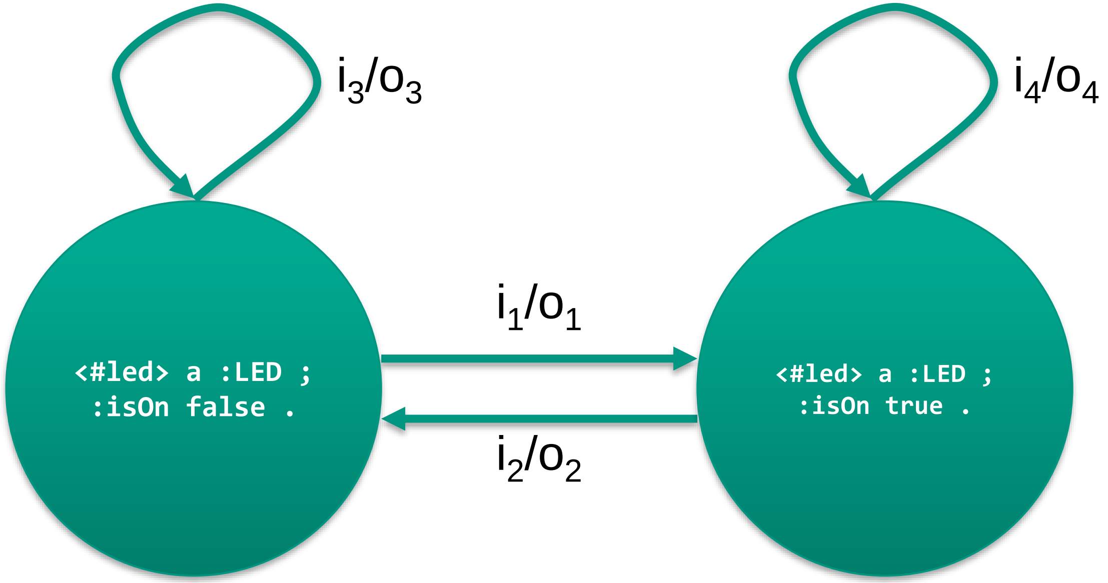
Finite State Machines (Mealy Automata) and Transition Systems



A Transition System

i_n: input
o_n: output

Describing an Origin Server in an Linked Data Transition System [1]

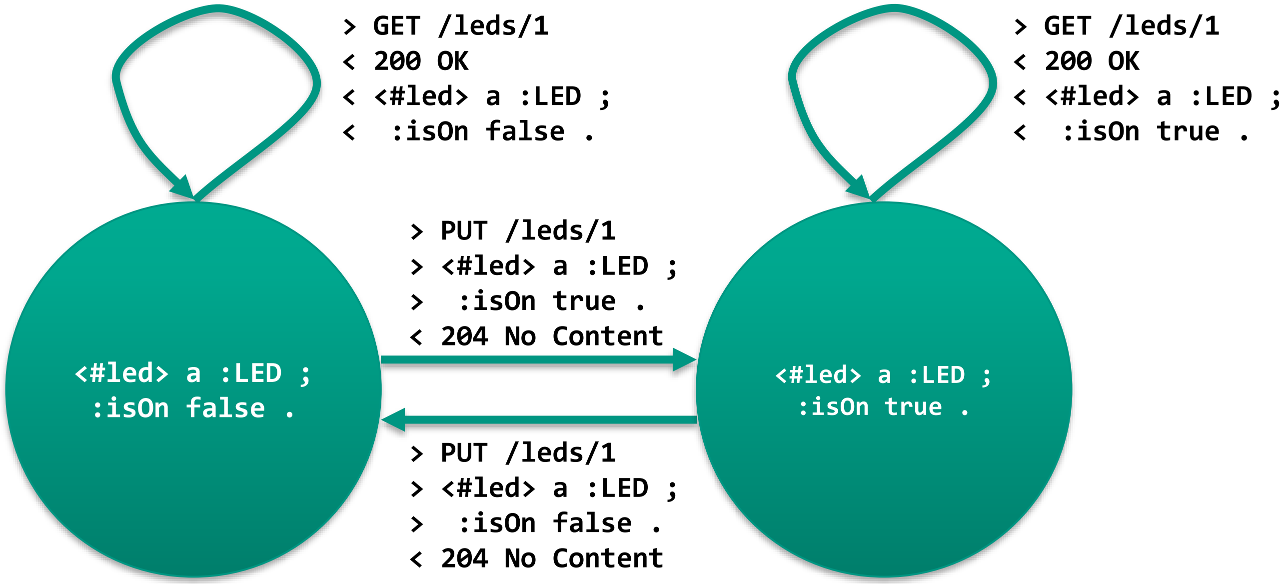


Linked Data Transition System of Resource /relay/1 on server http://t2-ambient-relay.lan/

i_n: input
o_n: output

[1] Harth and Käfer: "Towards Specification and Execution of Linked Systems". Proceedings of the 28th GI-Workshop on Foundations of Database Systems (Grundlagen von Datenbanken, GvD), May 24 - 27, 2016, Nörten-Hardenberg, Germany.

Describing an Origin Server in an Linked Data Transition System [1]



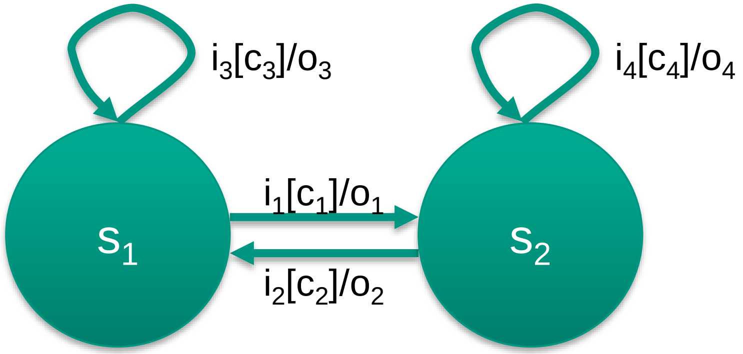
Transition System of Resource /leds/1 on server http://t2-led.lan/

From the perspective of the resource on the server:
> denote incoming requests, cf. inputs in automata terminology
< denote outgoing responses, cf. outputs in automata terminology

[1] Harth and Käfer: “Towards Specification and Execution of Linked Systems”. Proceedings of the 28th GI-Workshop on Foundations of Database Systems (GvD), 2016

Labelled Transition Systems

- Labelled Transition System $LTS = (S, L, \rightarrow)$
  - S: Set of States
  - L: Set of Labels
    - Typically some of: input/event, condition, output/action
  - $\rightarrow \subset (S \times L \times S)$: Transition Relation



A Transition System

→ How can we describe Dynamic Linked Data as Transition System?

RDF Dataset

- Definition [1]
  - A collection of RDF graphs G
  - Each graph has a URI u as name
  - The default graph has an empty name
  - No restriction on the relation graph – name

- A Linked Data view [2, section 3.5]:
  - Name = the information resource‘s URI where the graph can get obtained

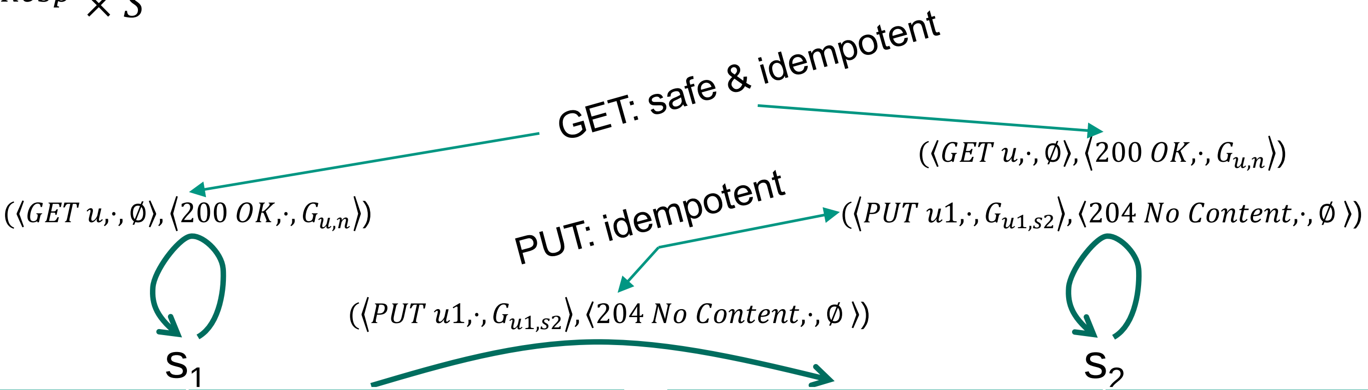
Name	RDF Graph
u_1	G_{u1}
`/leds/1`	`<#led> a :LED ; :isOn false .`

An RDF Dataset

[1] Cyganiak, Wood, Lanthaler (eds.): “RDF 1.1 Concepts and Abstract Syntax”. W3C Recommendation (2014)
[2] Zimmermann (ed.): “RDF 1.1 On Semantics of RDF Datasets”. W3C WG Note (2014)

Linked Data Transition System [1]

- Linked Data Transition System $LDTS \coloneqq (S, \rightarrow)$
  - S : Possible states of all resources (set of RDF datasets)
    - $s_n = \{(u, G_{u,n})\} \in S$: RDF dataset at point in time n
  - $\rightarrow$: Transition Relation
    - $\rightarrow \subseteq S \times 2^{Req,Resp} \times S$

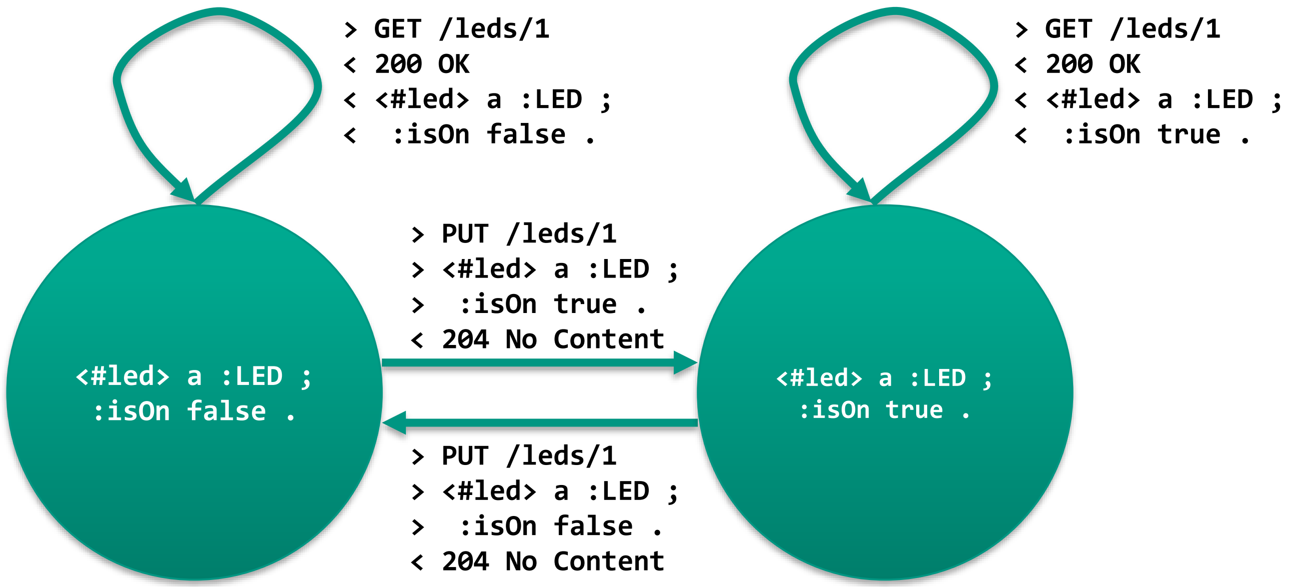


Name	RDF Graph
u_1	$G_{u1,s1}$
`/leds/1`	`<#led> a :LED; :isOn false .`

Name	RDF Graph
u_1	$G_{u1,s2}$
`/leds/1`	`<#led> a :LED; :isOn true .`

[1] Harth and Käfer: "Towards Specification and Execution of Linked Systems". Proceedings of the 28th GI-Workshop on Foundations of Database Systems (GvD), 2016.

State Machines, Transition Systems, and Linked Data [1]



Transition System of Resource /leds/1

[1] Harth and Käfer: "Towards Specification and Execution of Linked Systems". Proceedings of the 28th GI-Workshop on Foundations of Database Systems (GvD), 2016.

Käfer and Harth. “Specifying, Monitoring, and Executing Workflows in Linked Data Environments”. Proc. 17th International Semantic Web Conference (ISWC), 2018.

SPECIFYING APPLICATIONS FOR READ-WRITE LINKED DATA USING WORKFLOWS

Fine-granular distributed components with REST / RWLD intefaces everywhere

Karlsruhe Institute of Technology

- How to create integrated applications that maintain application state? Use workflows [1]?
- → We need a solution that combines:
  - Workflows
  - Semantic reasoning
  - RESTful interaction

WiLD
(Workflows in Linked Data)

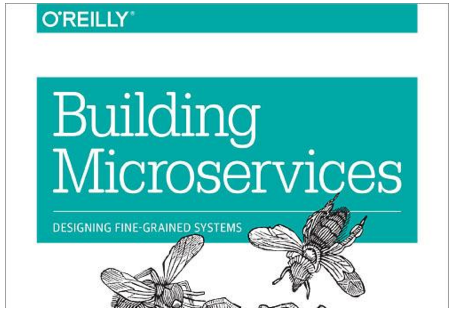
Uniform Interface: Read-Write Linked Data
- REST as uniform interaction mechanism between systems
- RDF as uniform data model, ready for semantic data integration

WEB OF THINGS

- Interfaces to IoT sensors/actuators
- Built on Linked Data

inrupt

- Interfaces to personal data storages
- Built on Linked Data



- Interfaces to core company functions
- Built on REST (lift to Linked Data)

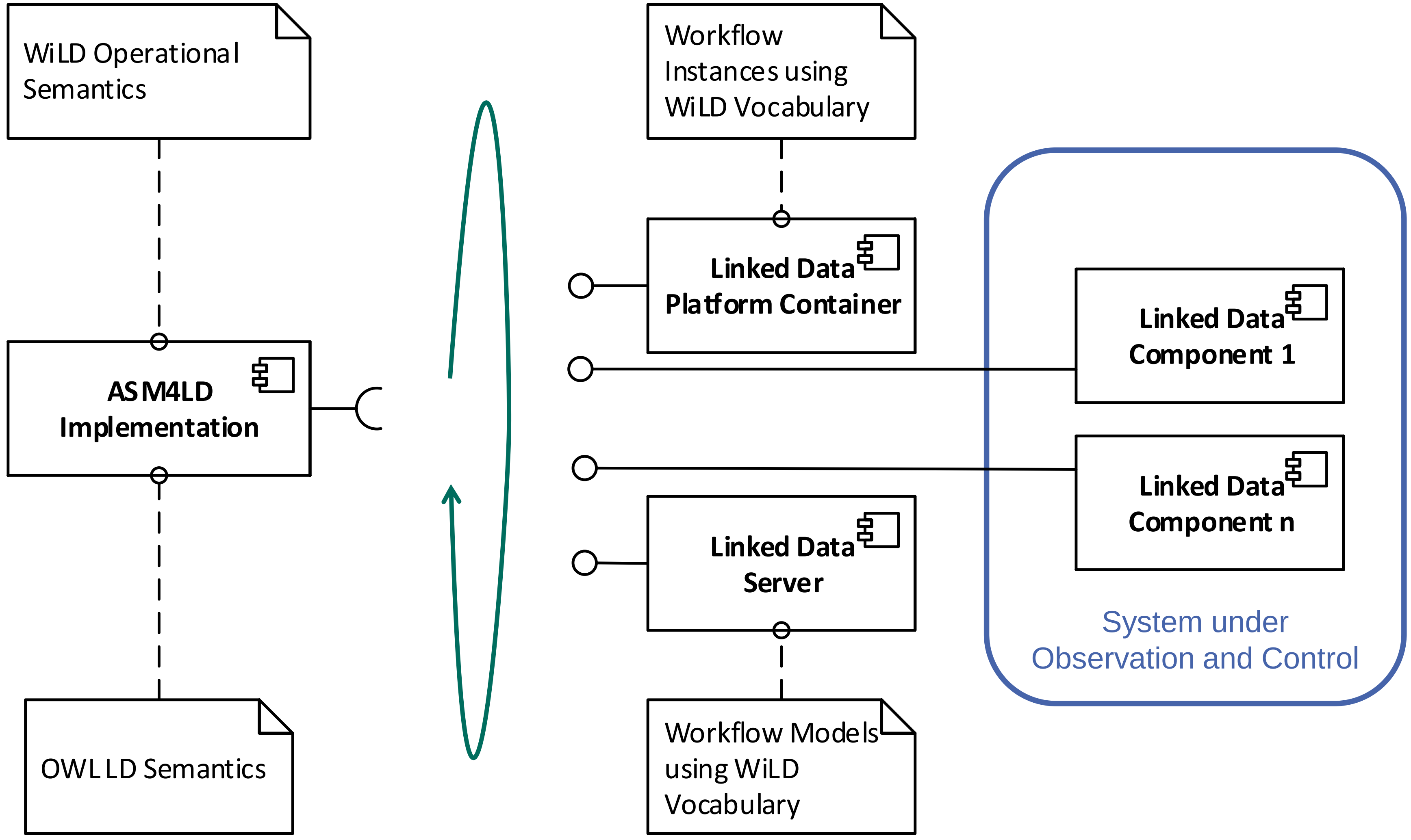
[1] Jablonski and Bussler: Workflow Management. London: International Thompson (1996)

WiLD in our Layer Cakes

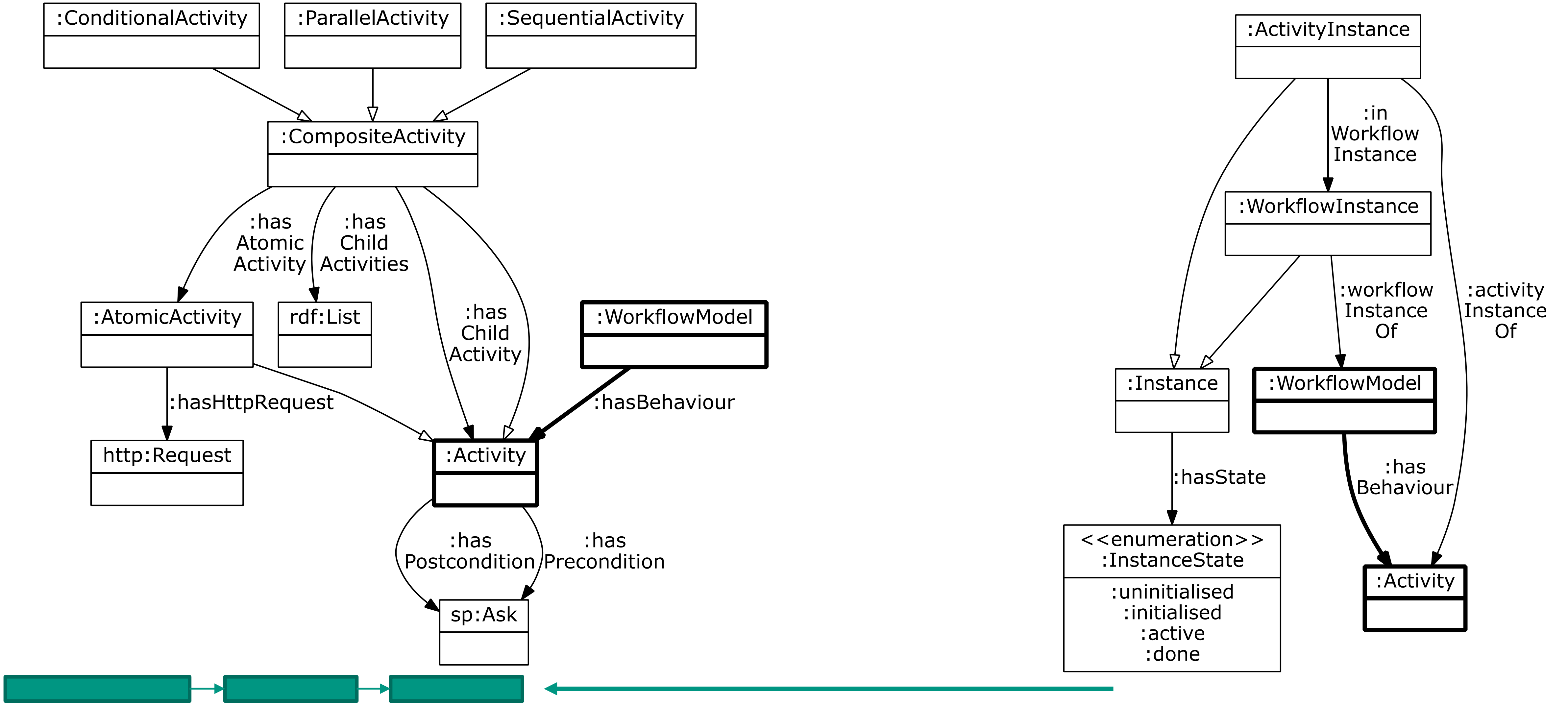
Russell / Norvig‘s Agent Layer Cake	**Our Dynamic Linked Data Layer Cake**	
Agents with goals		
Agents with internal state	*Workflow Meta Model*	**WiLD**
Simple reflex agents	*Model of Computation*	ASM4LD
	Data Model + Access	Read-Write Linked Data
	System Interaction	HTTP

SW Layer Cake
RDFS/OWL LD
RDF
URI

On our Workflow Management System Architecture and Overall System State

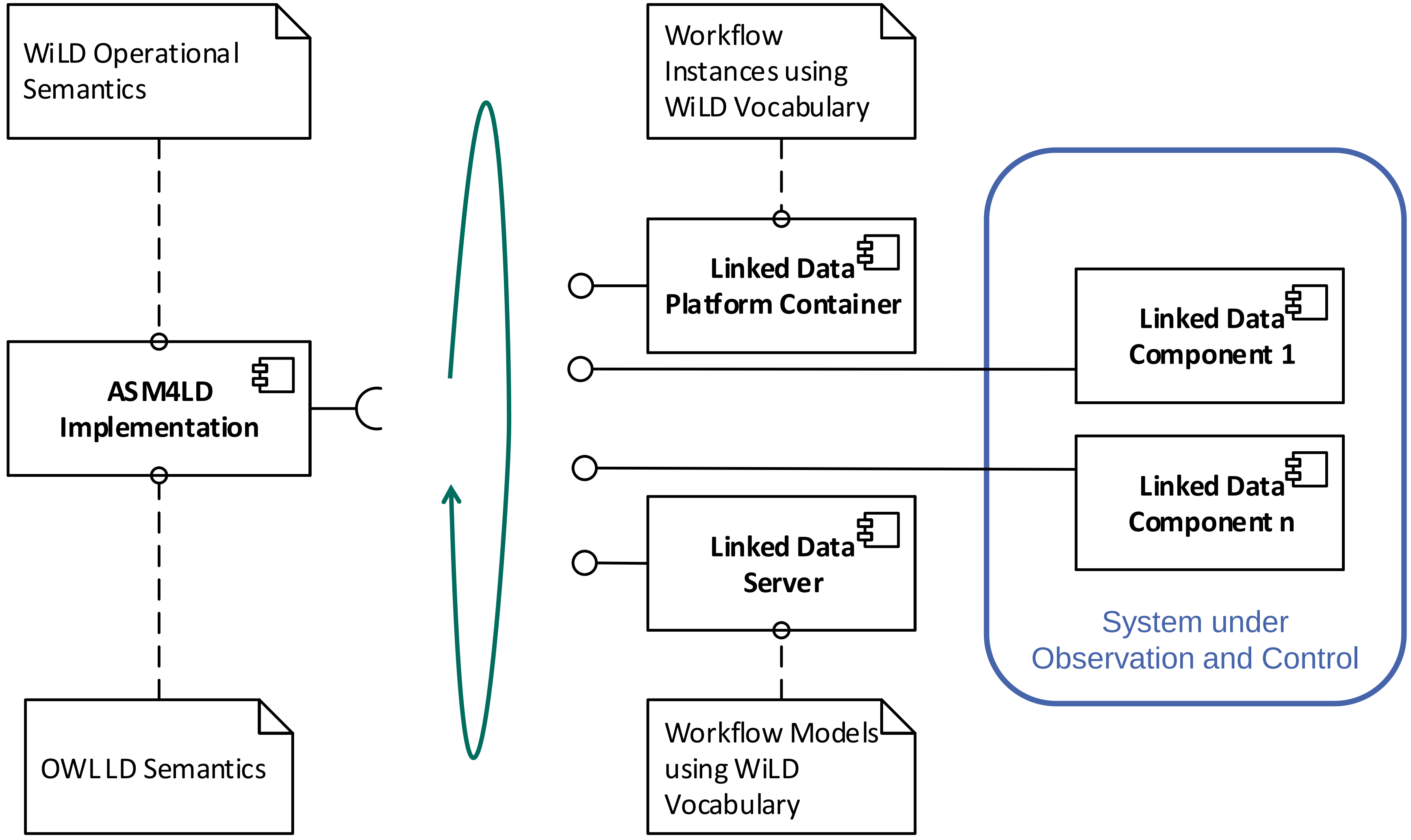


The WiLD Ontology – our Workflow Language for Workflow Models and Instances



Excerpt

On our Workflow Management System Architecture and Overall System State

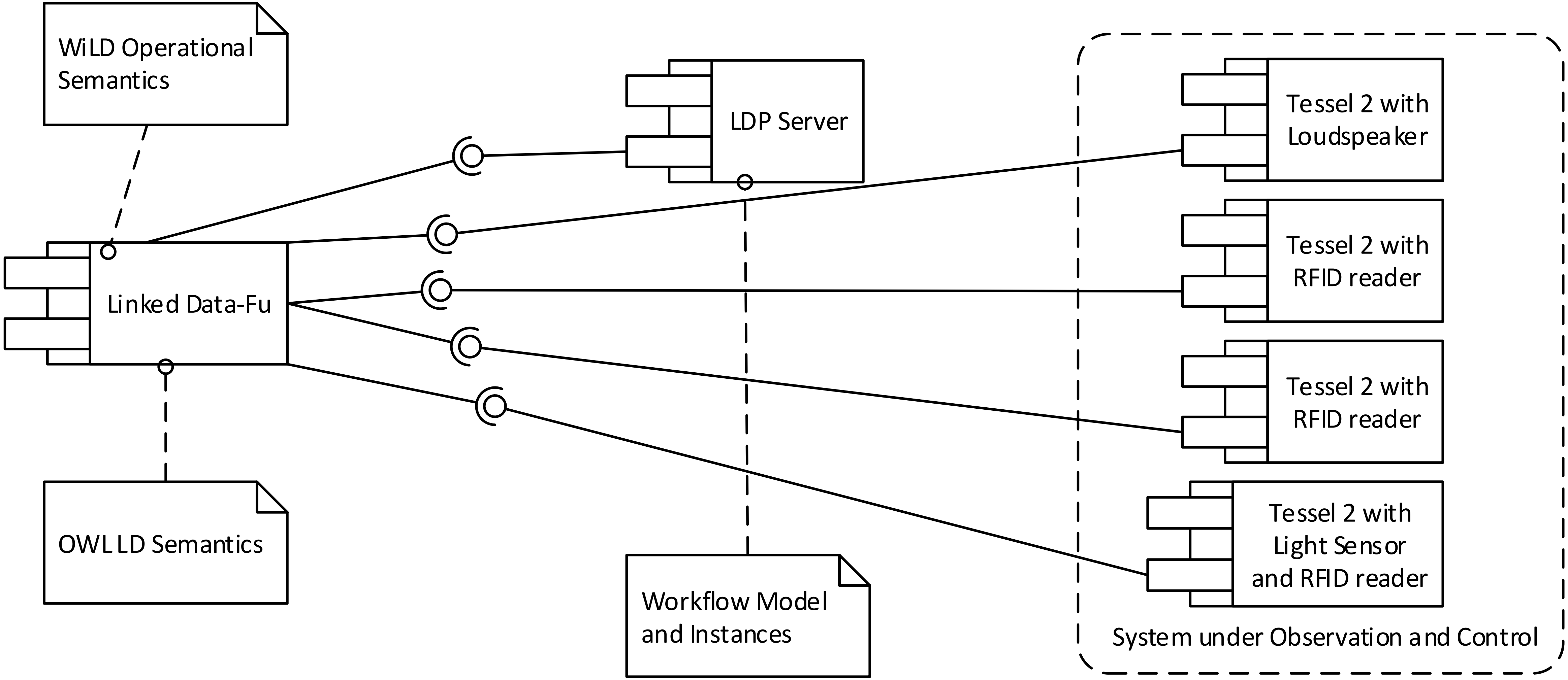


Käfer, Lauber, Harth: “Using Workflows to Build Compositions of Read-Write Linked Data APIs on the Web of Things”. Posters & Demos at the 17th International Semantic Web Conference, 2018.

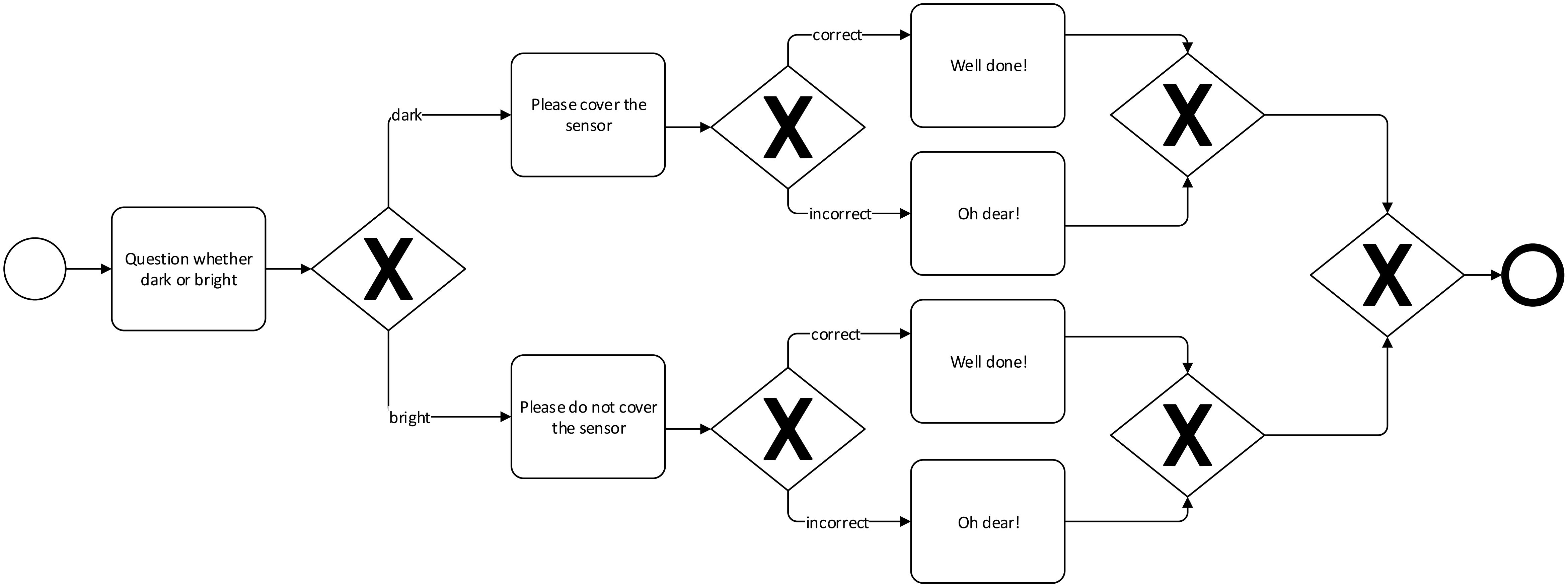
WiLD DEMO

Institute for Applied Informatics and Formal Description Methods

Set-up

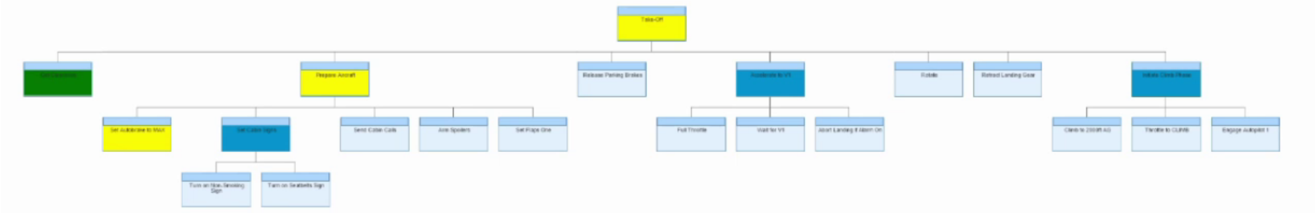

WiLD Operational Semantics
LDP Server
Tessel 2 with Loudspeaker
Tessel 2 with RFID reader
Tessel 2 with RFID reader
Tessel 2 with Light Sensor and RFID reader
Linked Data-Fu
OWL LD Semantics
Workflow Model and Instances
System under Observation and Control

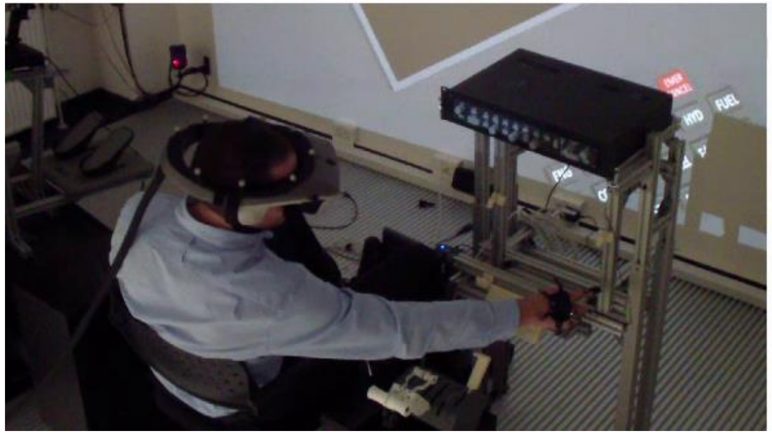
Workflow Model



WiLD Example: i-VISION

"SELECT the push-buttons in the Virtual Reality that are involved in the upcoming steps of the currently running take-off workflow and highlight them"





http://www.ivision-project.eu/

RELATED WORK

Mapping the Field of Service Composition and Web Agent Specification

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	…
Dynamics model	ASM	LTS	Situation Calculus	Unformalised Implementation	…		
Data model	Graph (RDF)	Nested (JSON, XML)	…				
Data access	RMM2	RMM1	RMM0	push	…		

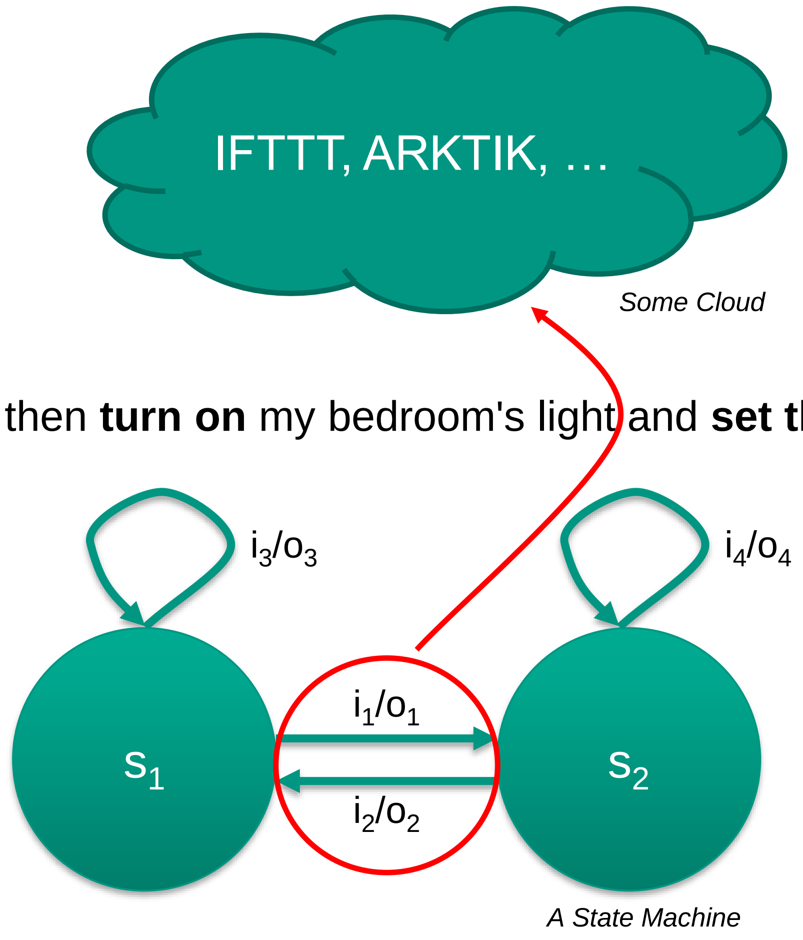
*Semantics of BPEL have been given eg. in Petri Nets and ASMs, but Petri Nets are also used to describe compositions

Related Work

BUILDING APPLICATIONS

IFTTT [1] and ARKTIK [2] (&Co.)

- IFTTT "if-this-then-that"
  - Automate tasks on the web
    - Eg. "If I tweet, post that also on Facebook"
- Samsung ARKTIK rules
  - Automate on the Internet of Things
    - Eg. "If the temperature of the room is **more than 72°F**, then **turn on** my bedroom's light and **set the color** to red" (sic!)
- …Centralised Platforms
- Event-Action rules
- Events = Notifications from devices/APIs



[1] https://ifttt.com/maker_webhooks
[2] https://developer.artik.cloud/documentation/data-management/develop-rules-for-devices.html

Turn on the Light using IFTTT Maker Channel

Maker event “light_state_change”

- Create an account, register key
- Register event type, eg. light_state_change
- Whenever there is a change:
  - POST to `https://maker.ifttt.com/trigger/light_state_change/with/key/{secret_key}`
  - HTTP body (must be JSON, keys have to be named exactly like that):

```
{ "value1“ : "test",
  "value2“ : 0.5 ,
  "value3“ : True }
```

Adapted from http://www.makeuseof.com/tag/ifttt-connect-anything-maker-channel/

Turn on the Light using ARKTIK

```
{
 "if": {
  "and": [ {
   "sdid": "sensor123" ,
   "field": "value",
   "operator": "<",
   "operand": 0.5
  } ]
 },
```

```
 "then": [ {
  "action": "httpRequest",
  "parameters": {
   "method": { "value": "PUT" },
   "url": {
   "value":
 "http://t2-ambient-relay.lan/relay/1"
   },
   "body":
    {
"@id" : "#r",    "http://example.org/isOn" : true
    }
   }
  } ]
}
```

Classifying IFTTT and ARKTIK (&Co.)

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	…
Dynamics model	ASM	LTS	Situation Calculus	Unformalised Implementation	…	
Data model	Graph (RDF)	Nested (JSON, XML)	…			
Data access	RMM2	RMM1	RMM0	push	…	

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

ASM4LD / Looped Linked Data-Fu

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	…
Dynamics model	ASM	LTS	Situation Calculus	Unformalised Implementation			…
Data model	Graph (RDF)	Nested (JSON, XML)					…
Data access	RMM2	RMM1	RMM0	push			…

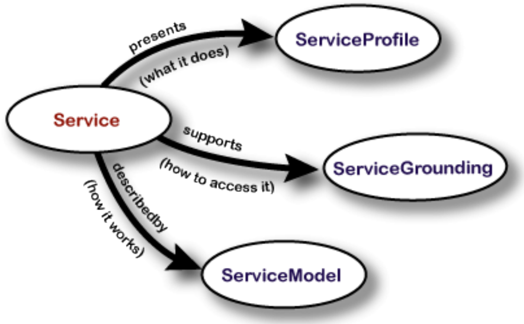
*Semantics of BPEL have been given eg. in Petri Nets and ASMs, but Petri Nets are also used to describe compositions

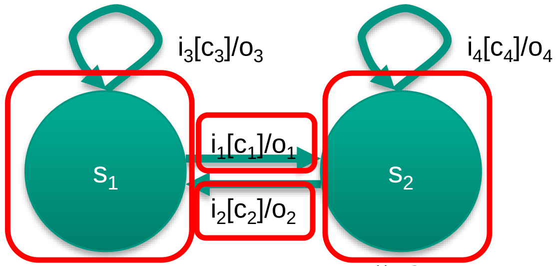
Related Work

SERVICE AND AFFORDANCE DESCRIPTIONS

OWL-S

- OWL-S: for descriptions of (SOAP) web services
  - Aim: Automated web service discovery, invocation, composition, monitoring
  - WSDL descriptions of web services (SOAP) do not suffice
- OWL-S Service Profile / Model:
  - Functionality description of a service
  - Profile: “Advertising” eg. to be put in a registry for service discovery
  - Model: For service composition and invocation
  - Contents (~ for both Profile and Model):
    - Input (what to give to a service when invoking)
    - Output (what the service will return when invoked)
    - Precondition (what has to hold before the service invocation)
    - Effect / Postcondition / Result (what holds after the service invocation)
- Results of a composition:
  - BPEL, Proofs, … [1]





A Transition System

http://www.w3.org/Submission/OWL-S/

[1] Baryannis and Plexousakis: “Automated Web Service Composition: State of the Art and Research Challenges”. ICS-FORTH/TR-409 (2010)

Classifying OWL-S and WSMO

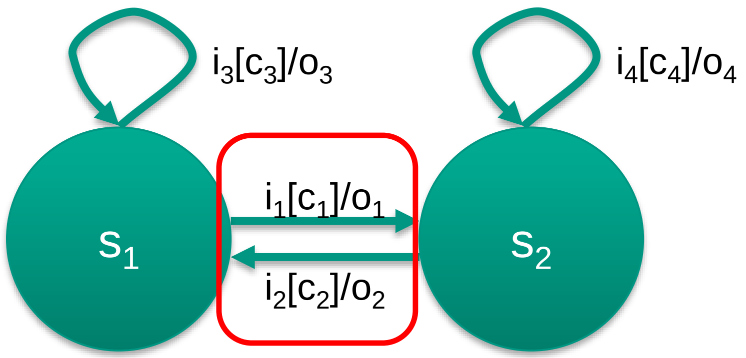
Level	Foundational approaches / categories						
Capability description	Input, Output, Precondition, Effect (for automated composition)	Affordance (for manual composition)	…				
Composition description	Rules	BPEL *	Pi cal-culus	Petri Nets	(Temporal) logic	Unformalised Implementation	…
Dynamics model	ASM	LTS	Situation Calculus	Unformalised Implementation	…		
Data model	Graph (RDF)	Nested (JSON, XML)	…				
Data access	RMM2	RMM1	RMM0	push	…		

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

Next: Affordance Descriptions

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	…
Dynamics model	ASM	LTS	Situation Calculus	Unformalised Implementation			…
Data model	Graph (RDF)	Nested (JSON, XML)					…
Data access	RMM2	RMM1	RMM0	push			…

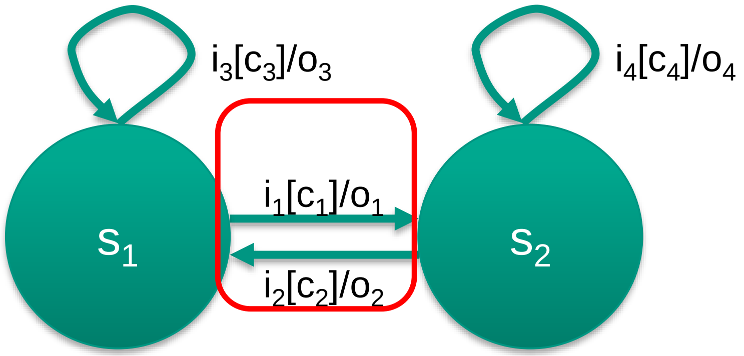
*Semantics of BPEL have been given eg. in Petri Nets and ASMs, but Petri Nets are also used to describe compositions



A Transition System

Hydra

- Motivation:
  - Many web APIs do essentially similar things using differing terminology
  - With some standardisation, we could build generic agents
- Hydra: an API documentation standardisation effort building on established technologies:
  - Linked Data vocabularies, JSON-LD, and HTTP headers
- Contents
  - Links between resources that allow for requests
  - Possible requests
  - Required data in requests
  - Detailing out HTTP status information
- Similar and related concepts
  - LDP, ATOM (Collections)
  - HTTP headers (Allow)

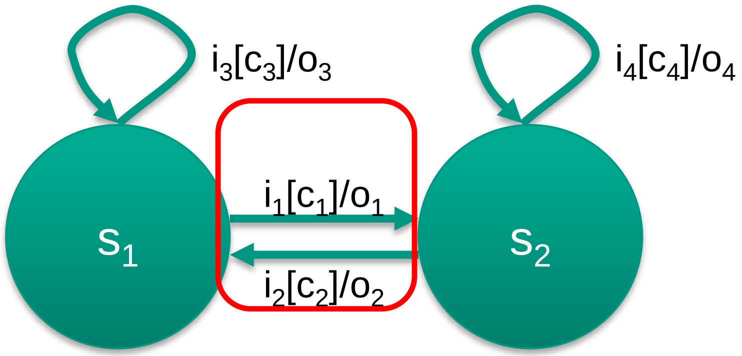


A Transition System

http://www.hydra-cg.com/

schema.org Potential Actions and WoT Actions

- schema.org Potential actions
  - Typed actions (eg. BuyAction)
  - Optional fields include:
    - Input and output schema
    - Result
    - Target
    - HTTP method



A Transition System

- WoT Thing Descriptions
  - Defines (for a thing):
    - Actions (horizontal arrows)
    - Properties
    - Events
  - Well-known relative URIs for actions and properties of a thing
  - Requirements on the use of HTTP and resource representations

http://schema.org
http://w3c.github.io/wot-thing-description/

LIDS [1]

- Motivation:
  - Web APIs provide non-RDF output data for some input values
  - Even if we lift the output to RDF, the relation between input and output is missing
  - With some descriptions of the Web API, we can relate the inputs to the lifted output
- Example:
  - We want **`foaf:based_near`** triples for places characterised using **`geo:lat`** and **`geo:long`**
  - We have the description

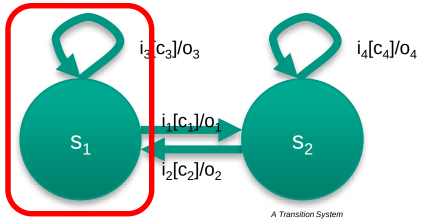
```
CONSTRUCT { ?point foaf:based_near ?feature }
FROM      <http://geowrap.openlids.org/findNearbyWikipedia>
WHERE     { ?point geo:lat ?lat . ?point geo:long ?lng }
```

  - We query for the WHERE clause in the data we already have

```
SELECT * WHERE { ?point geo:lat ?lat . ?point geo:long ?lng }
```

  - We call the API with the variables from the WHERE clause (that do not appear in the CONSTRUCT) as parameters and get back data like described in the CONSTRUCT

```
> GET /findNearbyWikipedia?lat=37.416&lng=-122.152#point HTTP/1.1
> Host: geowrap.openlids.org
< 200 OK
<http://geowrap...Wikipedia?lat=37.416&lng=-122.152#point>
  foaf:based_near dbp:Palo_Alto%2C_California ;
  foaf:based_near dbp:Packard%27s_garage .
```



[1] Speiser, Harth: “Integrating Linked Data and Services with Linked Data Services”. Proc.8th ESWC (2011)

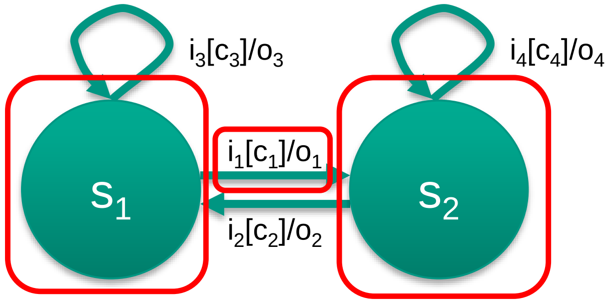
RESTdesc [1]

- 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



A Transition System

```
@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	…
Dynamics model	ASM	LTS	Situation Calculus	Unformalised Implementation	…		
Data model	Graph (RDF)	Nested (JSON, XML)	…				
Data access	RMM2	RMM1	RMM0	push	…		

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

Conclusion

- Correspondence of Read-Write Linked Data to the WoT architecture
- Standards and practices around Read-Write Linked Data
- ASM4LD and WiLD for specifying behaviour on Read-Write Linked Data
- Demo
- Related work