technische universität
dortmund

Lehrstuhl für Programmiersysteme /
Fakultät für Informatik

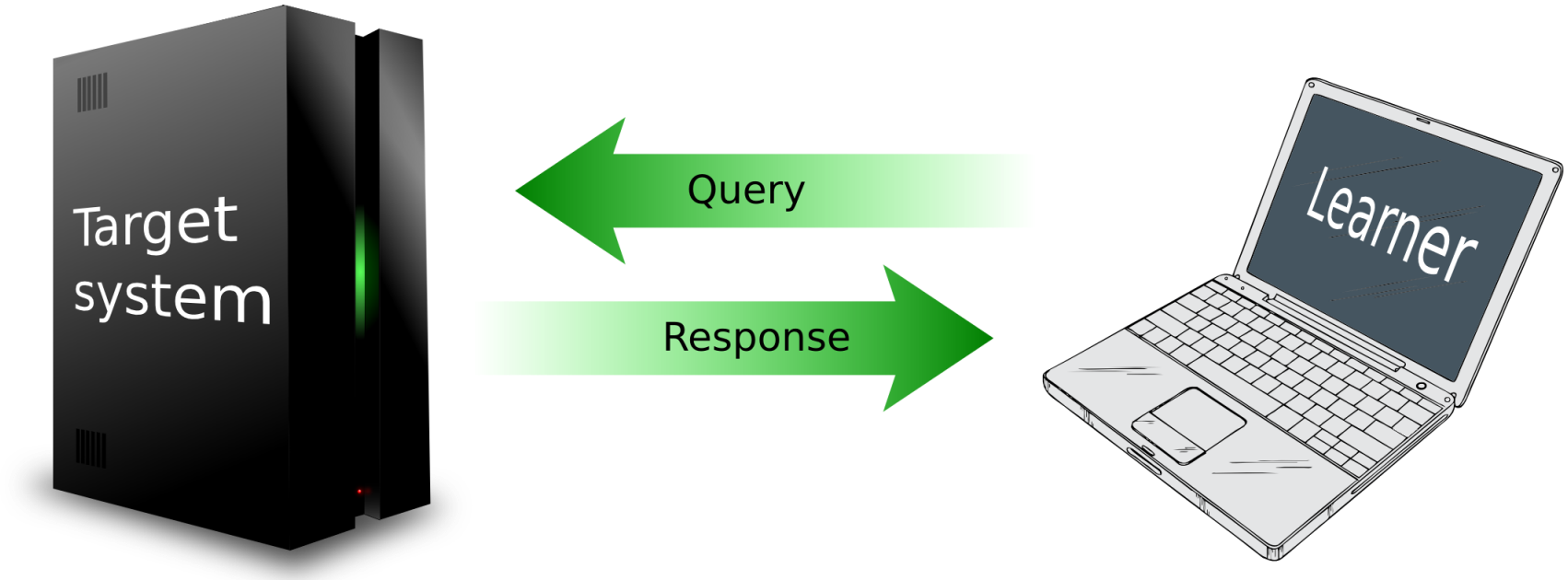
Automated mapper generation and monitoring integration

Maik Merten | ISoLA 2011

Active Automata Learning

- L* Algorithm
  - Originally for finite state acceptors (+ ,- output)
  - Mealy Machines (arbitrary output) better fit for reactive systems
- Two types of queries in active automata learning
  - Membership Queries (MQs): Traces of input actions, hypothesis constructed from response
  - Equivalence Queries (EQs): “Is the hypothesis correct?” – answer usually approximated

Overall Setup



Conveniently omits essential detail…

Test-driver

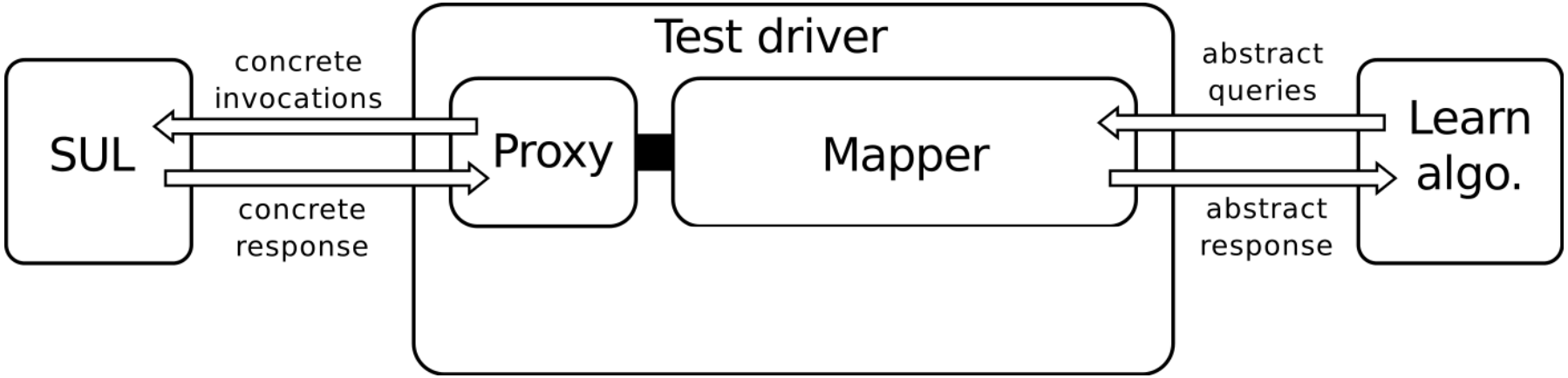
- Test-diver executes queries on system
  - Invokes target system
  - Catches system response
- Mediates between Learner and System Under Learning (SUL)

Mind the gap!

- Abstract learning alphabet
- Concrete system invocations

- Equip test-driver with a mapper that translates between the two

Test-driver mit mapper module

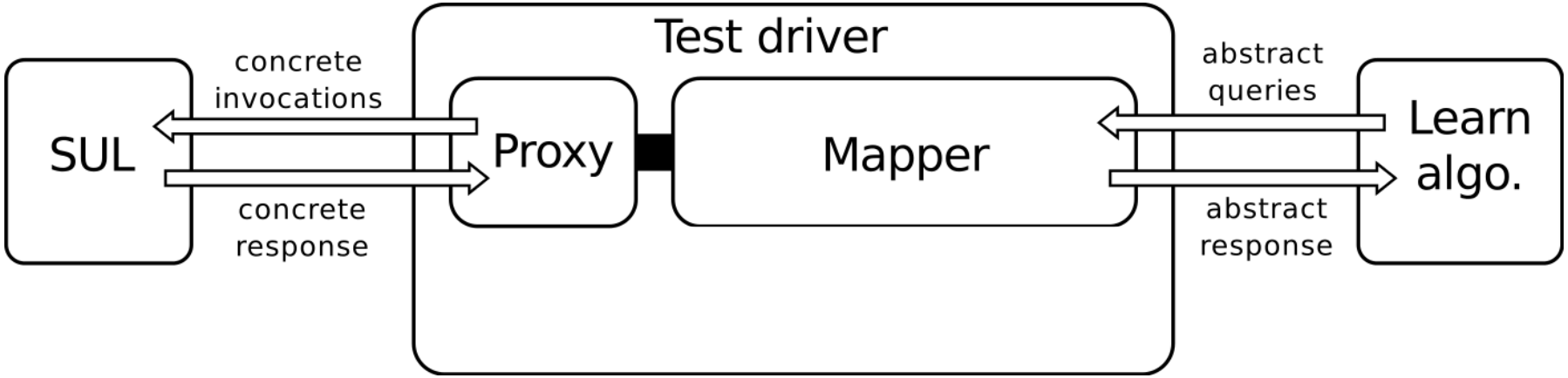


The influence of data values

- Real-life systems often have parameterized actions
  - For example, login-actions need credentials
- Actions may return values needed later on
  - For example, login-action returns session identifier, needed for all subsequent actions

- Thus: Mapper needs facility for data management!

Test-driver mit mapper module



Challenges

- Current practice: Create mapper manually, for each system
  - Account for data-dependencies between actions
  - Expensive, no automated learning of systems without preparation
- System instrumentation has to be provided (“Proxy”)
- Automated solution desirable

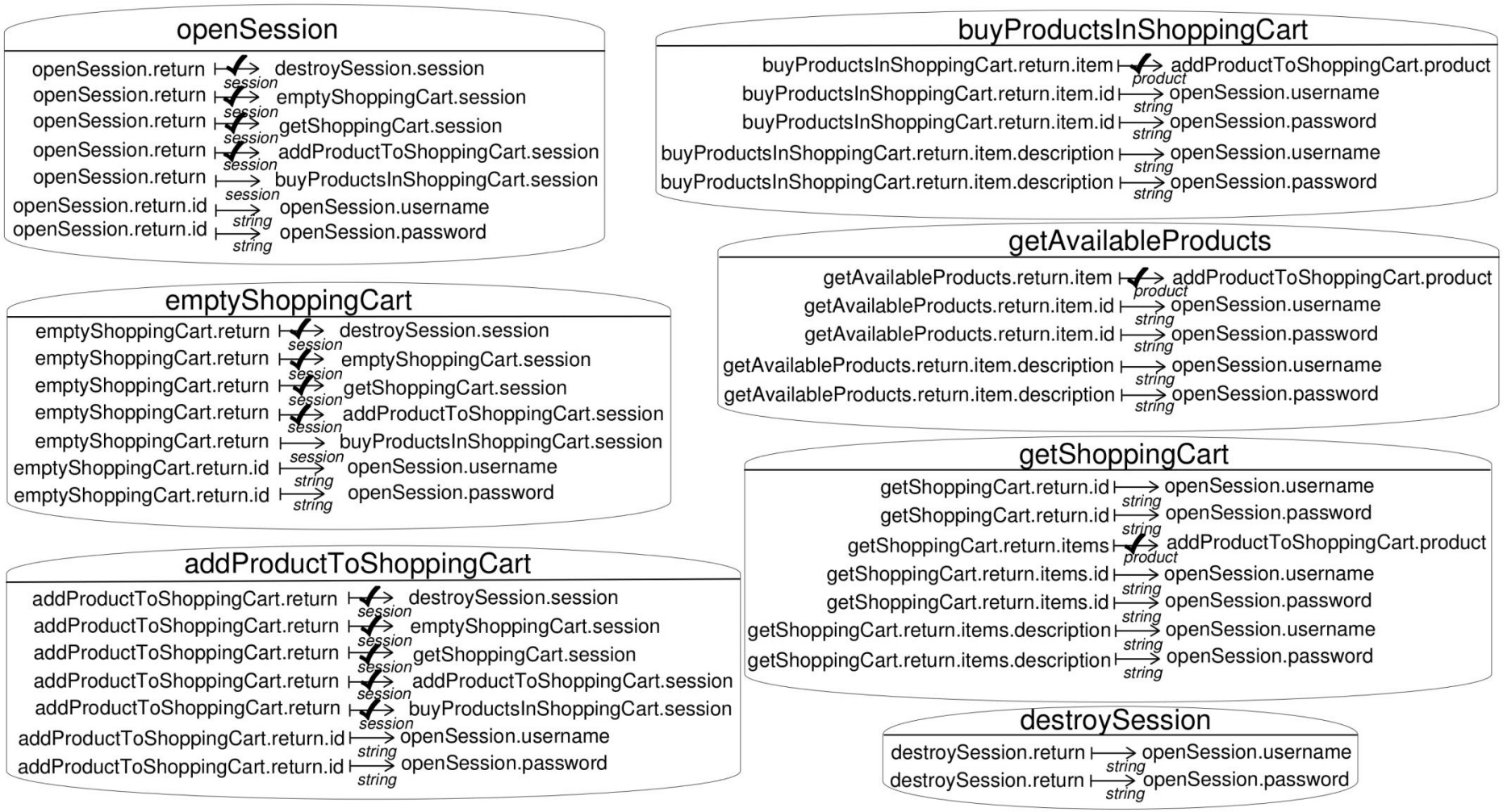
- Our approach: Combine interface analysis of WSDL interfaces to determine data-dependencies, instantiate scripted mapper automatically
  - Joint work with Patrizio Pellicione and Massimo Tivoli of Università dell’Aquila
- Proxy generated with standard tool “wsimport” for WSDL services

Strawberry

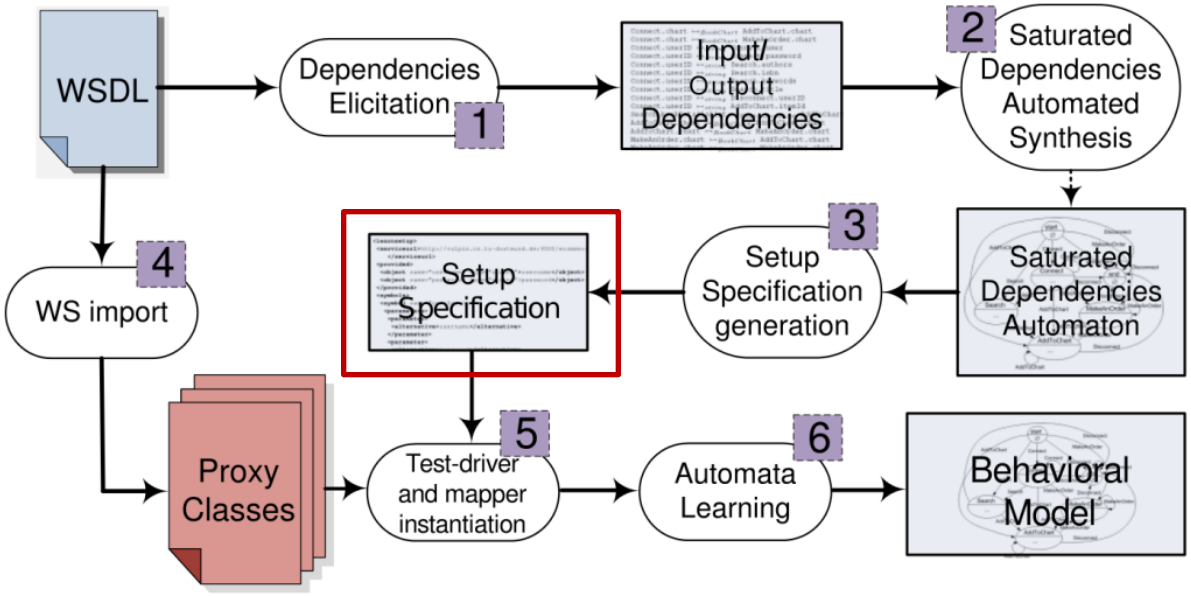
- Strawberry is developed at the university of l'Aquila, Italy
- Analyses WSDL interface descriptions
  - Determines potential data dependencies by type analysis, optionally backed up by tests

Ecommerce service

- WSDL mock-up service modeled after ecommerce applications
  - `openSession(username, password)` `-> session`
  - `destroySession(session)` `-> string`
  - `getAvailableProducts()` `-> productArray`
  - `emptyShoppingCart(session)` `-> session`
  - `getShoppingCart(session)` `-> shoppingCart`
  - `addProductToShoppingCart(session, product)` `-> session`
  - `buyProductsInShoppingCart(session)` `-> productArray`


openSession
openSession.return ⊢✓→ destroySession.session (session)
openSession.return ⊢✓→ emptyShoppingCart.session (session)
openSession.return ⊢✓→ getShoppingCart.session (session)
openSession.return ⊢✓→ addProductToShoppingCart.session (session)
openSession.return ⊢→ buyProductsInShoppingCart.session (session)
openSession.return.id ⊢→ openSession.username (string)
openSession.return.id ⊢→ openSession.password (string)
buyProductsInShoppingCart
buyProductsInShoppingCart.return.item ⊢✓→ addProductToShoppingCart.product (product)
buyProductsInShoppingCart.return.item.id ⊢→ openSession.username (string)
buyProductsInShoppingCart.return.item.id ⊢→ openSession.password (string)
buyProductsInShoppingCart.return.item.description ⊢→ openSession.username (string)
buyProductsInShoppingCart.return.item.description ⊢→ openSession.password (string)
emptyShoppingCart
emptyShoppingCart.return ⊢✓→ destroySession.session (session)
emptyShoppingCart.return ⊢✓→ emptyShoppingCart.session (session)
emptyShoppingCart.return ⊢✓→ getShoppingCart.session (session)
emptyShoppingCart.return ⊢✓→ addProductToShoppingCart.session (session)
emptyShoppingCart.return ⊢→ buyProductsInShoppingCart.session (session)
emptyShoppingCart.return.id ⊢→ openSession.username (string)
emptyShoppingCart.return.id ⊢→ openSession.password (string)
getAvailableProducts
getAvailableProducts.return.item ⊢✓→ addProductToShoppingCart.product (product)
getAvailableProducts.return.item.id ⊢→ openSession.username (string)
getAvailableProducts.return.item.id ⊢→ openSession.password (string)
getAvailableProducts.return.item.description ⊢→ openSession.username (string)
getAvailableProducts.return.item.description ⊢→ openSession.password (string)
addProductToShoppingCart
addProductToShoppingCart.return ⊢✓→ destroySession.session (session)
addProductToShoppingCart.return ⊢✓→ emptyShoppingCart.session (session)
addProductToShoppingCart.return ⊢✓→ getShoppingCart.session (session)
addProductToShoppingCart.return ⊢✓→ addProductToShoppingCart.session (session)
addProductToShoppingCart.return ⊢✓→ buyProductsInShoppingCart.session (session)
addProductToShoppingCart.return.id ⊢→ openSession.username (string)
addProductToShoppingCart.return.id ⊢→ openSession.password (string)
getShoppingCart
getShoppingCart.return.id ⊢→ openSession.username (string)
getShoppingCart.return.id ⊢→ openSession.password (string)
getShoppingCart.return.items ⊢✓→ addProductToShoppingCart.product (product)
getShoppingCart.return.items.id ⊢→ openSession.username (string)
getShoppingCart.return.items.id ⊢→ openSession.password (string)
getShoppingCart.return.items.description ⊢→ openSession.username (string)
getShoppingCart.return.items.description ⊢→ openSession.password (string)
destroySession
destroySession.return ⊢→ openSession.username (string)
destroySession.return ⊢→ openSession.password (string)

Bootstrapping learning



```
1  <learnsetup>
2   <serviceurl>http://vulpis.cs.tu-dortmund.de:9000/ecommerceservice?wsdl
      </serviceurl>
3   <provided>
4    <object name="username" type="string">username</object>
5    <object name="password" type="string">password</object>
6   </provided>
7   <symbols>
8    <symbol name="openSession">
9     <parameters>
10     <parameter>
11      <alternative>username</alternative>
12     </parameter>
13     <parameter>
14      <alternative>password</alternative>
15     </parameter>
16    </parameters>
17    <return>session</return>
18   </symbol>
19   ...
20   <symbol name="getAvailableProducts">
21    <parameters />
22    <return>productArray</return>
23   </symbol>
24   ...
25   <symbol name="addProductToShoppingCart">
26    <parameters>
27     <parameter>
28      <alternative>session</alternative>
29     </parameter>
30     <parameter>
```

```
<symbol name="addProductToShoppingCart">
 <parameters>
  <parameter>
   <alternative>session</alternative>
  </parameter>
  <parameter>
   <alternative selector="elementOf" field="item">productArray
      </alternative>
   <alternative selector="elementOf" field="items">shoppingCart
      </alternative>
  </parameter>
 </parameters>
 <return>session</return>
</symbol>
```

Executed on scripted data value context:

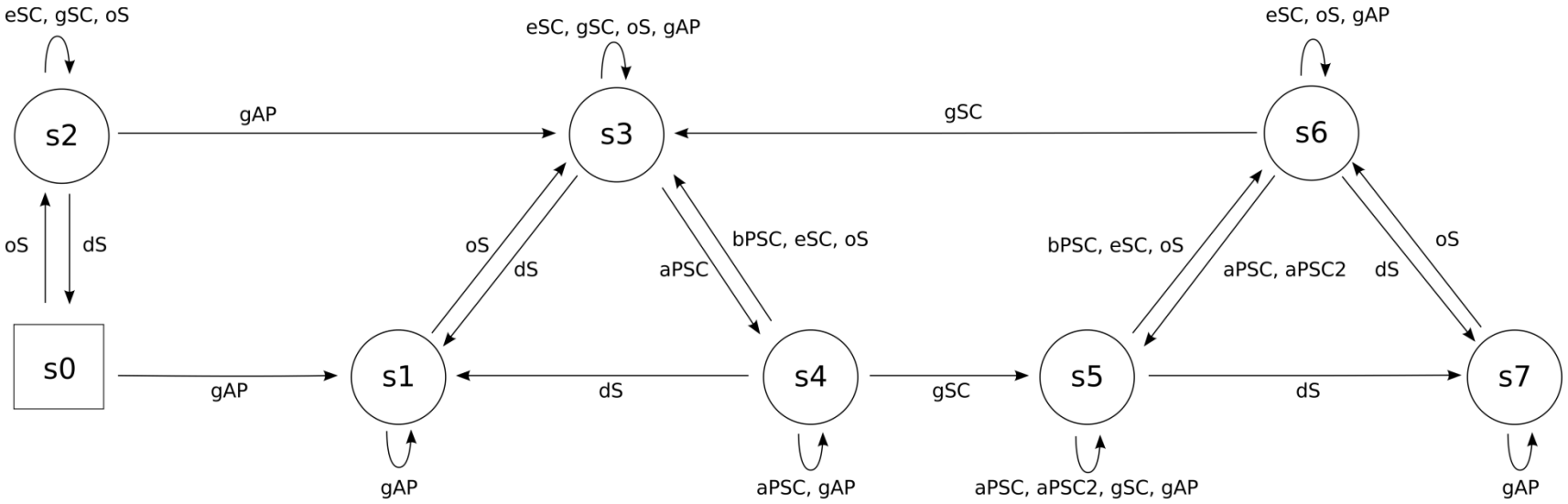
```
elementOf(shoppingCart.getItems())
```

Automated mapper instantiation

- Parse setup specification
  - URI to system
    - Proxy generation via wsimport
  - (parameterized) Alphabet
- Scripted data value context
  - Retrieve data values for parameters
  - System response stored as named variables

Learned model

Behavior captured not deducible from interface alone!



Legend	
os: openSession(username, password): session ds: destroySession(session): string gAP: getAvailableProducts(): productArray eSC: emptyShoppingCart(session): session	gSC: getShoppingCart(session): shoppingCart aPSC: addProductToShoppingCart(session, elementOf(productArray.getItem())): session aPSC2: addProductToShoppingCart(session, elementOf(shoppingCart.getItems()): session bPSC: buyProductsInShoppingCart(session): productArray

Conclusion

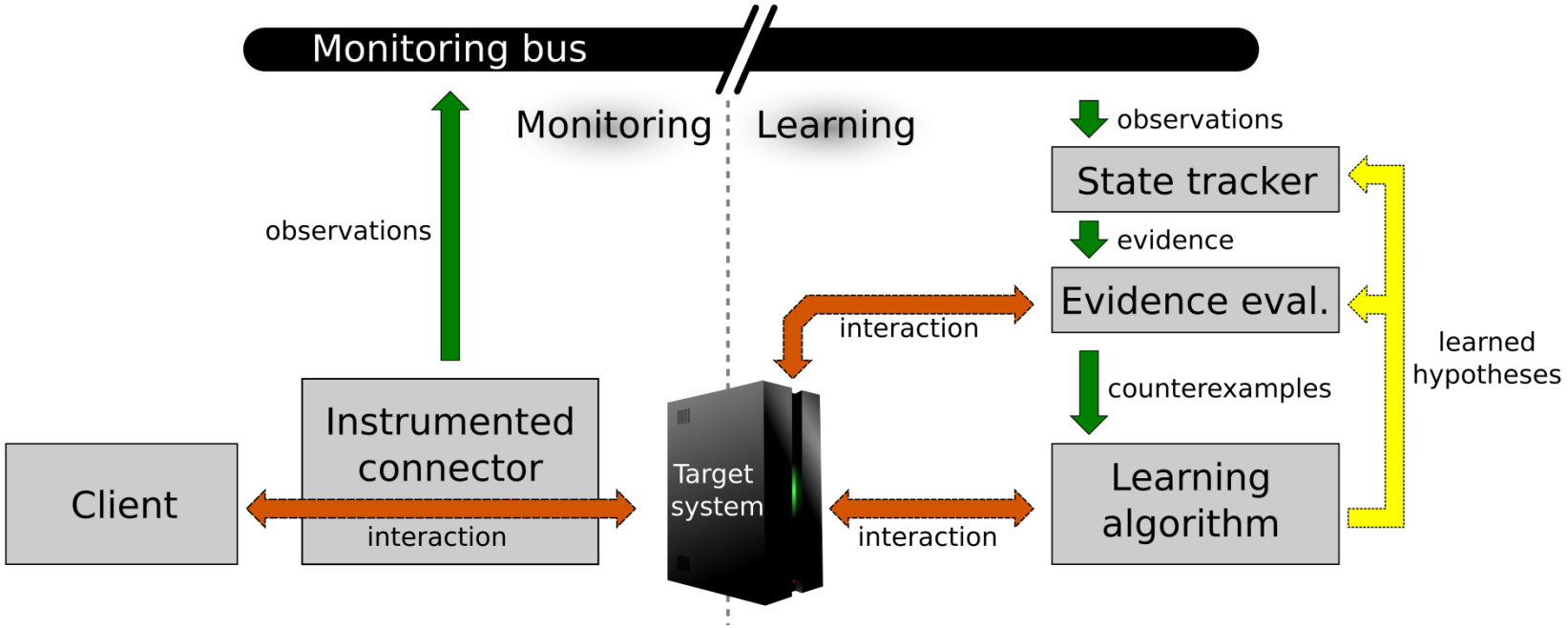
- Automated test-driver instantiation
  - Parameterized actions
  - Alphabet from interface analysis
  - Scriptable data value context for limited data manipulation
- Learns behavioral models

- Future work: Go beyond data-type compatibility

Monitoring

- Joint work with Antonello Calabrò (CNR Pisa, Italy)
- Use CNR Glimpse monitoring framework to gather observations of runtime interaction
- Compare to current learning hypothesis

Monitoring setup



(No) conclusion

- Experimentation just begun, promising initial results

Thank you!