

WP4 Activities: A Prototype of LSA-space

Elena Nardini

ALMA MATER STUDIORUM—Università di Bologna

SAPERE Meeting
23/06/2011, St. Andrews, Scotland



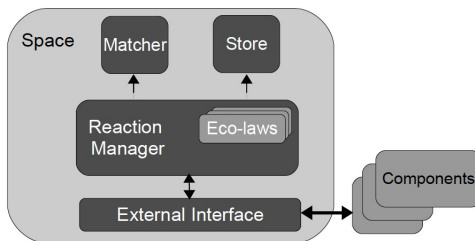
LSA-spaces

General features

- A mobile network (graph) of LSA-spaces
- Links denote direct interaction abilities
- An LSA-space has a store of LSAs and an eco-law engine
- Eco-laws apply in specific space-time points of the SAPERE fabric
 - I.e., an eco-law applies in one LSA-space (can affect neighbourhood)
 - More refined view: an eco-law can create LSAs into one neighbour
- (LSAs can link to LSAs in a neighbouring space)



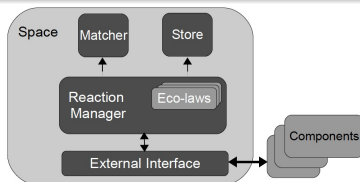
LSA-space Model



LSA-space Model

Space modules: external interface

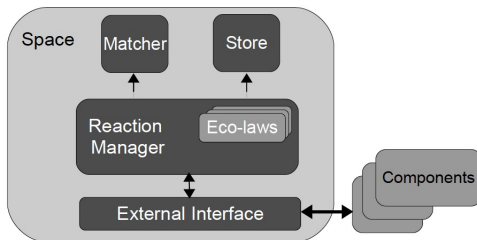
- Interface by which components can interact with a SAPERE space
- Provides the operations:
 - **inject**: takes a property descriptions, yields LSA-id
 - **update**: takes LSA-id and some new property descriptions, updates the LSA
 - **remove**: takes LSA-id, drops whole LSA
 - **observe**: takes LSA-id and property names, yields LSA-id (also by notification)
 - read
 - notify “one shot”
 - continuous notify
 - stop notification



LSA-space Model

Space modules: LSA store

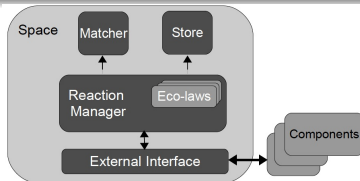
- Passive element similar to a tuple space
- Stores LSAs that are local to the SAPERE node
- Provides an interface to **add**, **consume**, **read** and **modify** LSAs in the space
- Responsible for the LSA identification within the node
- Responsible for the implementation of any possibly sophisticated indexing algorithm and data structure for quickly retrieving and filtering LSAs



LSA-space Model

Space modules: reaction manager

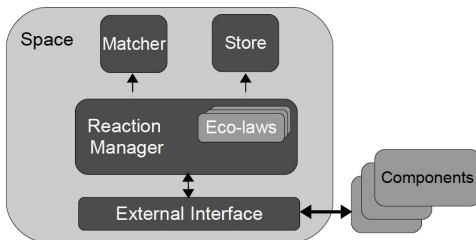
- Handles the events occurring within the space:
 - External operation upon an LSA (except for the **observe**)
 - Occurs instantaneously
 - As soon as possible, it is processed by an interaction with the space
 - Activation of a specific eco-law to be executed (it encapsulates the **eco-law engine**)
 - Occurs as soon as the time associate to the eco-law is lapsed
 - As soon as possible, it is processed by generating the eco-law results (products): interactions with the space or firing of some LSAs to neighbourhood
- Exploits a simple **scheduling engine** maintaining a list of scheduled events sorted by their occurrence time
 - Interactions with the space can change the space: this requires to update the event list and to notify LSA observers



LSA-space Model

Space modules: matcher

- Exploited by the reaction manager when it processes eco-laws
 - It checks whether an eco-law can apply to a candidate set of LSAs
 - It computes what would be the result of applying the eco-law to that set of LSAs
 - LSA checking and result computing may involve some syntactical/semantic matching and functions ($\{x:x \text{ f } e\}$)



LSA-space Implementation

Some facts

- Developed by Master students in UNIBO, tutored by Mirko & Elena
- A Java-based implementation supporting WP1 model above
- Available through the SVN and documented in [TR.WP4.2011.02](#)
- Will be developed by UNIBO and UNIMORE



LSA-space Implementation

Current state

- Supports a subset of eco-law language (e.g. functions in the expression $\{x:x \text{ f } e\}$ are not supported)
- LSA store does not implement any smart algorithm to access to LSAs
- The diffusion mechanism is enacted by the *diffuser* that is encapsulated in the space manager
- Supports a subset of interaction primitives: *inject*, *remove*, *update* and *observe* with *notify* “one shot”
- Connection between individuals and nodes and among nodes is managed by the space. No support of mobility
- The matching mechanism exploited to select LSAs from eco-laws is synthactict (some experiments with *fuzzyDL*)



LSA-space Implementation

Tools

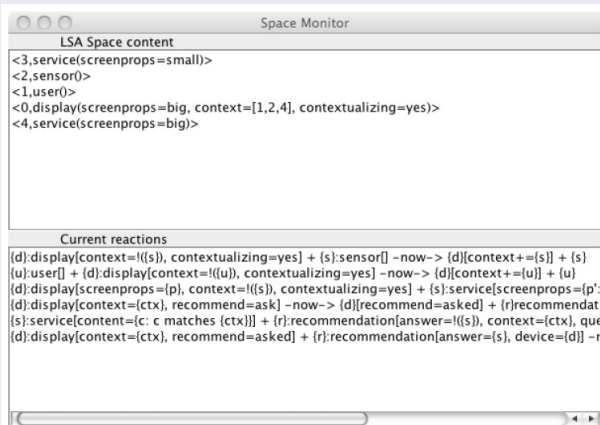
Sapere Agent GUI 10001

LSA service(content(1))		INJECT
ID	FILTER	OBSERVE
ID		REMOVE
ID	CONTENT	UPDATE
LSA	FILTER	INJECT-OBSERVE

```
request = operation[type=Inject,lsa=<null,display(screenprops=big, contextualizing=yes, recc
true - operation[type=Inject,lsa=<0,display(screenprops=big, contextualizing=yes, recommen
request = operation[type=Inject,lsa=<null,sensor()>]
true - operation[type=Inject,lsa=<null,sensor()>]
request = operation[type=Inject,lsa=<null,service(content=1)>]
true - operation[type=Inject,lsa=<null,service(content=1)>]
```

LSA-space Implementation

Tools



Current activities

Should be over by M12

- Full support of the eco-law language described in TR.WP1.2011.03
- Connection and diffusion supported out of the space
- Some support for wifi/bluetooth connections
- Some support for Android

Other interesting activities

- Described in TR.WP1.2011.04
- RDF-based implementation: LSA as OWL descriptions and eco-laws as a set of SAPRQL and SPARQL update instructions
- Semantic matching: the **matcher** exploits a reasoner (like Pellet) [Sirin et al., 2007] in both LSAs selection and functions in expressions like $\{x:x \text{ f } e\}$



References I



Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007).
Pellet: A practical owl-dl reasoner.
J. Web Sem., 5(2):51–53.



WP4 Activities: A Prototype of LSA-space

Elena Nardini

ALMA MATER STUDIORUM—Università di Bologna

SAPERE Meeting
23/06/2011, St. Andrews, Scotland

