

Logic Programming in Space-Time: The Case of Situatedness in LPaaS

Roberta Calegari¹ Giovanni Ciatto¹ Stefano Mariani²
Enrico Denti¹ Andrea Omicini¹

{roberta.calegari, giovanni.ciatto, enrico.denti,
andrea.omicini}@unibo.it
stefano.mariani@unimore.it

¹Dipartimento di Informatica, Scienza e Ingegneria—Università di Bologna

²Dipartimento di Scienze e Metodi dell'Ingegneria, Università di Modena e Reggio Emilia

Talk @ *19th Workshop From Objects to Agents (WOA 2018)*
Palermo, June 29, 2018

- 1 Motivations & Context
- 2 Logic Programming as a Service
 - Vision & Architecture
- 3 Spatio-temporal awareness in LPaaS
 - Spatio-temporal awareness in LPaaS
 - Interface & API
- 4 Example
- 5 Conclusion & Further Work



Next in Line...

- 1 Motivations & Context
- 2 Logic Programming as a Service
- 3 Spatio-temporal awareness in LPaaS
- 4 Example
- 5 Conclusion & Further Work



Motivation & Context

Need for **intelligence** for the IoT:

- Not only **smart**, inter-connected things
 - Internet of **Intelligent** Things (IoT) [Arsénio et al., 2014]
- Things need to be:
 - Interoperable** — interact with each other with no prior knowledge
 - Context-aware** — observe the environment and react accordingly
 - Goal-oriented** — understand us and our goals
 - Understandable** — explain their decision and actions
- Within **open** environments and in a **distributed** fashion
 - **spatial** locality does matter
 - **temporal** reference does matter, too
 - ! reasoning capabilities still missing in nowadays Things



Motivation & Context

Need for **intelligence** for the IoT:

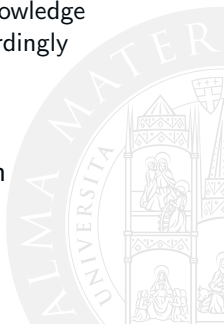
- Not only **smart**, inter-connected things
 - Internet of **Intelligent** Things (IoT) [Arsénio et al., 2014]
- Things need to be:
 - Interoperable** — interact with each other with no prior knowledge
 - Context-aware** — observe the environment and react accordingly
 - Goal-oriented** — understand us and our goals
 - Understandable** — explain their decision and actions
- Within **open** environments and in a **distributed** fashion
 - **spatial** locality does matter
 - **temporal** reference does matter, too
 - ! reasoning capabilities still missing in nowadays Things



Motivation & Context

Need for **intelligence** for the IoT:

- Not only **smart**, inter-connected things
 - Internet of **Intelligent** Things (IoT) [Arsénio et al., 2014]
- Things need to be:
 - Interoperable** — interact with each other with no prior knowledge
 - Context-aware** — observe the environment and react accordingly
 - Goal-oriented** — understand us and our goals
 - Understandable** — explain their decision and actions
- Within **open** environments and in a **distributed** fashion
 - **spatial** locality does matter
 - **temporal** reference does matter, too
 - ! reasoning capabilities still missing in nowadays Things



Motivation & Context

Logic Programming as a Service (LPaaS): [Calegari et al., 2017]

- our proposal endowing IoT with logic programming
 - reasoning capabilities for IoT devices, provided as-a-Service
 - to possibly heterogeneous clients
 - interested in reasoning results
 - only making sense somewhere and for a while
 - LP re-interpreted within the *service perspective*:
 - Encapsulation of theories accesses and updates
 - Stateless interface for stateful interaction
 - Acceptance of locality of the logic service
- ! APIs letting clients focus on specific regions & time intervals
- ✓ proposed in this work



Motivation & Context

Logic Programming as a Service (LPaaS): [Calegari et al., 2017]

- our proposal endowing IoT with logic programming
 - reasoning capabilities for IoT devices, provided **as-a-Service**
 - to possibly **heterogeneous** clients
 - interested in reasoning results
 - only making sense **somewhere** and **for a while**
 - LP re-interpreted within the *service perspective*:
 - **Encapsulation** of theories accesses and updates
 - **Stateless interface** for stateful interaction
 - Acceptance of **locality** of the logic service
- ! APIs letting clients focus on specific **regions** & time **intervals**
- ✓ proposed in this work



Motivation & Context

Logic Programming as a Service (LPaaS): [Calegari et al., 2017]

- our proposal endowing IoT with logic programming
- reasoning capabilities for IoT devices, provided **as-a-Service**
 - to possibly **heterogeneous** clients
 - interested in reasoning results
 - only making sense **somewhere** and **for a while**
- LP re-interpreted within the *service perspective*:
 - **Encapsulation** of theories accesses and updates
 - **Stateless interface** for stateful interaction
 - Acceptance of **locality** of the logic service

! APIs letting clients focus on specific **regions** & time **intervals**

✓ proposed in this work



Motivation & Context

Logic Programming as a Service (LPaaS): [Calegari et al., 2017]

- our proposal endowing IoT with logic programming
- reasoning capabilities for IoT devices, provided **as-a-Service**
 - to possibly **heterogeneous** clients
 - interested in reasoning results
 - only making sense **somewhere** and **for a while**
- LP re-interpreted within the *service perspective*:
 - **Encapsulation** of theories accesses and updates
 - **Stateless interface** for stateful interaction
 - Acceptance of **locality** of the logic service
- ! APIs letting clients focus on specific **regions** & time **intervals**
 - ✓ proposed in this work



Next in Line...

- 1 Motivations & Context
- 2 Logic Programming as a Service**
- 3 Spatio-temporal awareness in LPaaS
- 4 Example
- 5 Conclusion & Further Work



LPaaS Overview

- Several situated logic services may participate to the system
 - possibly **moving**, possibly leaving/entering
- They provide an abstract view of a LP **inference engine**
 - implementing the SLD resolution [Robinson, 1965]
 - through a technology-agnostic, network-enabled interface
- Each service **encapsulates** its logic theories (or Knowledge Base, KB)—which therefore represent truth from the service **p.o.v.**
 - sensors are assumed to be providing fresh & local information [Calegari et al., 2018a]
- Two roles for users:
 - **client** — may query a particular version of the service requesting goals to be solved
 - **configurator** — may arbitrarily alter the KB
- Implementation exists wrapping **tuProlog** [Denti et al., 2013] within a **RESTful** Web-Service, internally organized through **micro-services**



LPaaS Overview

- Several situated logic services may participate to the system
 - possibly moving, possibly leaving/entering
- They provide an abstract view of a LP **inference engine**
 - implementing the SLD resolution [Robinson, 1965]
 - through a technology-agnostic, network-enabled **interface**
- Each service **encapsulates** its logic theories (or Knowledge Base, KB)—which therefore represent truth from the service **p.o.v.**
 - sensors are assumed to be providing fresh & local information [Calegari et al., 2018a]
- Two roles for users:
 - **client** — may query a particular version of the service requesting goals to be solved
 - **configurator** — may arbitrarily alter the KB
- Implementation exists wrapping **tuProlog** [Denti et al., 2013] within a **RESTful** Web-Service, internally organized through **micro-services**



LPaaS Overview

- Several situated logic services may participate to the system
 - possibly moving, possibly leaving/entering
- They provide an abstract view of a LP **inference engine**
 - implementing the SLD resolution [Robinson, 1965]
 - through a technology-agnostic, network-enabled interface
- Each service **encapsulates** its logic theories (or Knowledge Base, KB)—which therefore represent truth from the service **p.o.v.**
 - **sensors** are assumed to be providing fresh & local information [Calegari et al., 2018a]
- Two roles for users:
 - client — may query a particular version of the service requesting goals to be solved
 - configurator — may arbitrarily alter the KB
- Implementation exists wrapping **tuProlog** [Denti et al., 2013] within a **RESTful** Web-Service, internally organized through **micro-services**



LPaaS Overview

- Several situated logic services may participate to the system
 - possibly moving, possibly leaving/entering
- They provide an abstract view of a LP inference engine
 - implementing the SLD resolution [Robinson, 1965]
 - through a technology-agnostic, network-enabled interface
- Each service encapsulates its logic theories (or Knowledge Base, KB)—which therefore represent truth from the service p.o.v.
 - sensors are assumed to be providing fresh & local information [Calegari et al., 2018a]
- Two roles for users:
 - client** — may query a particular version of the service requesting **goals** to be **solved**
 - configurator** — may arbitrarily alter the KB
- Implementation exists wrapping **tuProlog** [Denti et al., 2013] within a RESTful Web-Service, internally organized through **micro-services**

LPaaS Overview

- Several situated logic services may participate to the system
 - possibly moving, possibly leaving/entering
- They provide an abstract view of a LP inference engine
 - implementing the SLD resolution [Robinson, 1965]
 - through a technology-agnostic, network-enabled interface
- Each service **encapsulates** its logic theories (or Knowledge Base, KB)—which therefore represent truth from the service **p.o.v.**
 - sensors are assumed to be providing fresh & local information [Calegari et al., 2018a]
- Two roles for users:
 - client — may query a particular version of the service requesting goals to be solved
 - configurator — may arbitrarily alter the KB
- Implementation exists wrapping **tuProlog** [Denti et al., 2013] within a **RESTful** Web-Service, internally organized through **micro-services**



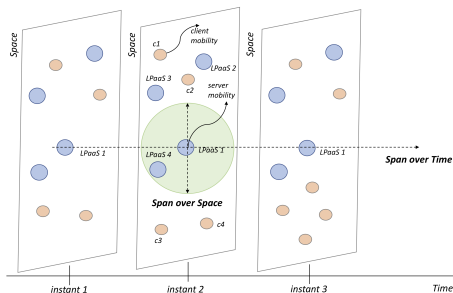
Next in Line...

- 1 Motivations & Context
- 2 Logic Programming as a Service
- 3 Spatio-temporal awareness in LPaaS**
- 4 Example
- 5 Conclusion & Further Work



Situatedness in LPaaS

Resolution process takes into account the service **spatio-temporal** context



- a goal is demonstrated against a logic theory considered true within a (possibly open) **interval of time**, and a **region of space**
- resolution process could **span** over time and space

The temporal/spatial dimensions

Time in LPaaS

- *time-dependant computations*, e.g. **periodic**
- **temporal validity** of logic theories, individual facts and clauses
- requests may restrict the **temporal scope** of the expected solutions

Space in LPaaS

- space-dependant computations
- **spatial validity** of logic theories, individual facts and clauses
- requests may restrict the **spatial scope** of the expected solutions



High-level, situation-aware **APIs** must be provided to clients

The temporal/spatial dimensions

Time in LPaaS

- *time-dependant computations*, e.g. **periodic**
- **temporal validity** of logic theories, individual facts and clauses
- requests may restrict the **temporal scope** of the expected solutions

Space in LPaaS

- **space-dependant** computations
- **spatial validity** of logic theories, individual facts and clauses
- requests may restrict the **spatial scope** of the expected solutions



High-level, situation-aware **APIs** must be provided to clients

The temporal/spatial dimensions

Time in LPaaS

- *time-dependant computations*, e.g. **periodic**
- **temporal validity** of logic theories, individual facts and clauses
- requests may restrict the **temporal scope** of the expected solutions

Space in LPaaS

- **space-dependant** computations
- **spatial validity** of logic theories, individual facts and clauses
- requests may restrict the **spatial scope** of the expected solutions



High-level, situation-aware **APIs** must be provided to clients

API – Temporal dimension

Clients choose **when** they want a solution

Timely solutions — `solve(?Goal, ..., -Results)` predicates with a `within(Time)` argument constraining resolution duration

Periodic solutions stream — `solve` predicates with a `every(Time)` arg.

Time aware resolution process

- Axiom in theories decorated with `time interval`, defining their `validity`
- Clients' queries could specify a(n open?) `TimeInterval` of interest
- Only axioms valid at `Instant` $\in$ `TimeInterval` affect resolution
? reason about the future?



API – Temporal dimension

Clients choose **when** they want a solution

Timely solutions — `solve(?Goal, ..., -Results)` predicates with a `within(Time)` argument constraining resolution duration

Periodic solutions stream — solve predicates with a `every(Time)` arg.

Time aware resolution process

- Axiom in theories decorated with `time interval`, defining their `validity`
- Clients' queries could specify a(n open?) `TimeInterval` of interest
- Only axioms valid at `Instant` $\in$ `TimeInterval` affect resolution
? reason about the future?

Spatial dimension

Clients specify **where** solutions should make sense

Local solutions — `solveNeighborhood(?Goal, -Results, ?Region)`
predicate provide a solution from a **neighborhood** of logic services

- implicit **interaction** between service instances may occur

Solutions stream across space — e.g. by **widening/narrowing** the region considered at consecutive resolution cycles

Space aware resolution process

- **Distribution** is a premise *per se* to spatial situatedness
- Logic services are (assumed to be) **aware** of their **positions**
 - w.r.t. some custom notion of **locality** (GPS lat/long, network hops, etc)
- They keep track of what is true in their **surroundings**

Spatial dimension

Clients specify **where** solutions should make sense

Local solutions — `solveNeighborhood(?Goal, -Results, ?Region)`
predicate provide a solution from a **neighborhood** of logic services

- implicit **interaction** between service instances may occur

Solutions stream across space — e.g. by **widening/narrowing** the region considered at consecutive resolution cycles

Space aware resolution process

- **Distribution** is a premise *per se* to spatial situatedness
- Logic services are (assumed to be) **aware** of their **positions**
 - w.r.t. some custom notion of **locality** (GPS lat/long, network hops, etc)
- They keep track of what is true in their **surroundings**

Next in Line...

- 1 Motivations & Context
- 2 Logic Programming as a Service
- 3 Spatio-temporal awareness in LPaaS
- 4 Example**
- 5 Conclusion & Further Work



Testbed Scenario

User looking for a parking in proximity of a metro station in London

- distance = num. of **stations** between LPaaS server and parking place



Testbed Scenario

```
?- solve( station_with_parking( $X$ ),  $S$  )
```



no.



Testbed Scenario

```
?- solveAllNeighborhood( ( station_with_parking(X),
    free_places(X, N) ), S, region( _, distance(1, Line) ) )
```



yes. $S = (X/\text{'LondonBridge'}, N/5, \text{Line}/\text{black})$



Testbed Scenario

```
?- solveAllNeighborhood( ( station_with_parking(X),
    free_places(X, N) ), S, region( _, distance(1, brown) ) )
```



no.



Testbed Scenario

```
?- solveAllNeighborhood( ( station_with_parking(X),
    free_places(X, N) ), S, region( _, distance(2, Line) ) )
```



```
yes. S = [ (X/'LondonBridge', N/5, Line/black),
    (X/'Charing Cross', N/1, Line/brown),
    (X/'Canada Water', N/2, Line/[black, gray]) ]
```

Next in Line...

- 1 Motivations & Context
- 2 Logic Programming as a Service
- 3 Spatio-temporal awareness in LPaaS
- 4 Example
- 5 Conclusion & Further Work



Conclusion

- Pervasive and situated **IoT** scenarios may take advantage of **context-aware** intelligence
- LPaaS aims at satisfying such a challenging requirement
 - exposing logic reasoning capabilities **as-a-Service**
 - to **heterogeneous** clients moving around
- It supports **spatio-temporal awareness** while reasoning, thus supporting the distribution of **situated intelligence**
 - through suitably-expressive set of spatio-temporal **primitives**



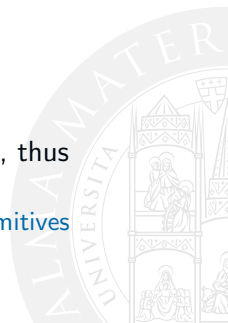
Conclusion

- Pervasive and situated **IoT** scenarios may take advantage of **context-aware** intelligence
- LPaaS aims at satisfying such a challenging requirement
 - exposing logic reasoning capabilities **as-a-Service**
 - to **heterogeneous** clients moving around
- It supports **spatio-temporal awareness** while reasoning, thus supporting the distribution of **situated intelligence**
 - through suitably-expressive set of spatio-temporal **primitives**



Conclusion

- Pervasive and situated **IoT** scenarios may take advantage of **context-aware** intelligence
- LPaaS aims at satisfying such a challenging requirement
 - exposing logic reasoning capabilities **as-a-Service**
 - to **heterogeneous** clients moving around
- It supports **spatio-temporal awareness** while reasoning, thus supporting the distribution of **situated intelligence**
 - through suitably-expressive set of spatio-temporal **primitives**



Further Work

- Designing & developing a specialised logic-based **middleware**
- Tests in pervasive deployment scenarios are required, mainly in the IoT landscape
e.g. testing directly LPaaS over BLE or Wifi-Direct connections
- Integrating LPaaS with **Labelled LP** [Calegari et al., 2018b] for domain-specific logic-based computation



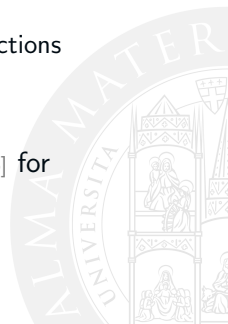
Further Work

- Designing & developing a specialised logic-based [middleware](#)
- Tests in pervasive deployment scenarios are required, mainly in the IoT landscape
e.g. testing directly LPaaS over BLE or Wifi-Direct connections
- Integrating LPaaS with [Labelled LP](#) [Calegari et al., 2018b] for domain-specific logic-based computation



Further Work

- Designing & developing a specialised logic-based **middleware**
- Tests in pervasive deployment scenarios are required, mainly in the IoT landscape
e.g. testing directly LPaaS over BLE or Wifi-Direct connections
- Integrating LPaaS with **Labelled LP** [Calegari et al., 2018b] for domain-specific logic-based computation



References I



Arsénio, A., Serra, H., Francisco, R., Nabais, F., Andrade, J., and Serrano, E. (2014). *Internet of Intelligent Things: Bringing Artificial Intelligence into Things and Communication Networks*, pages 1–37. Springer Berlin Heidelberg, Berlin, Heidelberg.



Calegari, R., Ciatto, G., Mariani, S., Denti, E., and Omicini, A. (2018a). *Micro-intelligence for the iot: Se challenges and practice in lpaas*. In *2018 IEEE International Conference on Cloud Engineering (IC2E 208)*, pages 292–297. IEEE Computer Society.



Calegari, R., Denti, E., Dovier, A., and Omicini, A. (2018b). *Extending logic programming with labelled variables: Model and semantics*. *Fundamenta Informaticae*, 161:53–74. Special Issue CILC 2016.



Calegari, R., Denti, E., Mariani, S., and Omicini, A. (2017). *Logic Programming as a Service (LPaaS): Intelligence for the IoT*. In Fortino, G., Zhou, M., Lukszo, Z., Vasilakos, A. V., Basile, F., Palau, C., Liotta, A., Fanti, M. P., Guerrieri, A., and Vinci, A., editors, *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC 2017)*, pages 72–77. IEEE.

References II

- 
- Denti, E., Omicini, A., and Calegari, R. (2013).
[tuProlog: Making Prolog ubiquitous.](#)
ALP Newsletter.
- 
- Robinson, J. A. (1965).
[A machine-oriented logic based on the resolution principle.](#)
Journal of the ACM, 12(1):23–41.



Logic Programming in Space-Time: The Case of Situatedness in LPaaS

Roberta Calegari¹ Giovanni Ciatto¹ Stefano Mariani²
Enrico Denti¹ Andrea Omicini¹

{roberta.calegari, giovanni.ciatto, enrico.denti,
andrea.omicini}@unibo.it
stefano.mariani@unimore.it

¹Dipartimento di Informatica, Scienza e Ingegneria—Università di Bologna

²Dipartimento di Scienze e Metodi dell'Ingegneria, Università di Modena e Reggio Emilia

Talk @ *19th Workshop From Objects to Agents (WOA 2018)*
Palermo, June 29, 2018