Functional Composition of Sensor Web APIs

Ruben Verborgh et al.
What makes people SMART?
Collaboration.
What makes sensors SMART?
Collaboration.
Smart collaborations run themselves.

manual sensor composition
intelligence comes from humans

automatic sensor composition
intelligence comes from the system
My research is about intelligent Web APIs.

Ruben Verborgh, PhD Student
Ghent University / iMinds

My dream: intelligent agents browsing the Web for us

Bridging between people and things bringing things to the Web
Functional composition of sensor Web APIs

- Capturing functionality
- Creating compositions
- Evaluating feasibility
Functional composition of sensor Web APIs

- Capturing functionality
- Creating compositions
- Evaluating feasibility
Why describe functionality?
Only functionality tells the whole story.
How to capture the functionality of these sensors?

**GPS device**
- determines current location

**temperature sensor**
- reads current temperature

**pressure sensor**
- reads current air pressure
We model sensors as Web APIs.

Web APIs = HTTP APIs = REST APIs
as opposed to “Big” Web services
resource-oriented

Sensor measurements are resources
location
temperature
air pressure
We capture the functionality of Web APIs with RESTdesc.

RESTdesc describes the functionality of hypermedia-driven APIs with Notation3 (an RDF superset).
RESTdesc can describe a temperature sensor.

```json
{
    ?sensor a ex:TemperatureSensor;
    ex:location ?location.
}
=>
{
    _:request http:methodName "GET";
    http:requestURI ?sensor;
    http:resp [ http:body ?temperature ].

    ?location ex:hasTemperature ?temperature.
}
```
RESTdesc can describe a temperature sensor.

```yaml
{  
  ?sensor a ex:TemperatureSensor; 
  ex:location ?location. 
}
=>
{
  _:request http:methodName "GET";
  http:requestURI ?sensor;  
  http:resp [ http:body ?temperature ].  
  
  ?location ex:hasTemperature ?temperature.
}.
```
RESTdesc can describe a temperature sensor.

```json
{
  ?sensor a ex:TemperatureSensor;
  ex:location ?location.
}
=>
{
  _:request http:methodName "GET";
  http:requestURI ?sensor;
  http:resp [ http:body ?temperature ].

  ?location ex:hasTemperature ?temperature.
}.
```
RESTdescription can describe a temperature sensor.

```json
{
  ?sensor a ex:TemperatureSensor;
  ex:location ?location.
}

=>
{
  _:request http:methodName "GET";
  http:requestURI ?sensor;
  http:resp [ http:body ?temperature ].

  ?location ex:hasTemperature ?temperature.
}
```
RESTdesc provides simple functional descriptions.

Hypermedia-driven
no inputs or outputs, just links

Capture functionality
relationship between resources

Rule-based
logic implication
Functional composition of sensor Web APIs

- Capturing functionality
- Creating compositions
- Evaluating feasibility
Why compose Web APIs?

To answer a query by combining multiple APIs.

Queries can be complex,

APIs not necessarily known in advance.
Reserve a table in a nearby restaurant. If weather allows, I want to sit outside.
What APIs do we have at our disposition?

Suppose we have many APIs, including:

- Location API
- Temperature API
- Pressure API
- Restaurant API
What would a candidate solution look like?

“Reserve a table in a nearby restaurant. If weather allows, I want to sit outside.”

Location API
Temperature API
Pressure API
Restaurant API
RESTdesc descriptions enable automated composition.

RESTdesc descriptions are rules.
Rules support inference.
Rules can be chained.
Does this composition satisfy the goal?

\{\textcolor{red}{\text{\Large \textbf{\textcolor{red}{\{}}}}} \quad \textcolor{green}{\text{\Large \textbf{\textcolor{green}{\{}}}}}

\textcolor{green}{\Rightarrow}

\{\textcolor{green}{\text{\Large \textbf{\textcolor{green}{\{}}}}} \quad \text{Since descriptions are rules, success is defined as entailment.}
If success is entailment, the proof is the composition.

\{
\text{success}\}
\Rightarrow
\{
\text{entailment}\}

The proof of the entailment contains the used rules,

\Rightarrow
\{
\text{API requests}\}

thus the needed API requests.
A semantic Web reasoner can create the composition.

“Reserve a table in a nearby restaurant. If weather allows, I want to sit outside.”
RESTdesc composing behaves as a good Semantic Web citizen.

works with generic semantic reasoners
RESTdesc descriptions are rules
no specific knowledge needed

ontologies fill gaps

domain-independent
Functional composition of sensor Web APIs

- Capturing functionality
- Creating compositions
- Evaluating feasibility
Does it scale?
Reasoner-based methods are often thought of as slow.

partly perception
limit reasoner experience

partly the method
translation into reasoner domain
We tested composition chains for performance.

$n$ APIs with 1/2/3 dependencies to test chaining speed

32 APIs in presence of $n$ dummies to test discrimination speed
RESTdesc composition is so fast that it works on Web scale.

chain of 512 APIs with 1 dependency in 0.4 seconds

chain of 512 APIs with 3 dependencies in 1.6 seconds

chain of 32 APIs with 512 dummies in 0.01 seconds
Efficient implementations on Arduino boards.

simple sensors as Web APIs

described with RESTdesc

central composer
RESTdesc is an automated solution for sensor Web API composition.

- Describe REST Web APIs
- Use Semantic Web logic
- Compose with generic reasoners
RESTdesc automatically composes sensor Web APIs.

Curious? Let’s get in touch!

Visit restdesc.org and
connect to @RubenVerborgh on Twitter.