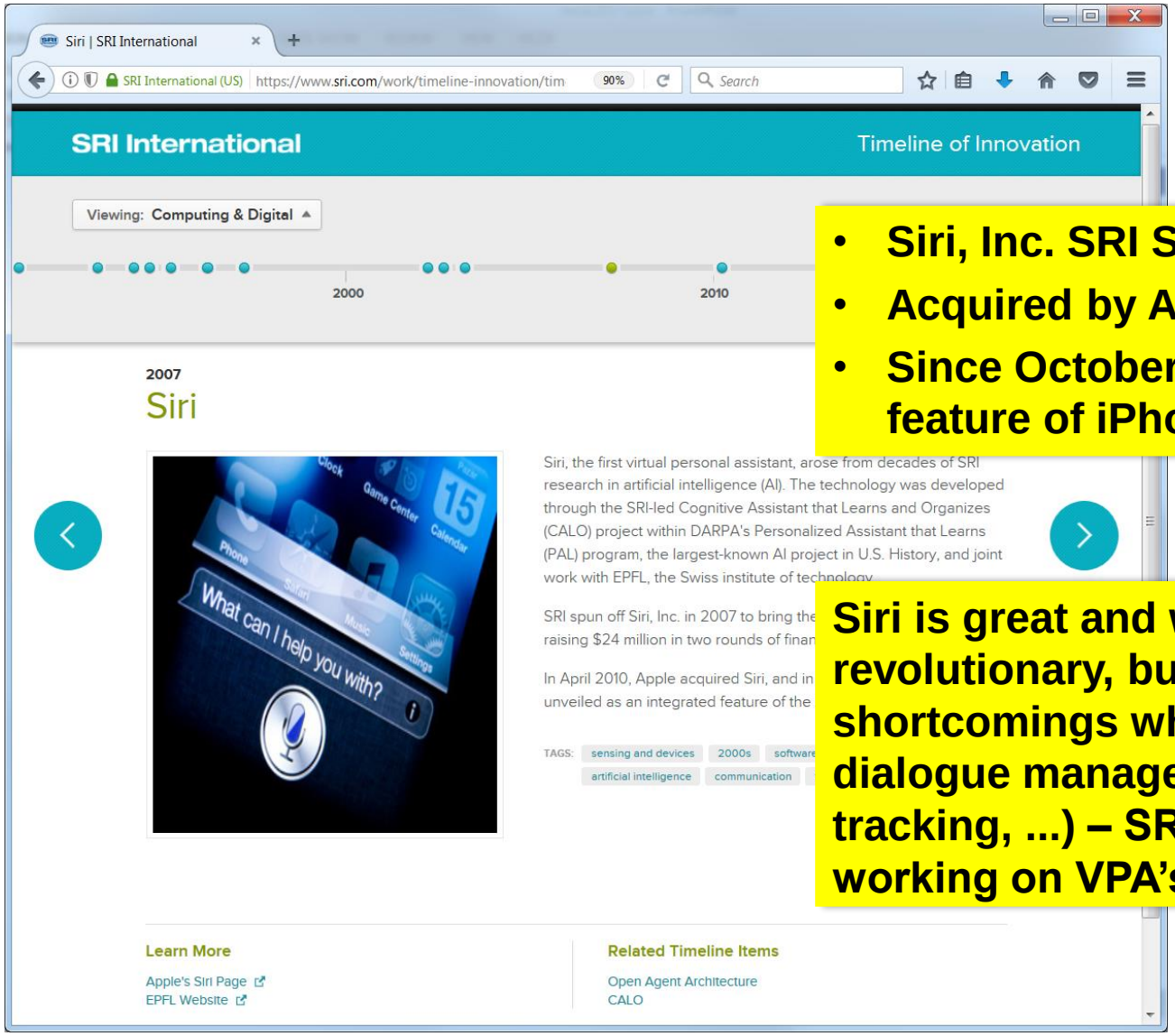
SRI International

OntoVPA -
Building an ontology-based rule engine for dialogue management on SPARQL + OWL

Michael Wessel,
Girish Acharya, James Carpenter and Min Yin

Background – VPAs @ SRI International (Think “Siri”)



- **Siri, Inc. SRI Spinoff 2007**
- **Acquired by Apple in 2010**
- **Since October 2011 integrated feature of iPhone 4S**

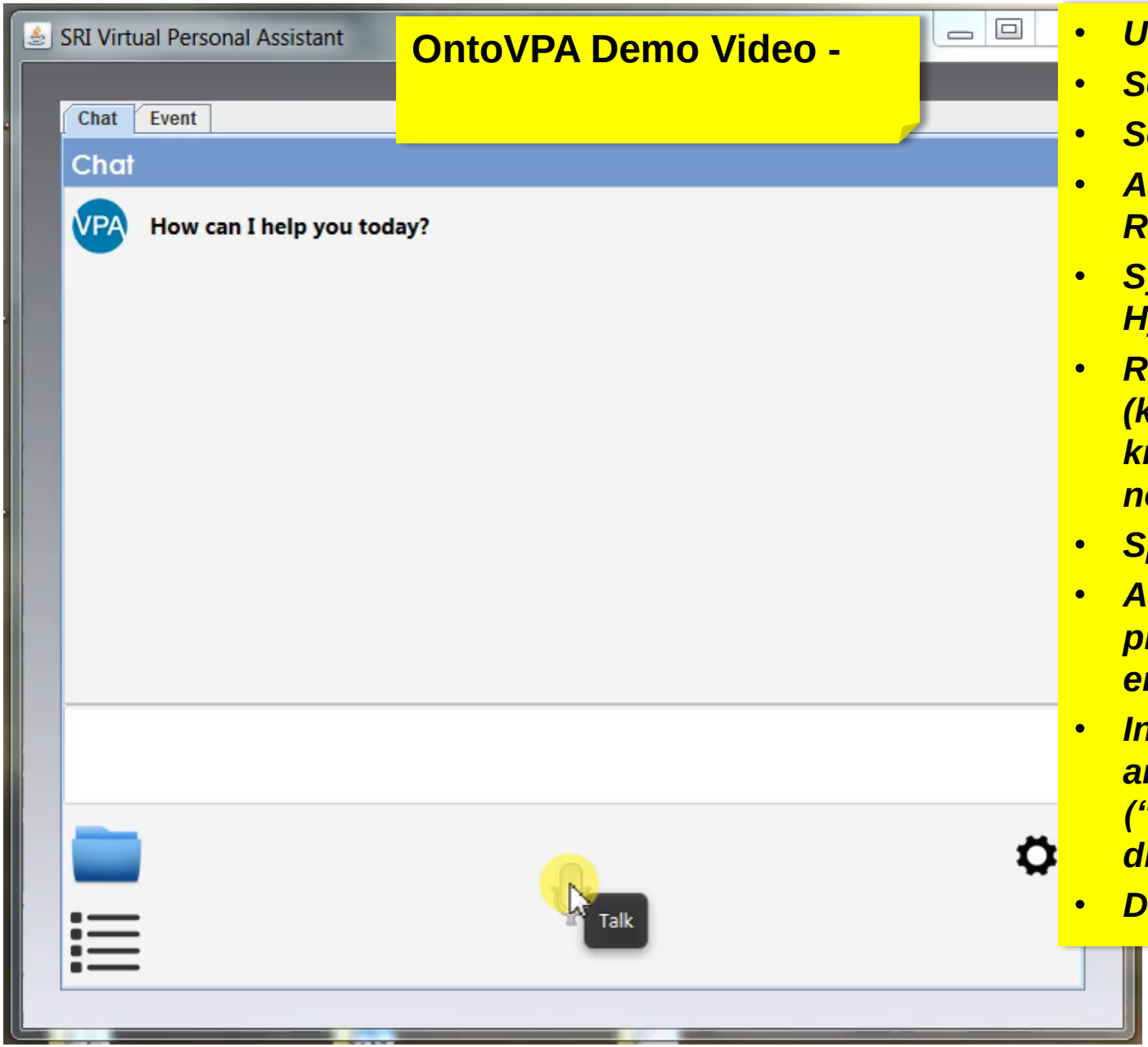
Siri is great and was / is revolutionary, but has some shortcomings when it comes to dialogue management (state tracking, ...) – SRI hasn’t stopped working on VPA’s…

What is OntoVPA?

- **A dialogue management system (DMS) for**
  - Conversational VPAs
    - Mixed initiative, usally task-oriented / goal driven, but not necessarily
    - Sub-dialogues
  - Deep domain knowledge and complex workflows
    - Virtual Personal ***Specialists***
  - Ontologies for domain- and dialogue-representation
    - Frame / intent -based (OWL2)
    - Rule-based – ***ontology-aware rules*** (SPARQL custom rule engine)
  - Declarative / knowledge-based
    - Customizable by swapping in and out ontologies
    - Little to no conventional programming
    - Reuse of standard ontologies and generic VPA Upper Level

Benefits of Formal Ontologies for DMS / VPA

- **More faithful domain representation**
  - Captures more semantics of the domain
  - Enables (sound and complete) reasoning over the domain
  - Closer to human conceptualization of the domain and hence „more useful“
- **Natural Language Understanding**
  - Understanding requires knowledge
  - Polysemy of natural language
    - Syonyms, hyponyms, hypernyms, ...
    - Disambiguation
    - Co-reference / anaphora resolution
    - Context-dependent interpretation of utterances in the context of the dialogue
- ***Formal* Knowledge Representation**
  - Explainability, transparency, understandability, reliabillity (Banking VPA) ...

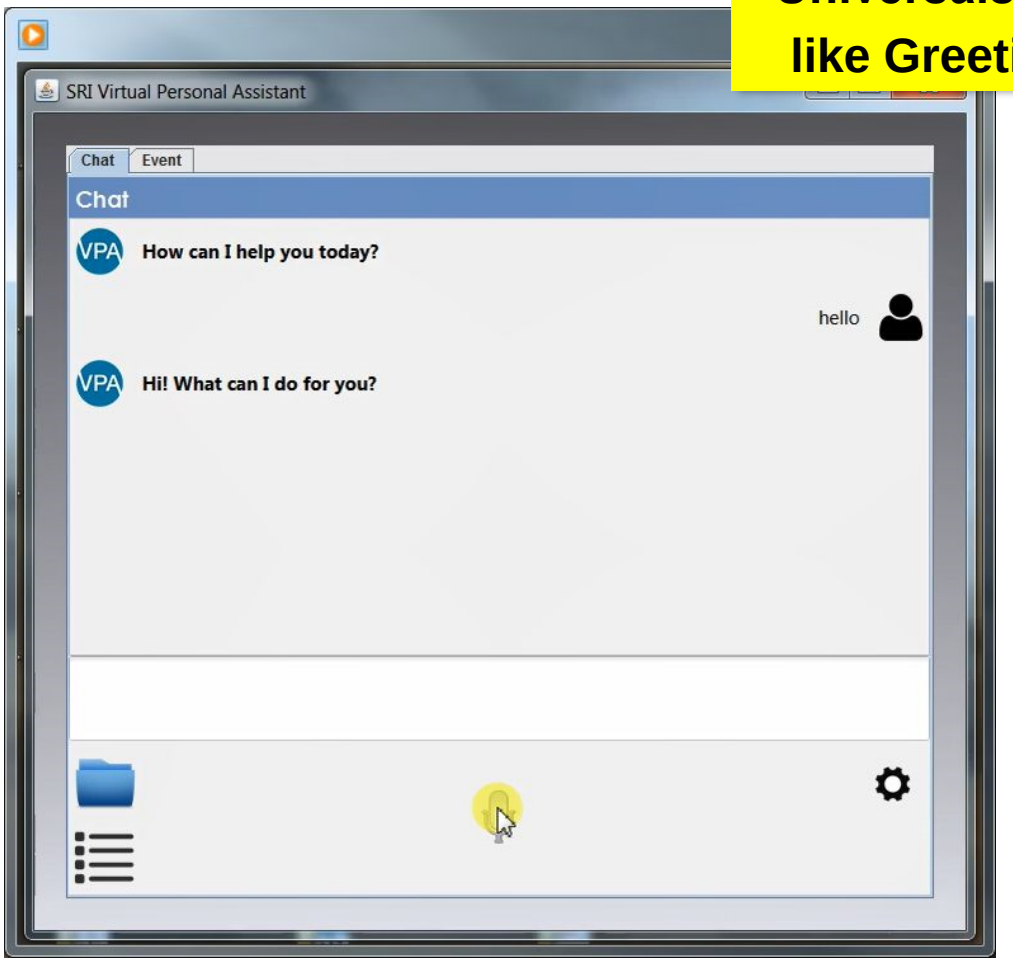


OntoVPA Demo Video -

- *Universals*
- *Semantic Search*
- *Semantic Similarity*
- *Anaphora Resolution*
- *Synonyms, Hypernyms, ...*
- *Reflexiveness (knows what it knows and what not)*
- *Spatial Search*
- *Ability to refer to previous dialogue entities*
- *Interpretation of arbitrary input (“Palo Alto”) in dialogue context*
- *Disambiguation*

OntoVPA Demo 1 / 23

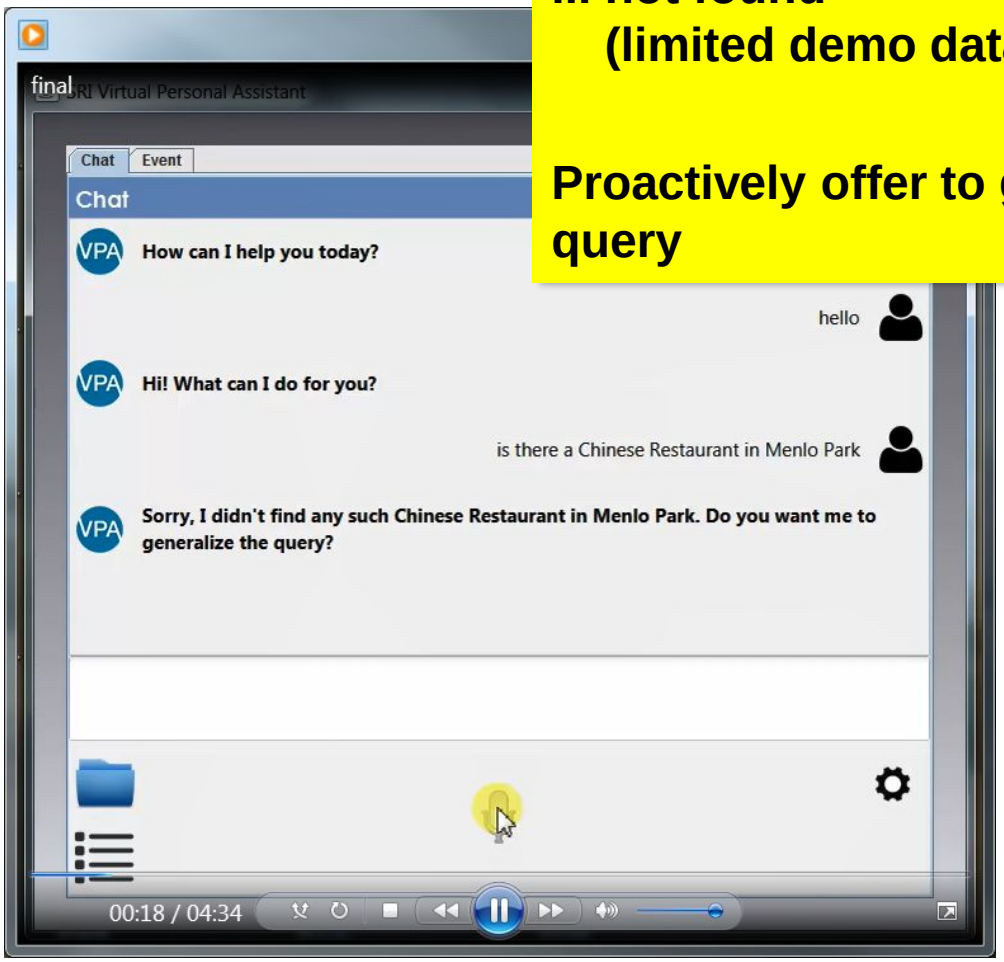
“Universals”
like GreetingIntent etc.



OntoVPA Demo 2 / 23


```
Intent1 : {
  type = FindPoiIntent
  dialogEntity = anon1 : {
      type = ChineseRestaurant
      directlyInside = MenloPark  }
 }
```

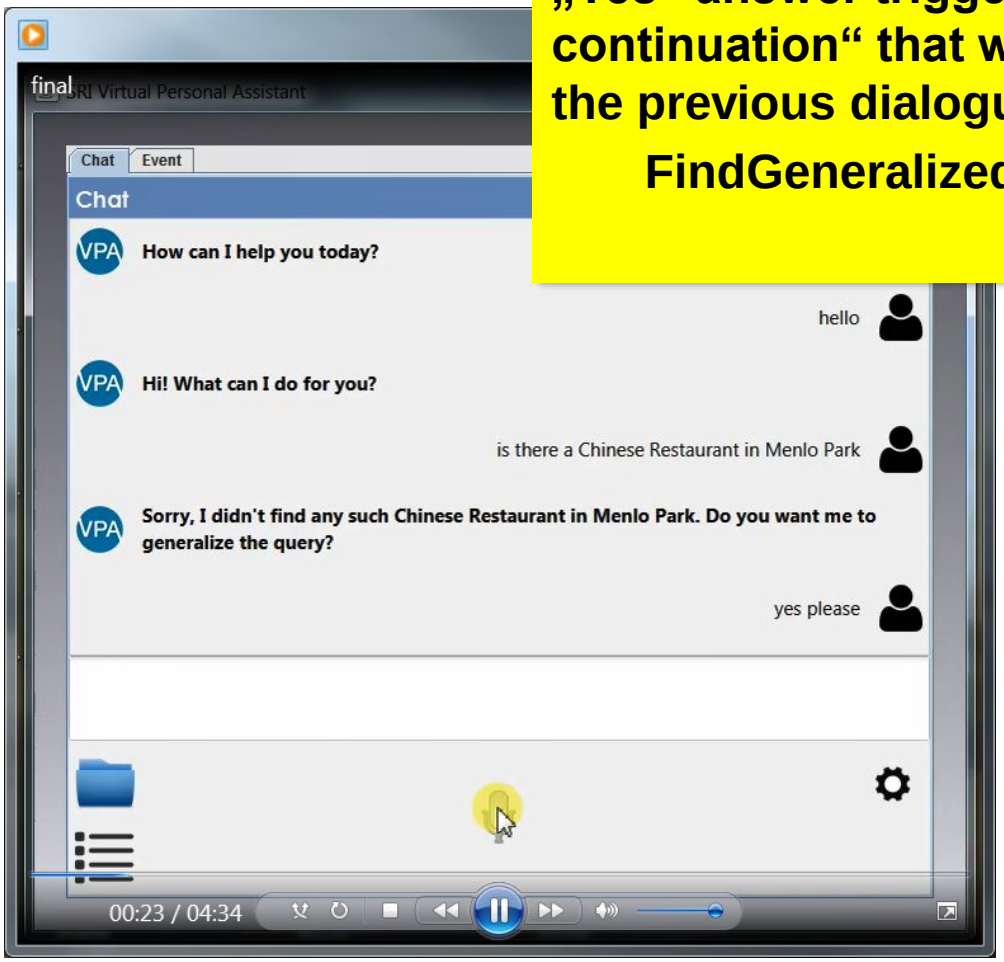
OntoVPA Demo 3 / 23



... not found
(limited demo data set)

Proactively offer to generalize query

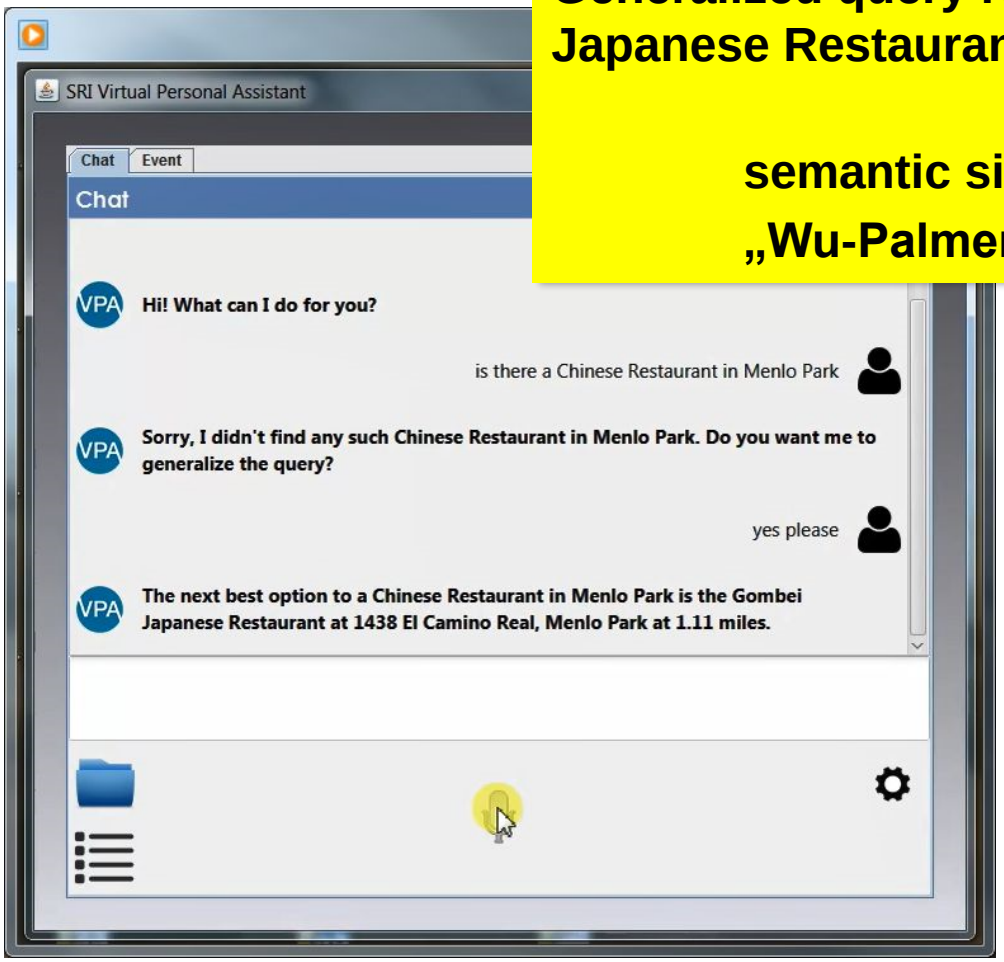
OntoVPA Demo 4 / 23



„Yes“ answer triggers the „yes continuation“ that was set up by the previous dialogue step ->

FindGeneralizedPOIIntent

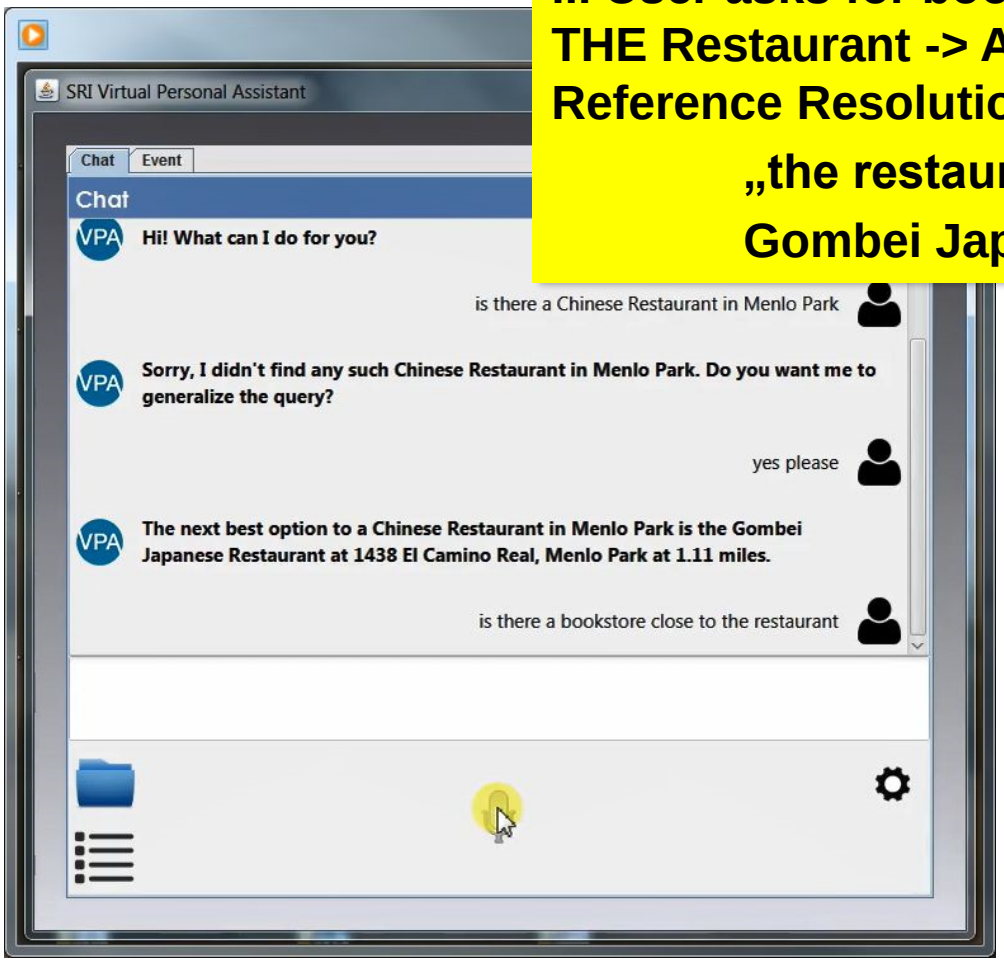
OntoVPA Demo 5 / 23



Generalized query returned Japanese Restaurant

semantic similarity „Wu-Palmer“

OntoVPA Demo 6 / 23

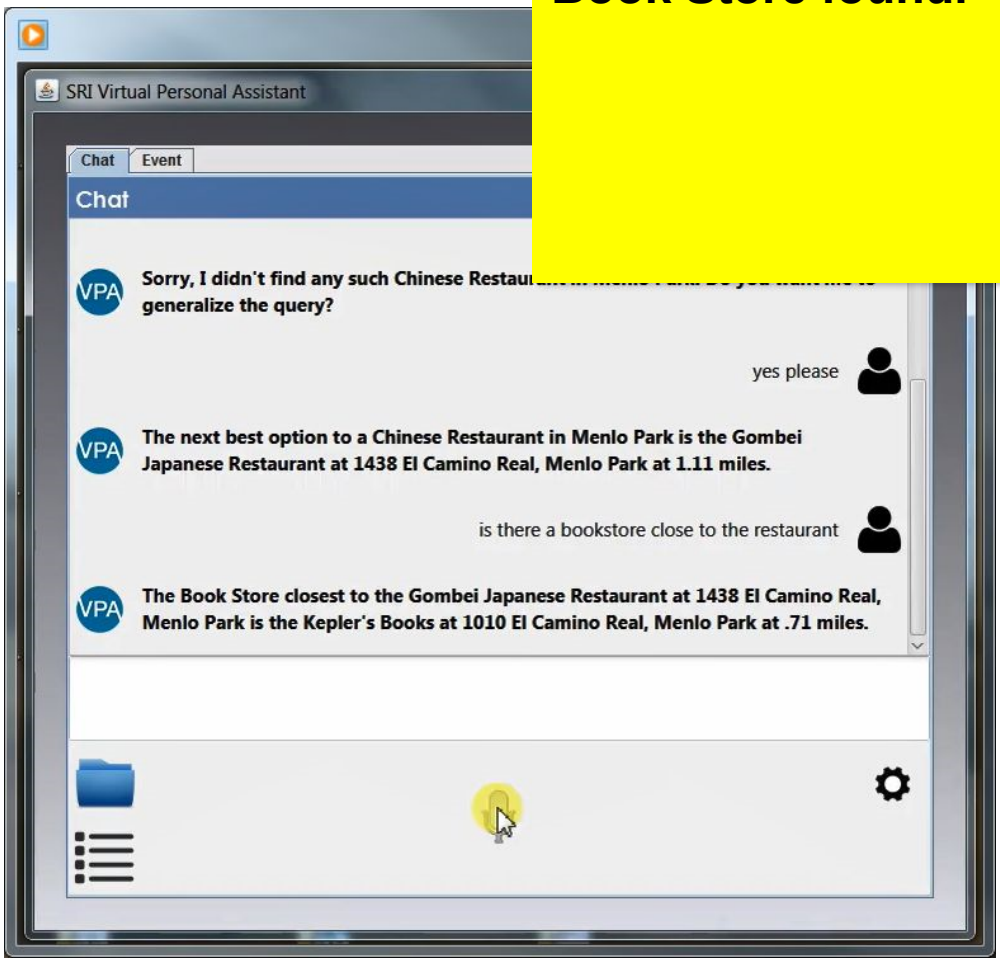


... User asks for bookstore close to THE Restaurant -> Anaphora / Co-Reference Resolution

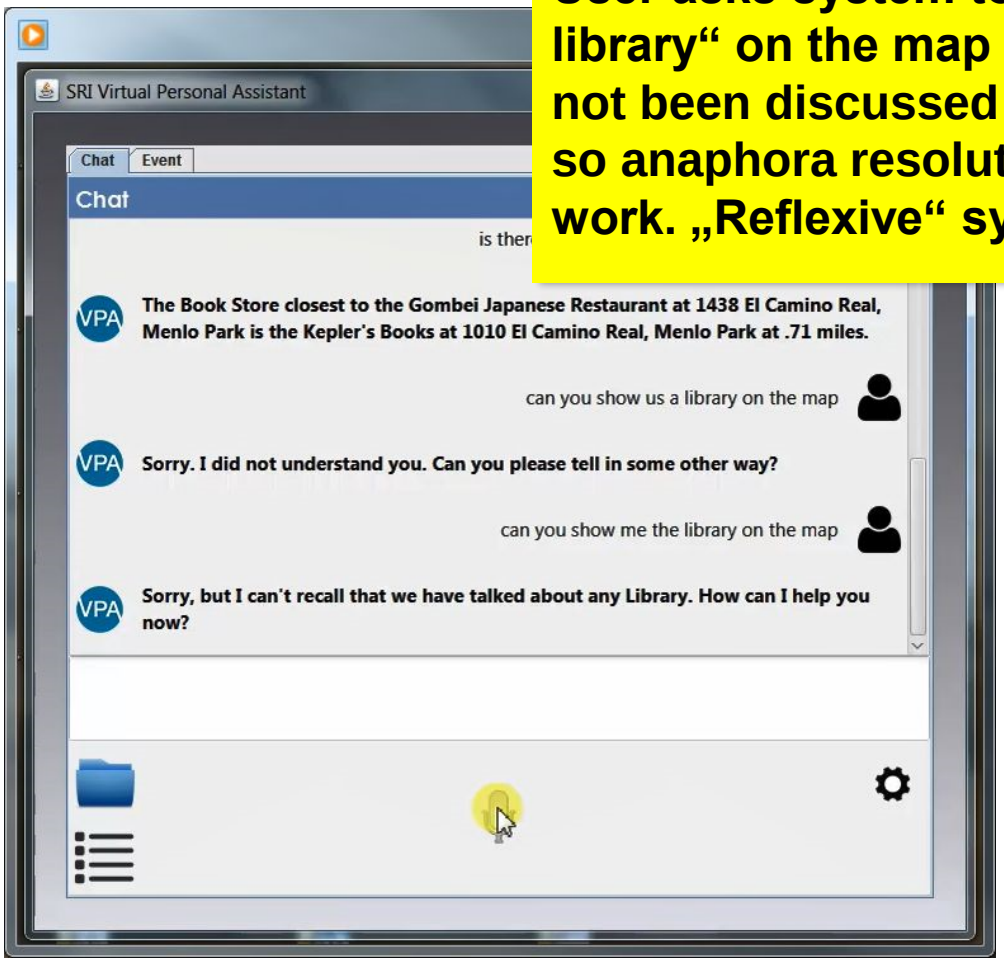
„the restaurant“ matches Gombei Jap. Restaurant

OntoVPA Demo 7 / 23

Book Store found.

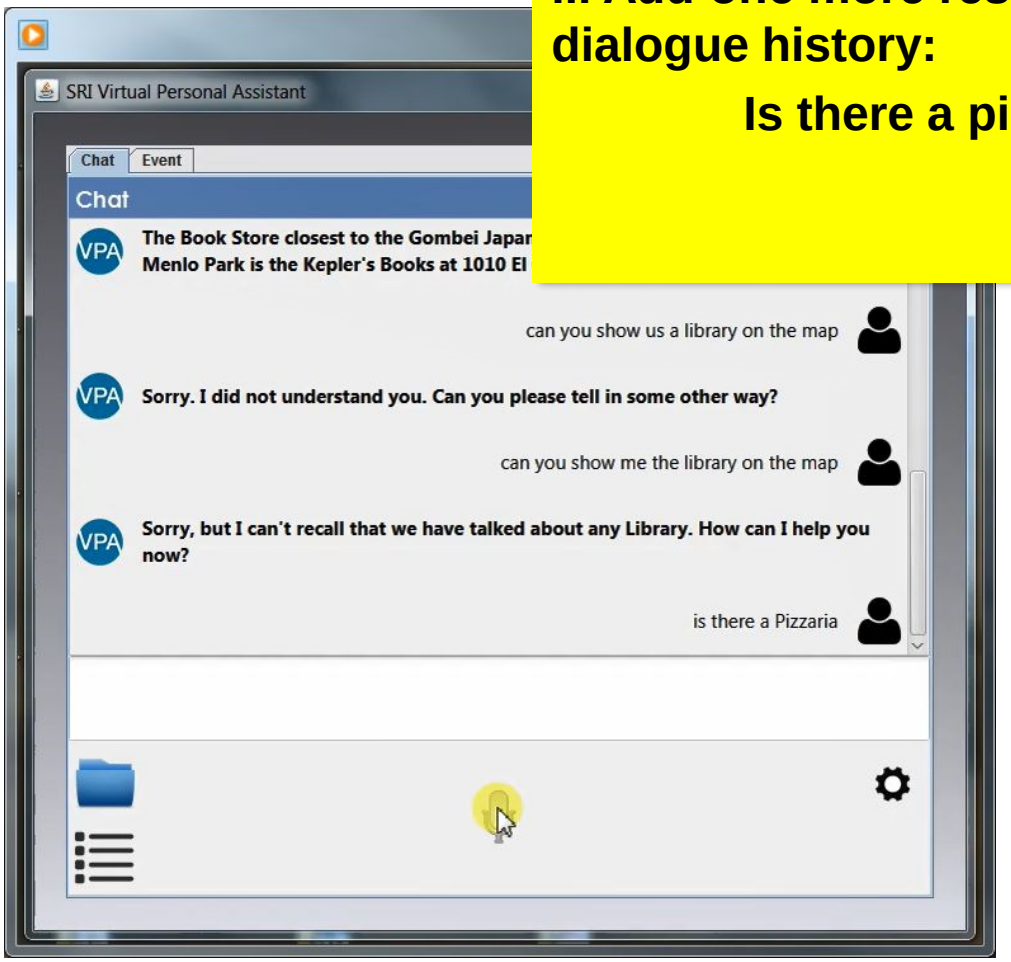


OntoVPA Demo 8 / 23



User asks system to show „the library“ on the map – library has not been discussed in the dialogue, so anaphora resolution does not work. „Reflexive“ system!

OntoVPA Demo 9 / 23

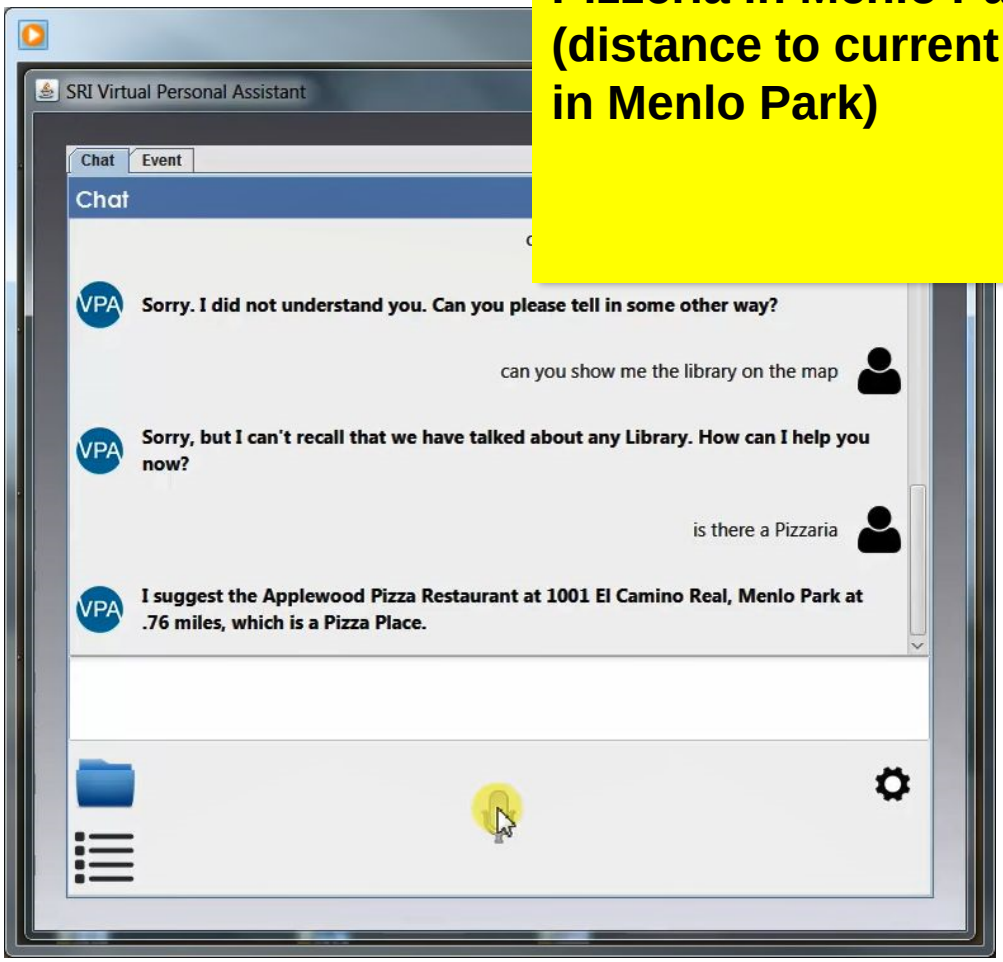


... Add one more restaurant to dialogue history:

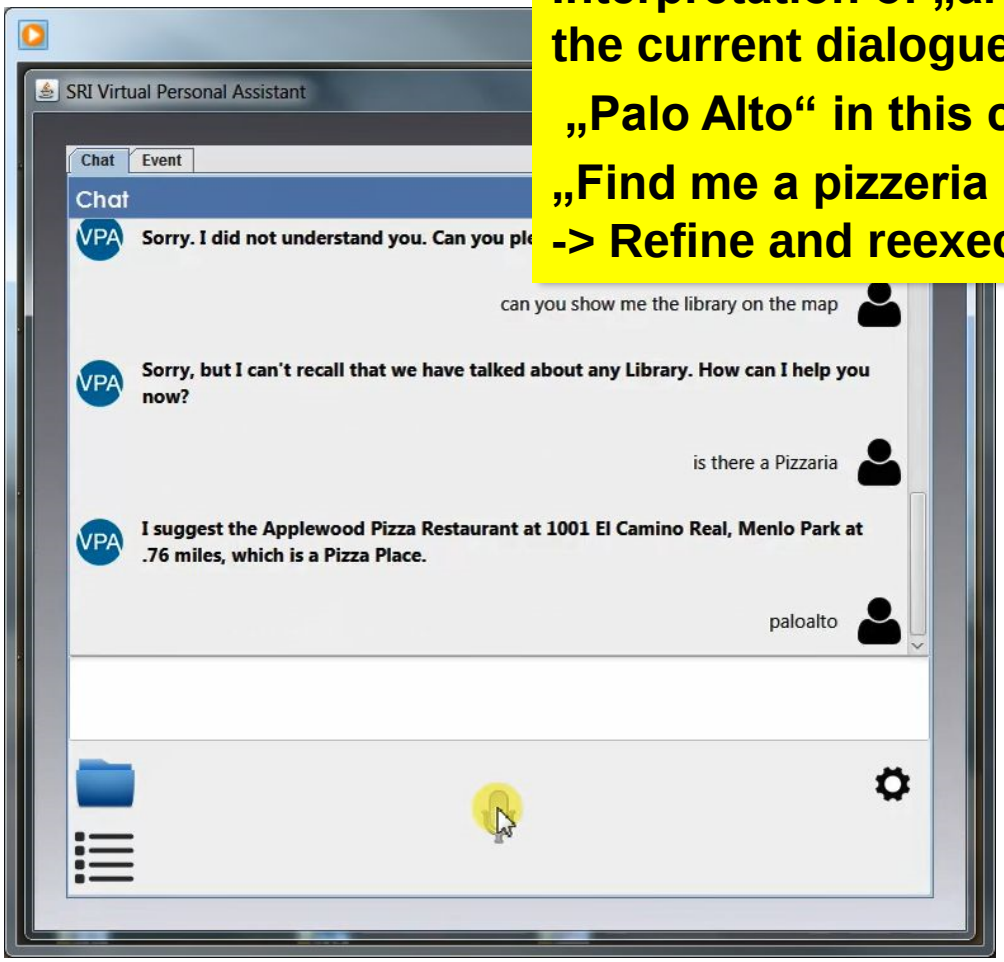
Is there a pizzeria?

OntoVPA Demo 10 / 23

Pizzeria in Menlo Park found (distance to current location at SRI in Menlo Park)



OntoVPA Demo 11 / 23



Interpretation of „arbitrary input“ in the current dialogue context

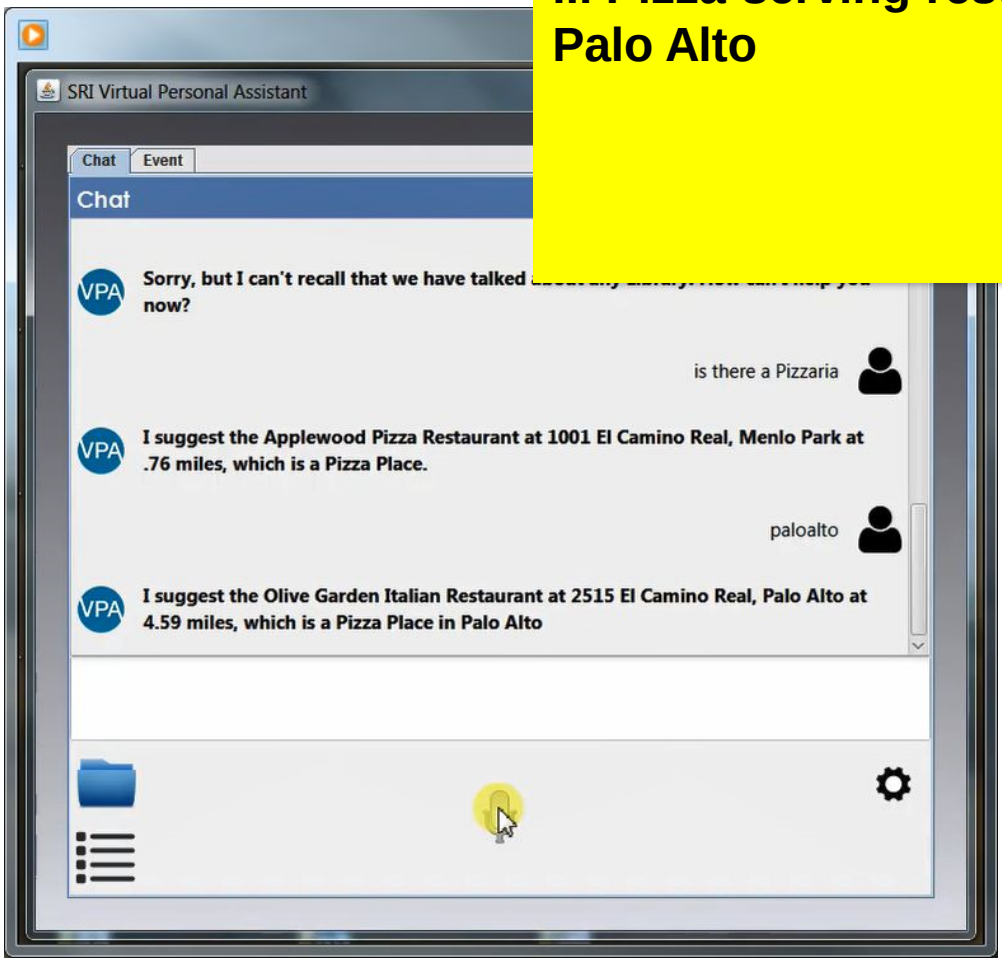
„Palo Alto“ in this context means:

„Find me a pizzeria in Palo Alto!“

-> Refine and reexecute prev. intent

OntoVPA Demo 12 / 23

... Pizza-serving restaurant found in Palo Alto



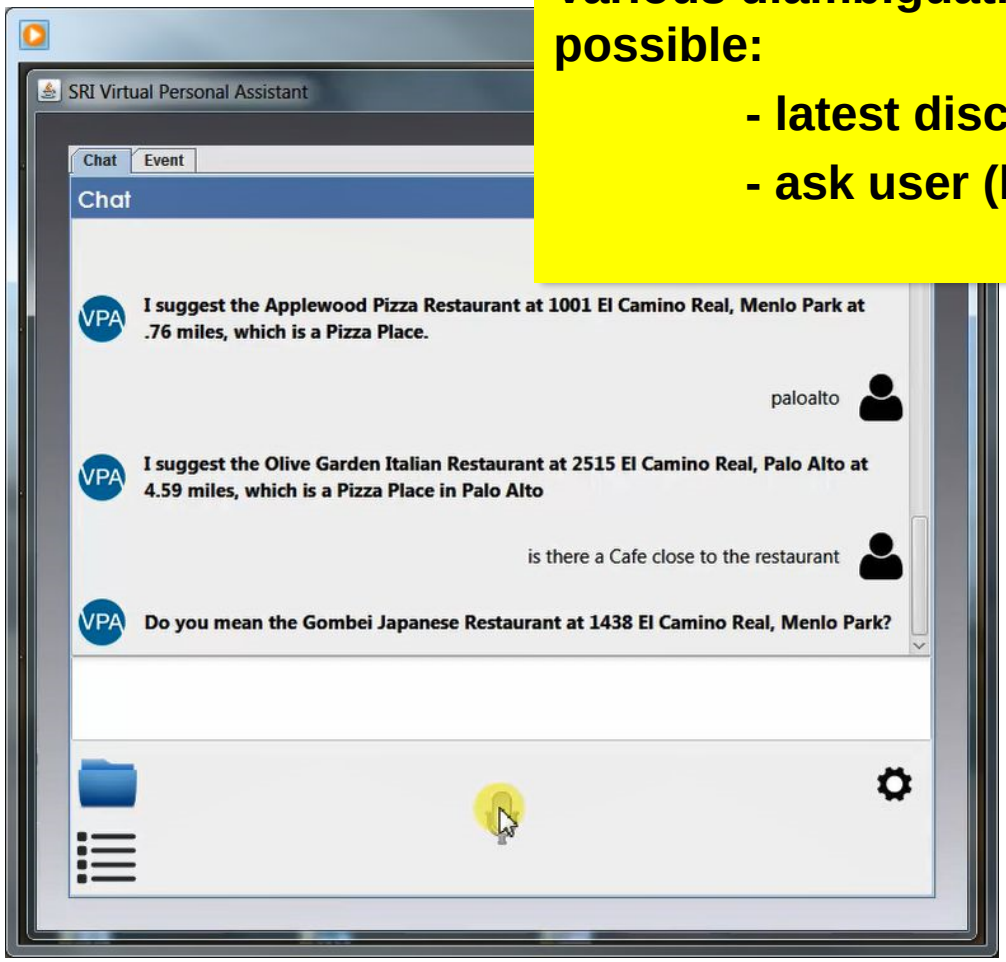
OntoVPA Demo 13 / 23

„Is there a cafe close to THE restaurant?“

-> ambiguous by now...

(3 restaurants in dialogue history)

OntoVPA Demo 14 / 23

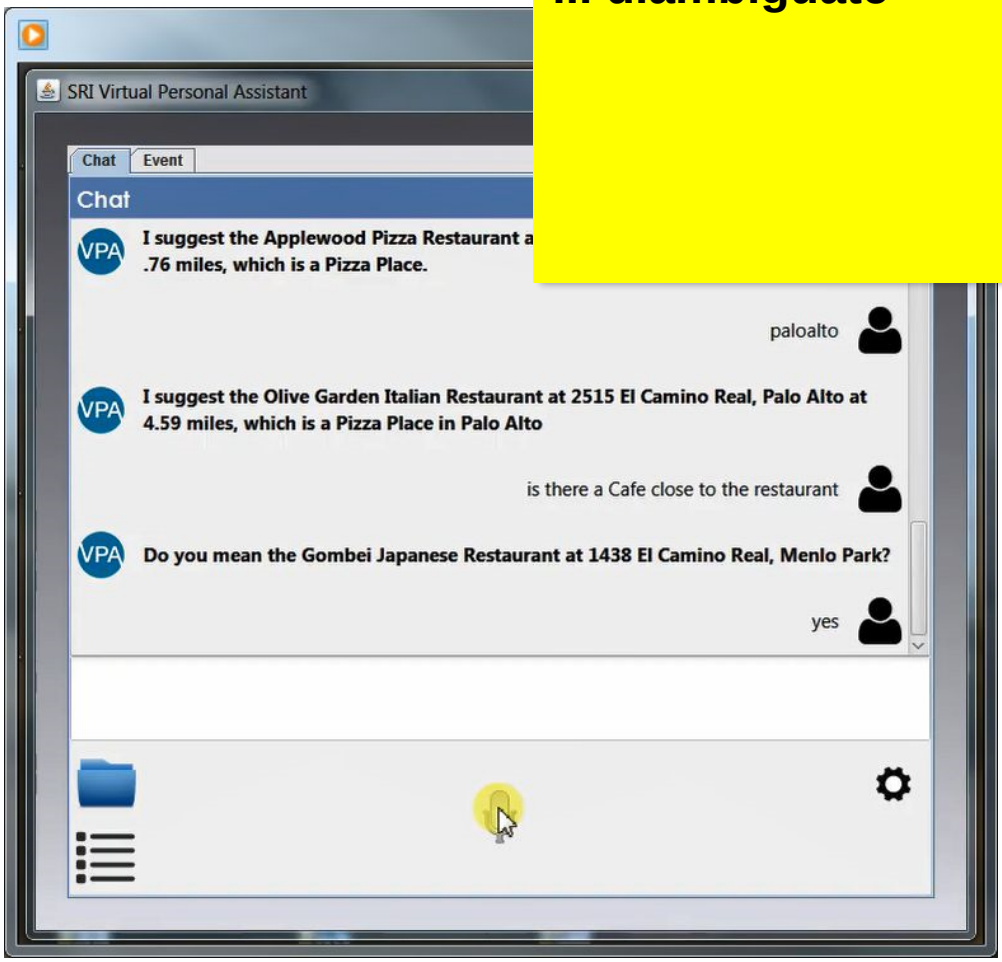


Various diambiguation strategies possible:

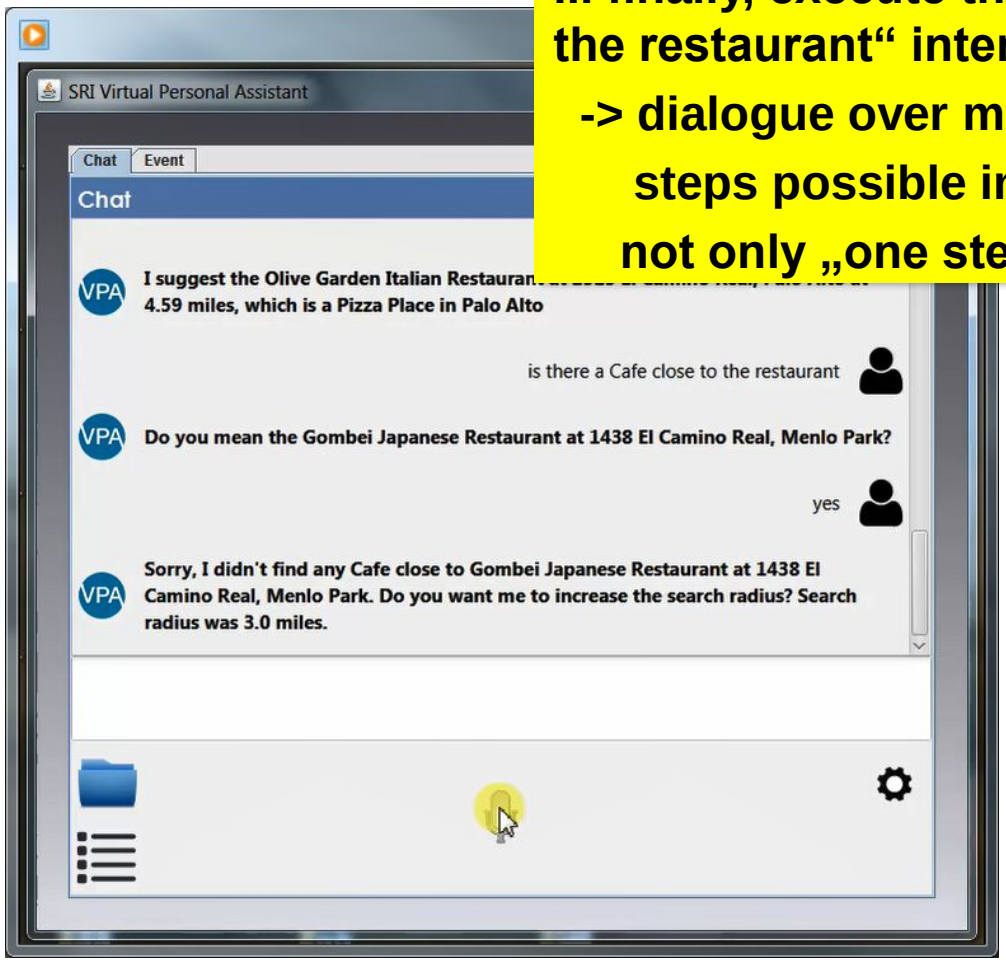
- latest discussed
- ask user (here!)

OntoVPA Demo 15 / 23

... diambiguate



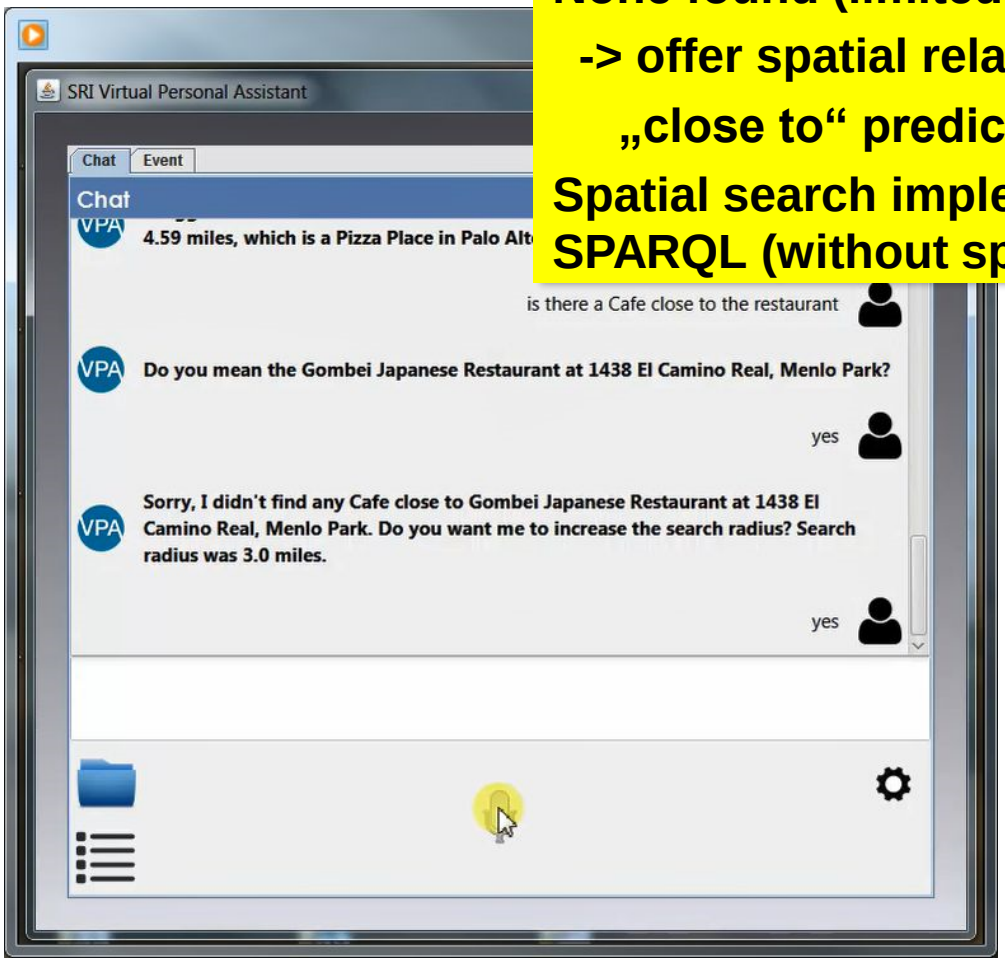
OntoVPA Demo 16 / 23



... finally, execute the „cafe close to the restaurant“ intent

-> dialogue over multiple steps possible in OntoVPA, not only „one step“ Q / A

OntoVPA Demo 17 / 23

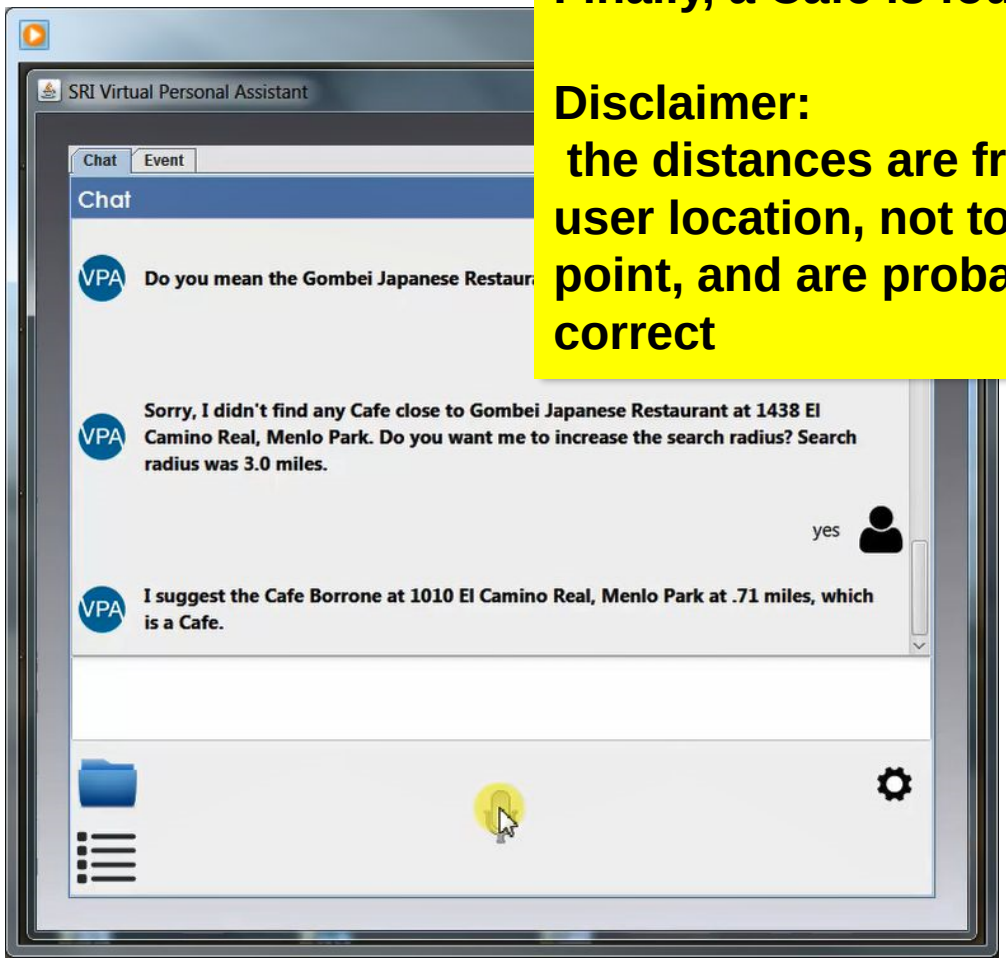


None found (limited dataset)

-> offer spatial relaxation of „close to“ predicate

Spatial search implemented in SPARQL (without spatial index)

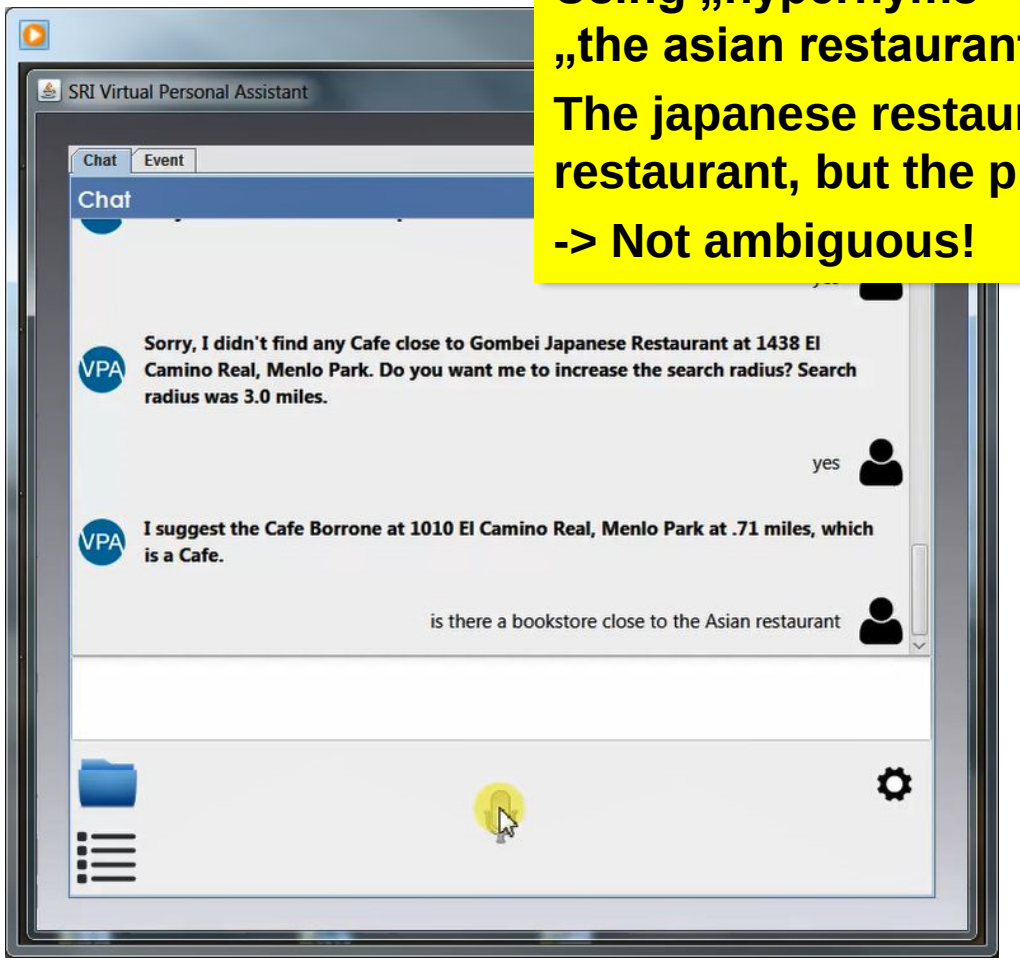
OntoVPA Demo 18 / 23



Finally, a Café is found...

Disclaimer:
the distances are from current user location, not to reference point, and are probably not 100% correct

OntoVPA Demo 19 / 23

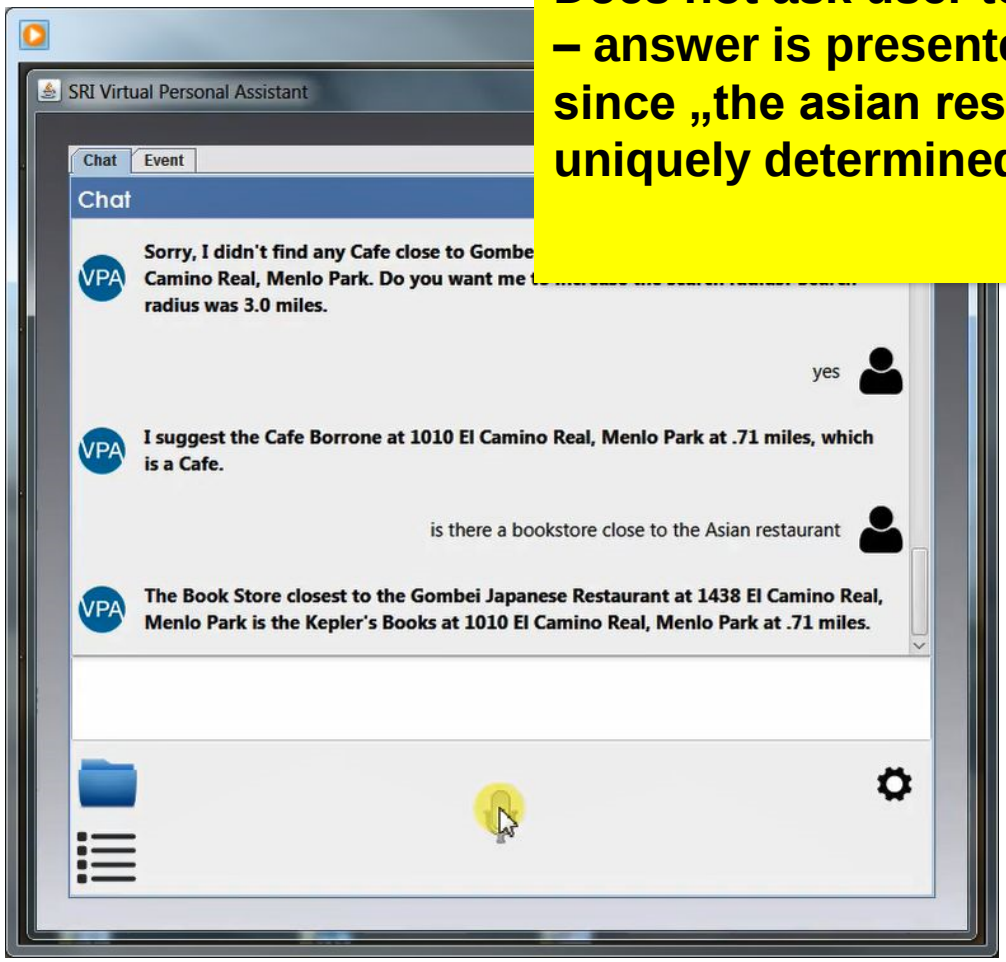


Using „hypernyms“ in anaphora – „the asian restaurant“

The japanese restaurant is an asian restaurant, but the pizzerias are not

-> Not ambiguous!

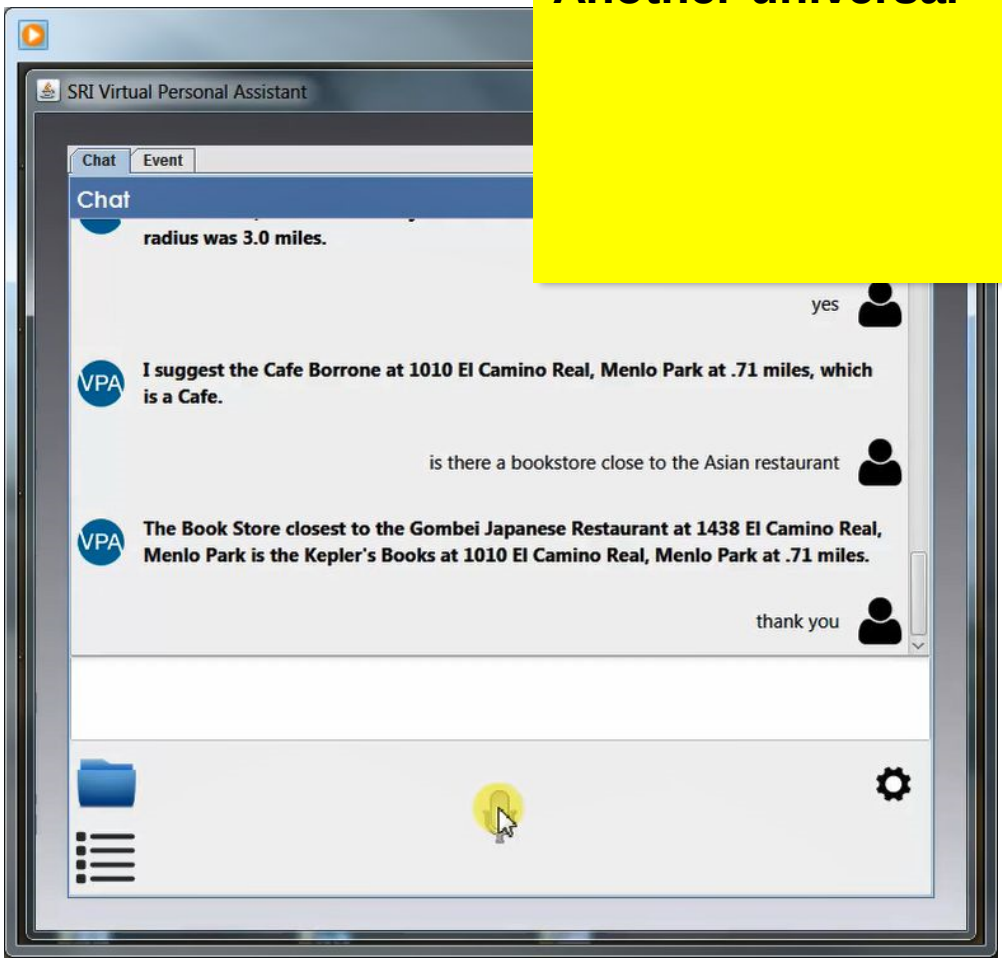
OntoVPA Demo 20 / 23



Does not ask user to disambiguate – answer is presented right away, since „the asian restaurant“ is uniquely determined.

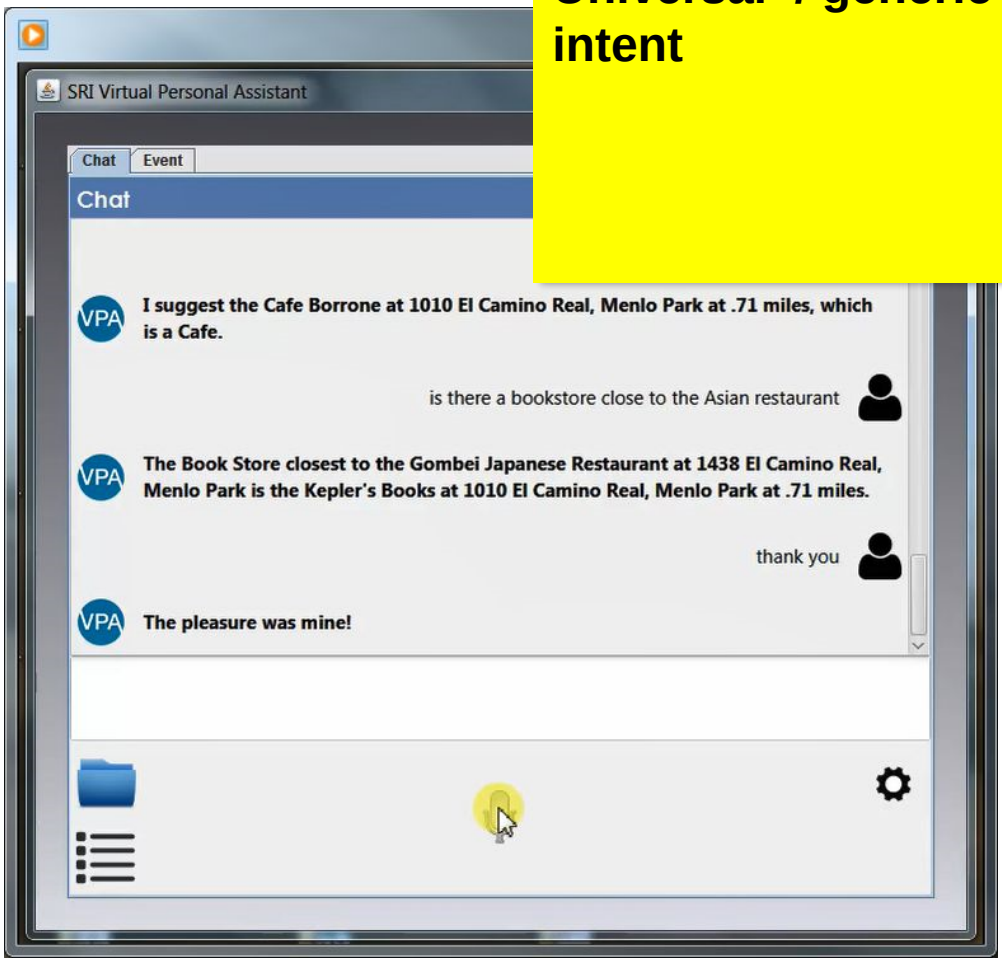
OntoVPA Demo 21 / 23

Another universal - „thank you“



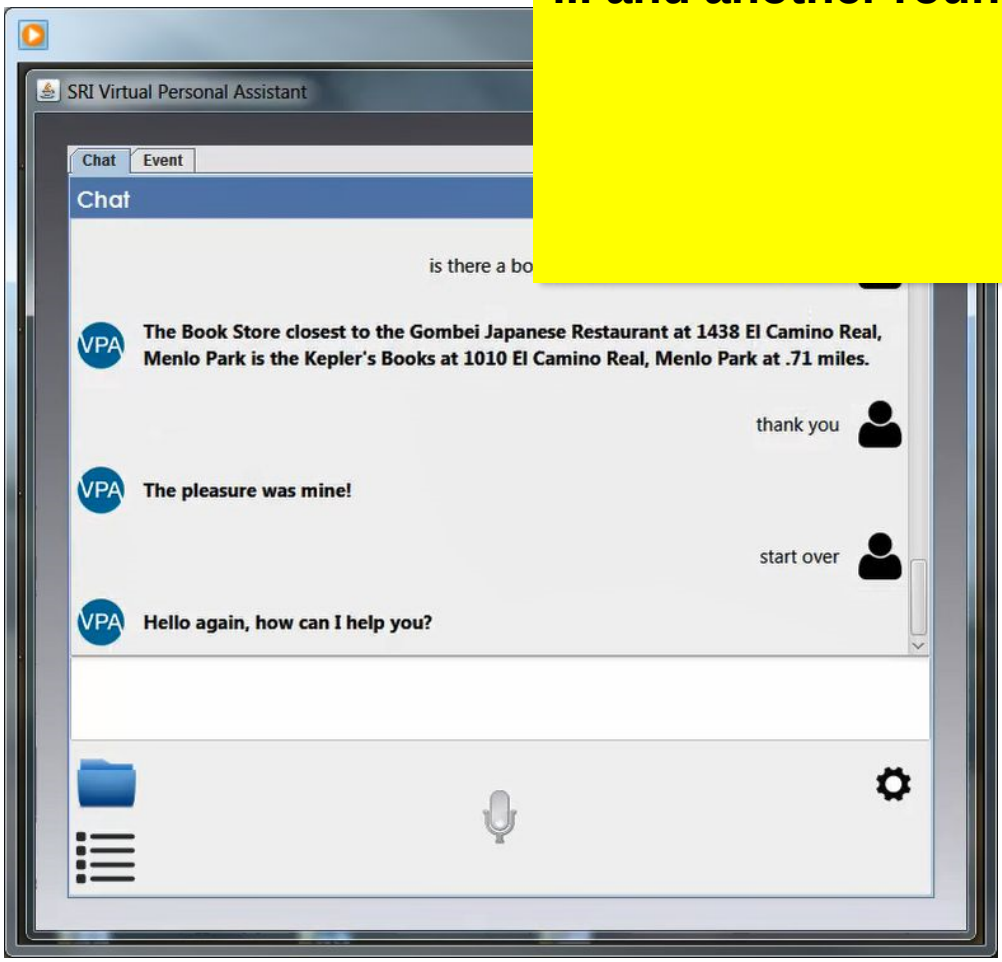
OntoVPA Demo 22 / 23

Universal / generic „Start over“ intent

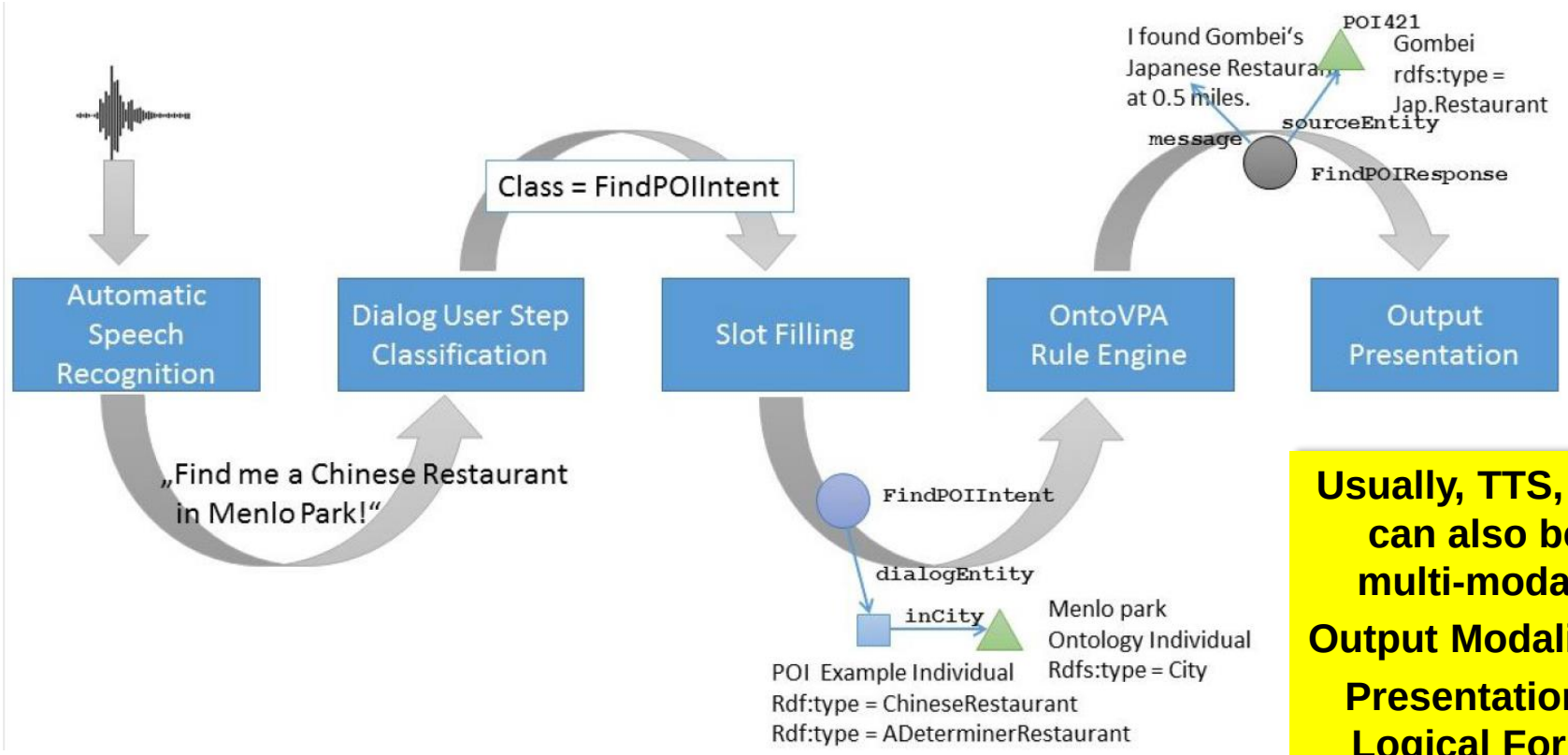


OntoVPA Demo 23 / 23

... and another round ☺



OntoVPA Processing Pipeline



Usually, TTS, but can also be multi-modal!

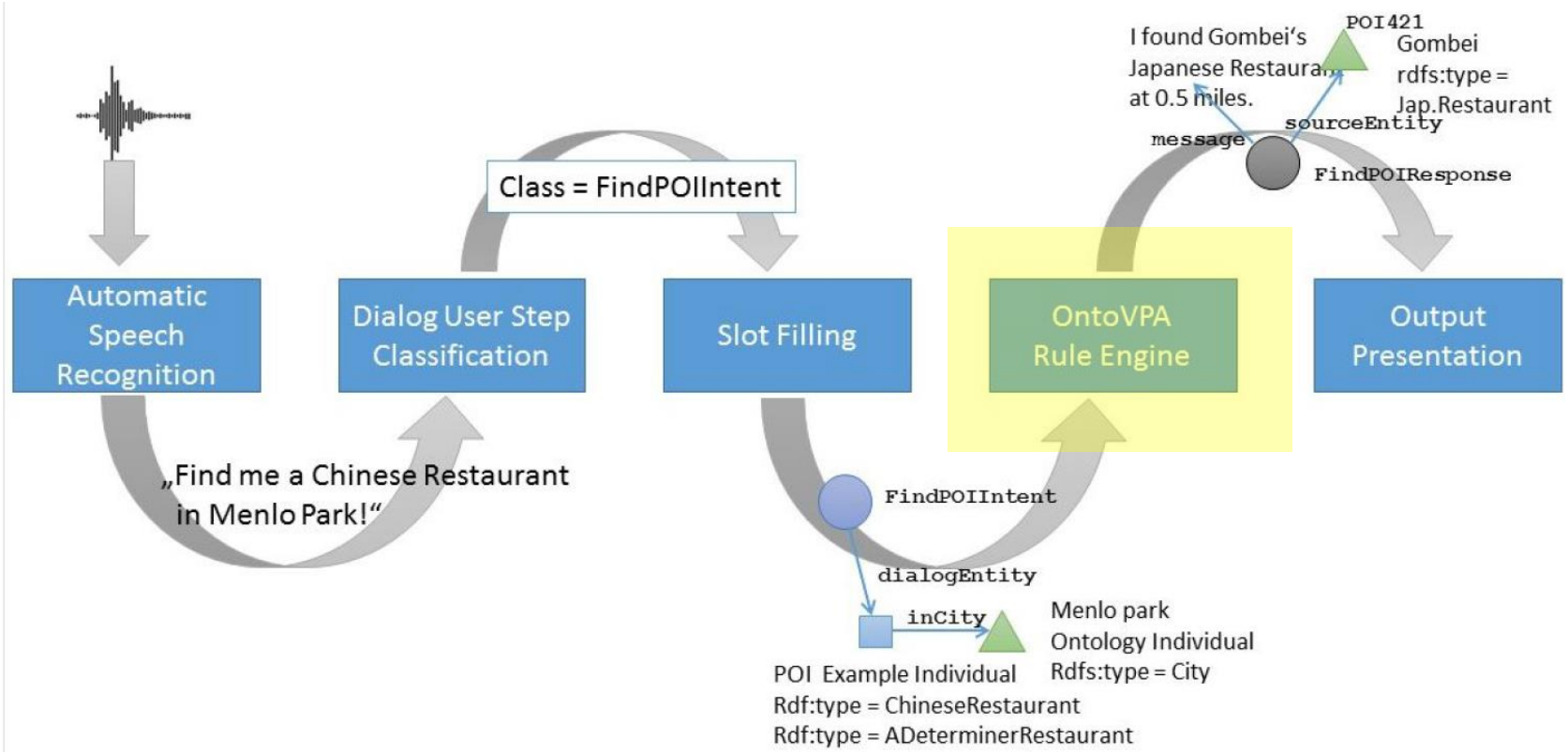
Output Modalities

Presentations

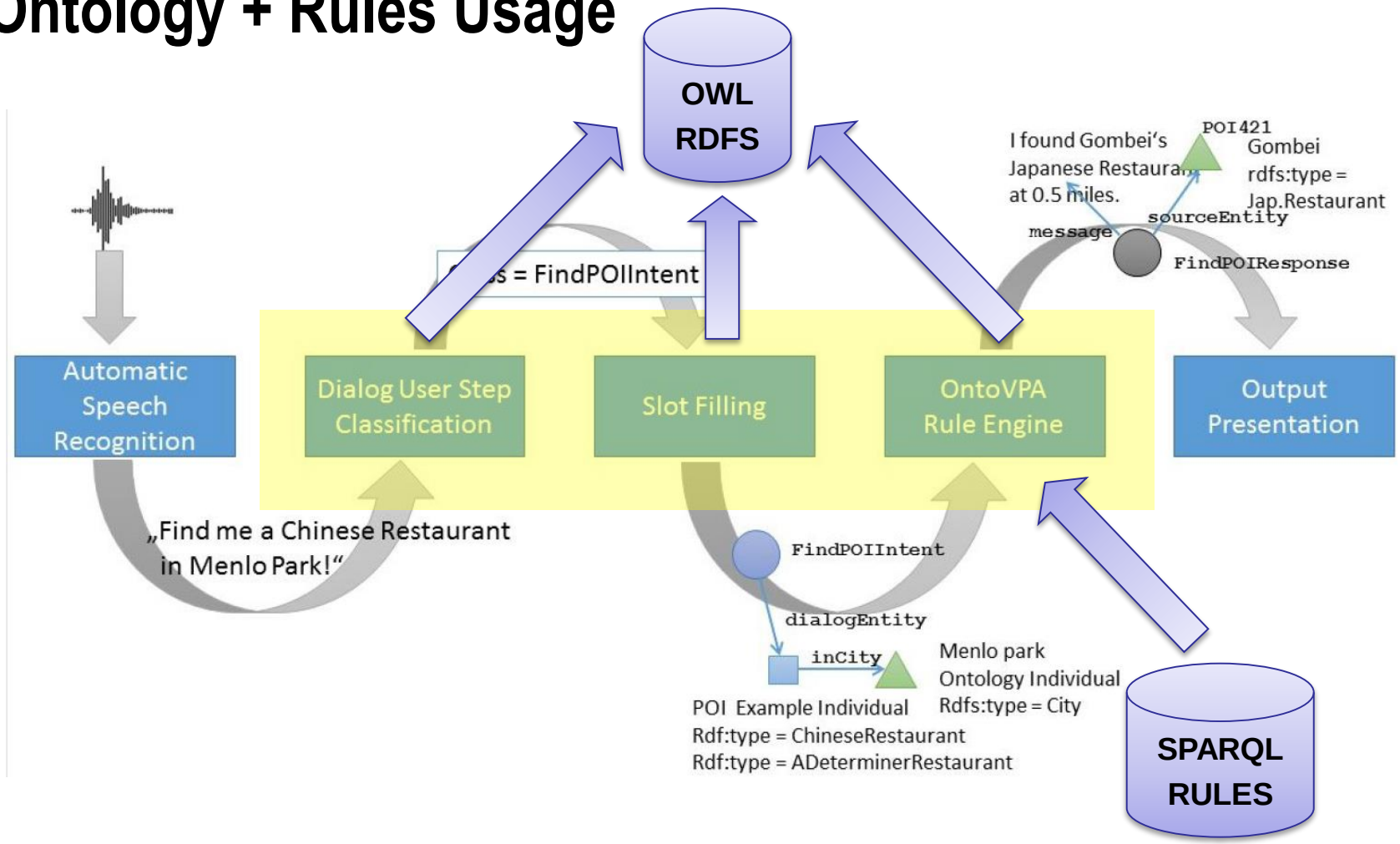
Logical Form

“Plug & Play” Component Architecture – ASR, TTS, Classifier exchangeable.

OntoVPA Processing Pipeline – Dialogue Management System



OntoVPA Processing Pipeline – Ontology + Rules Usage



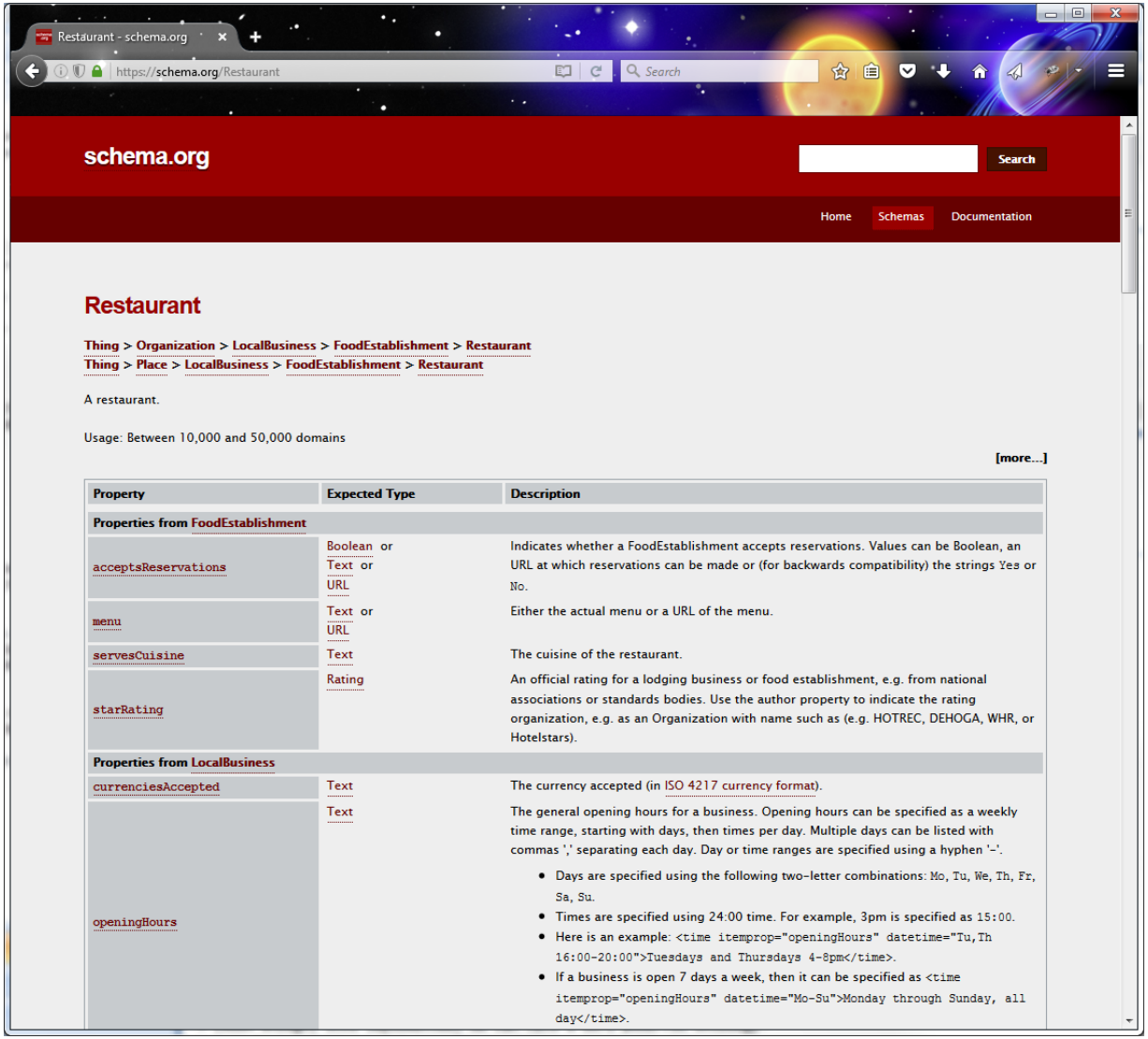
Structure of OntoVPA Ontology

- **Domain Ontology (Static)**
  - Domain Vocabulary („TBox“)
    - Classes: POI, Restaurant, ChineseRestaurant, City, ...
    - Relations: inCity, nearBy, name, hasAddress, ...
  - Reuse existing ontologies where possible

OWL2 offers

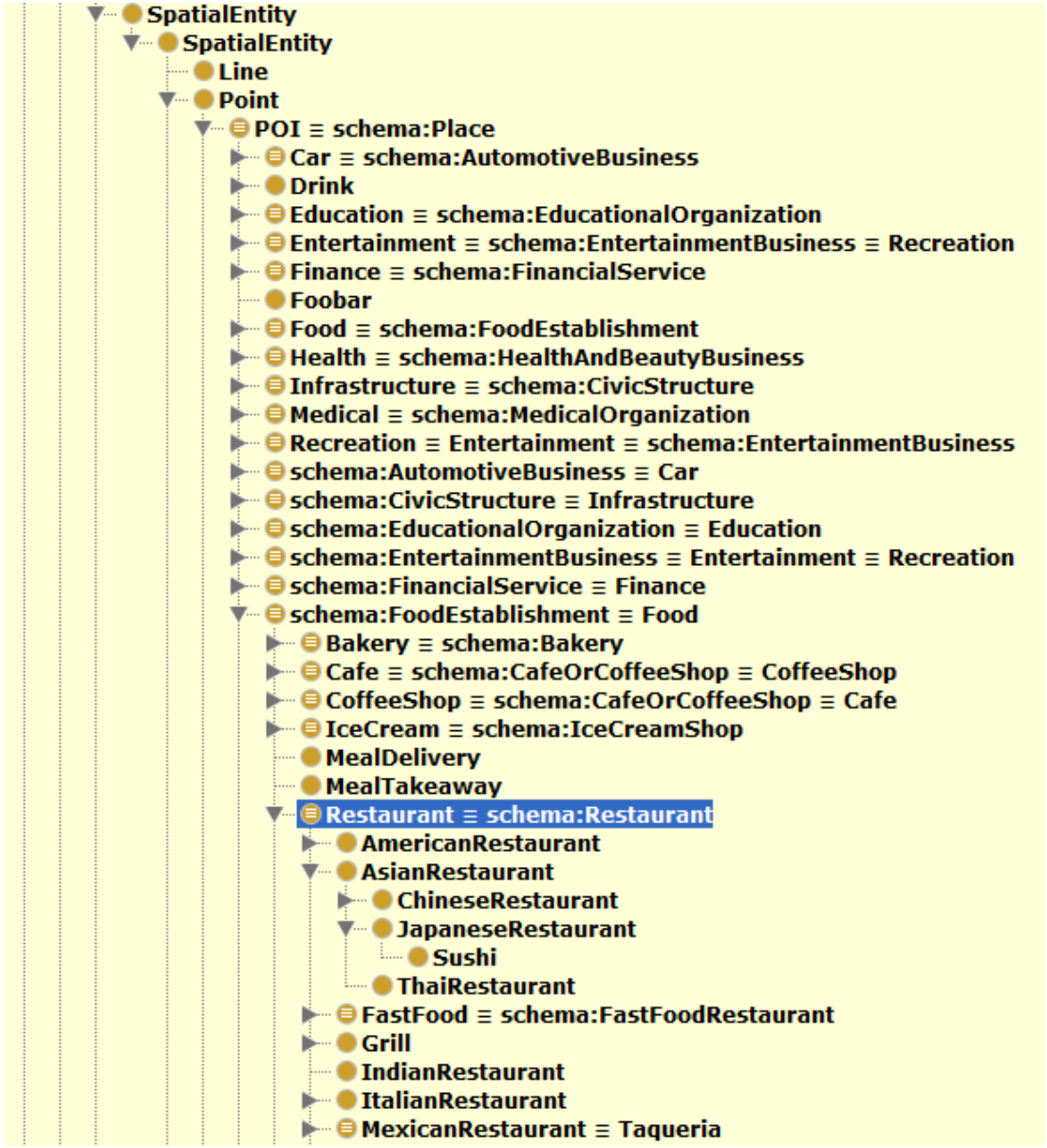
- **Multiple inheritance for classes & properties (relations)**
- **Multiple types per individual**
- **Semi-structured “data” representation**
- **Optional & required properties / parameters**

RDFS Version of Schema.org „Upper Level Ontology“

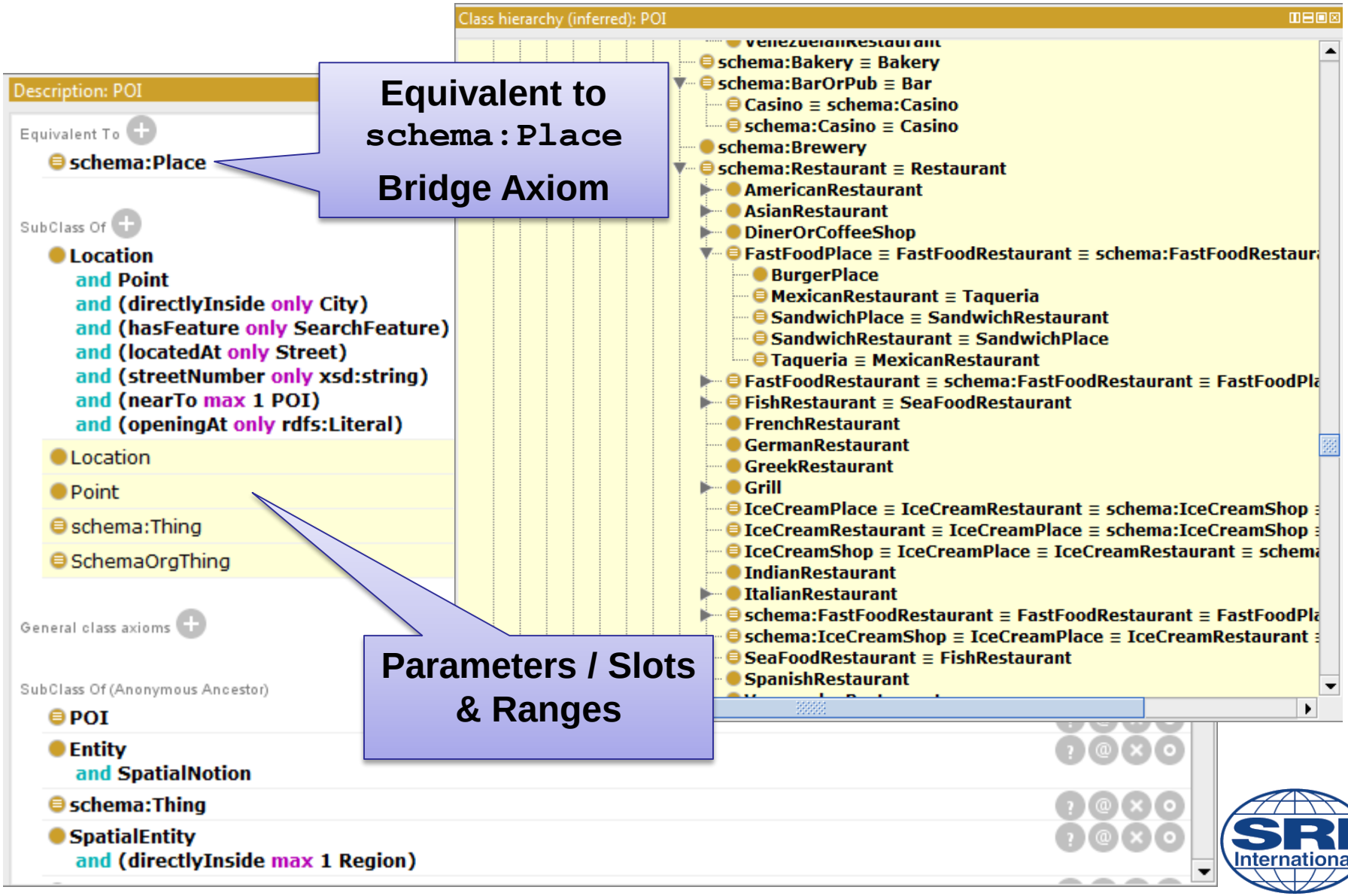


- Person
- Place
  - Accommodation
    - Apartment
    - CampingPitch
    - House
      - SingleFamilyResidence
    - Room
      - HotelRoom
      - MeetingRoom
    - Suite
  - AdministrativeArea
    - City
    - Country
    - State

POI Domain Ontology – Suclasses of schema:Restaurant

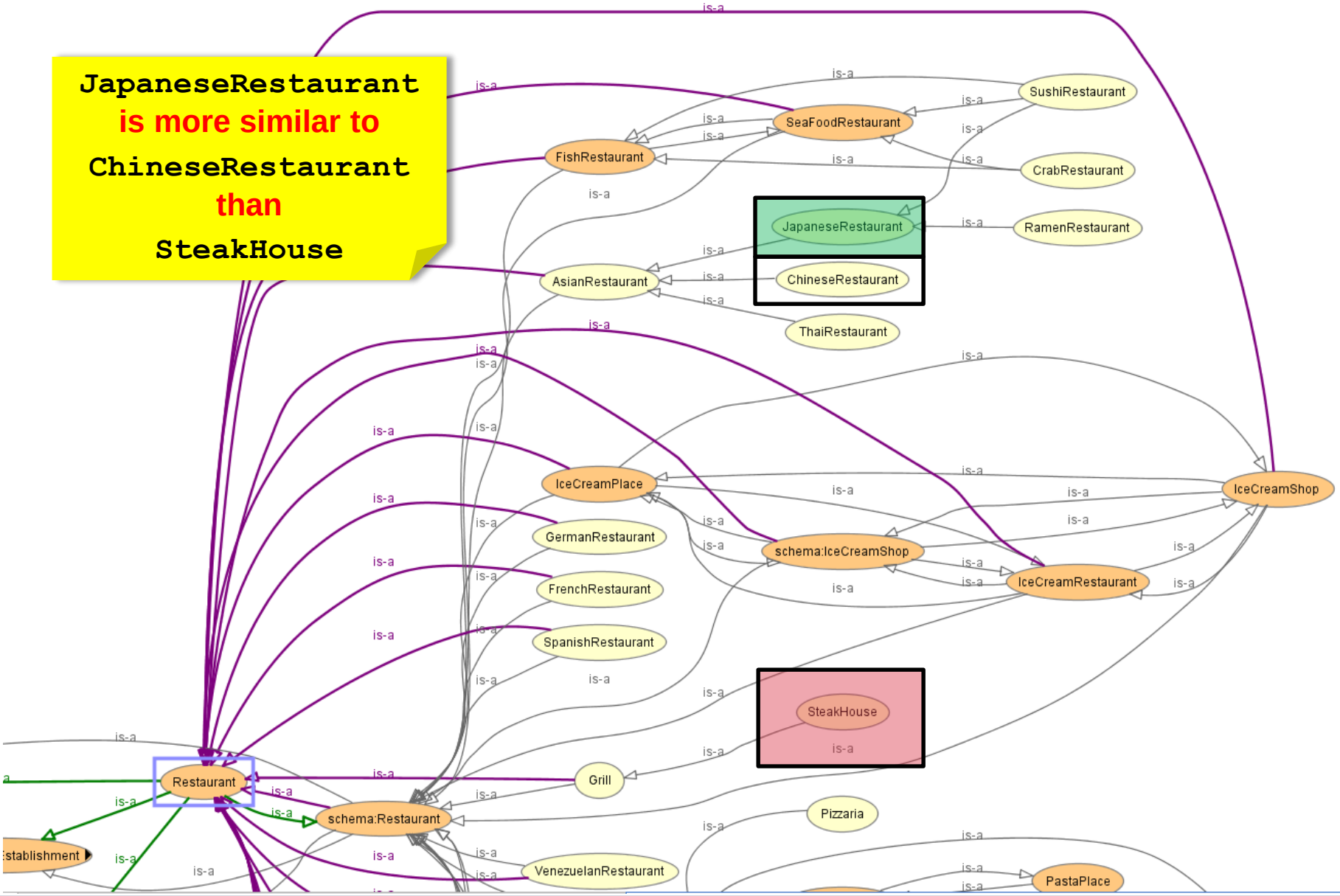


Definition of “Point of Interest (POI)” Class



Ontology-Based Similarity (~ Wu-Palmer Similarity)

JapaneseRestaurant
is more similar to
ChineseRestaurant
than
SteakHouse



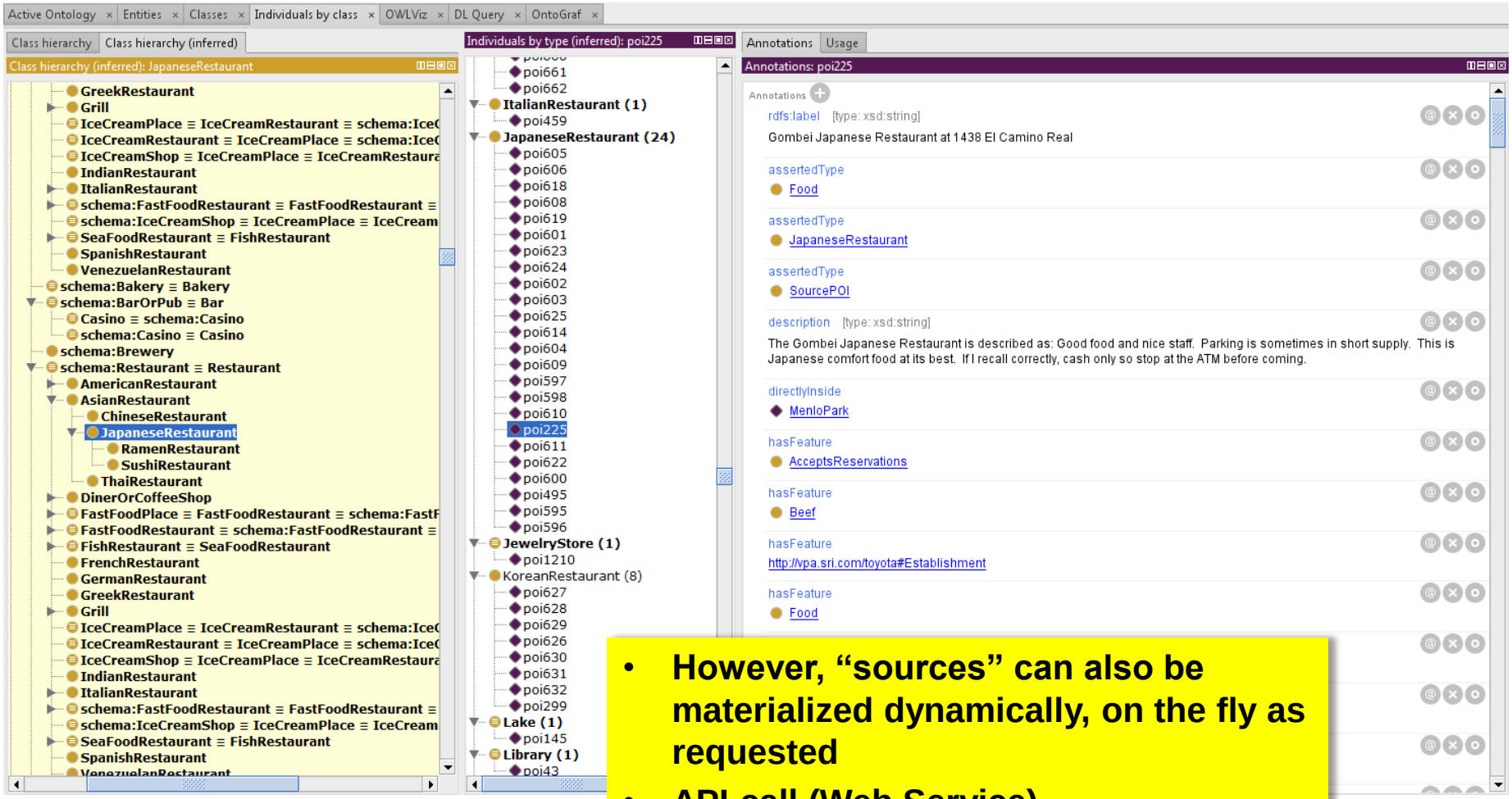
Description of OntoVPA Ontology

- **Domain Ontology (Static)**
  - Domain Vocabulary („TBox“)
    - Classes: POI, Restaurant, ChineseRestaurant, City, ...
    - Relations: inCity, nearBy, name, hasAddress, ...
  - Domain „Database“ („ABox“)
    - Instances & relationships: restaurant123, inCity(restaurant123, menloPark), ...

OWL2 offers

- **Multiple inheritance for classes & properties (relations)**
- **Multiple types per individual**
- **Semi-structured “data” representation**
- **Optional & required properties / parameters**

Ontology Domain Instances – “POI Database” ABox



- **However, “sources” can also be materialized dynamically, on the fly as requested**
- **API call (Web Service)**
- **Federated Sparql / Sparql endpoint**

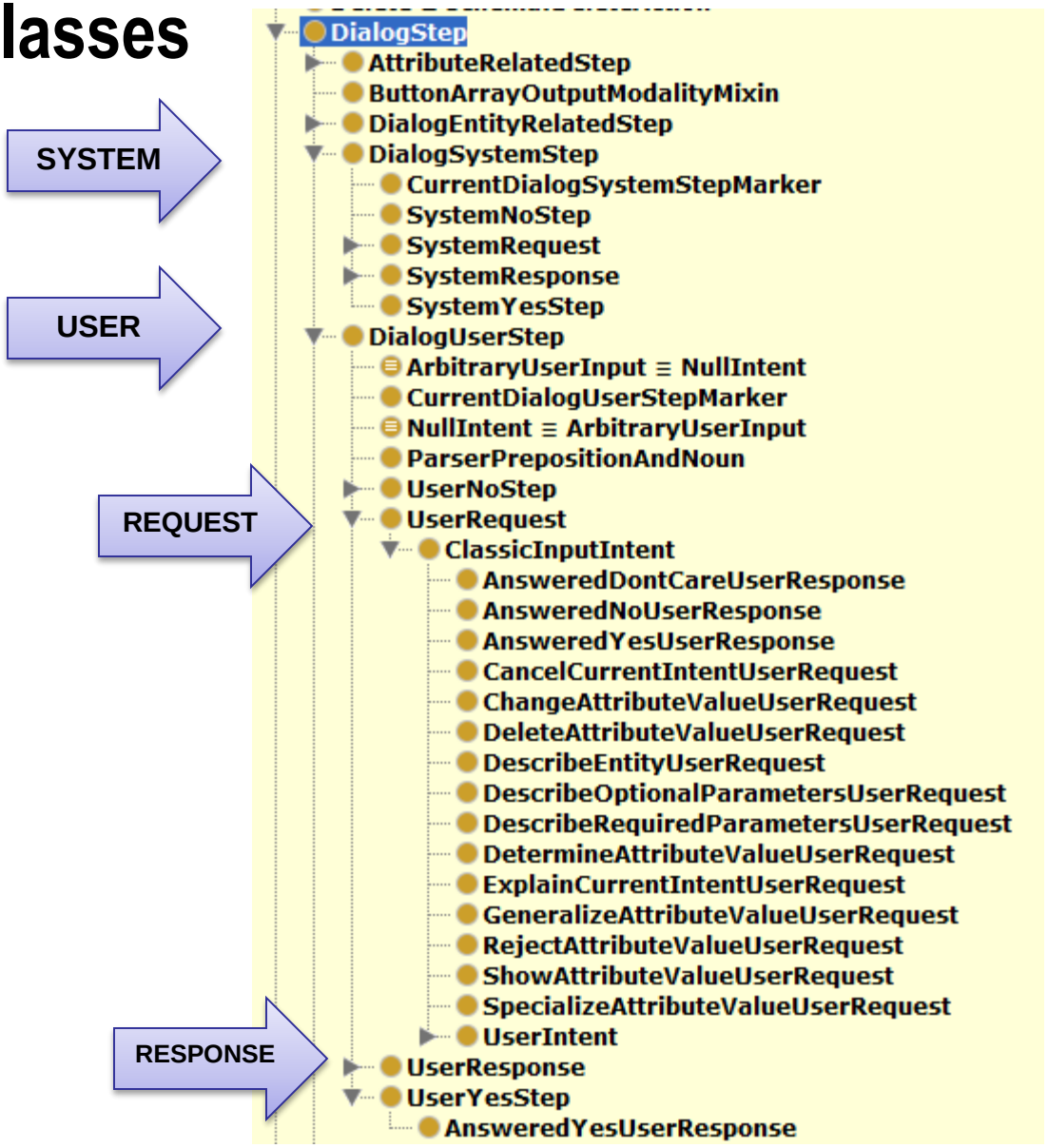
Structure of OntoVPA Ontology

- **Domain Ontology (Static)**
  - Domain Vocabulary („TBox“)
    - Classes: POI, Restaurant, ChineseRestaurant, City, ...
    - Relations: inCity, nearBy, name, hasAddress, ...
  - Domain „Database“ („ABox“)
    - Instances & relationships: restaurant123, inCity(restaurant123, menloPark), ...
- **Dialogue Ontology (Static)**
  - Speech Act Theory Inspired
    - Request, Response, UserRequest, UserResponse, SystemRequest, SystemResponse, Yes/No, Greeting, ...
    - UserIntents are special UserRequests: FindPOIIntent

OWL2 offers

- **Multiple inheritance for classes & properties (relations)**
- **Multiple types per individual**
- **Semi-structured “data” representation**
- **Optional & required properties / parameters**

OntoVPA Dialogue Ontology – Dialogue Steps & Subclasses



Structure of “Find Point of Interest” Intent

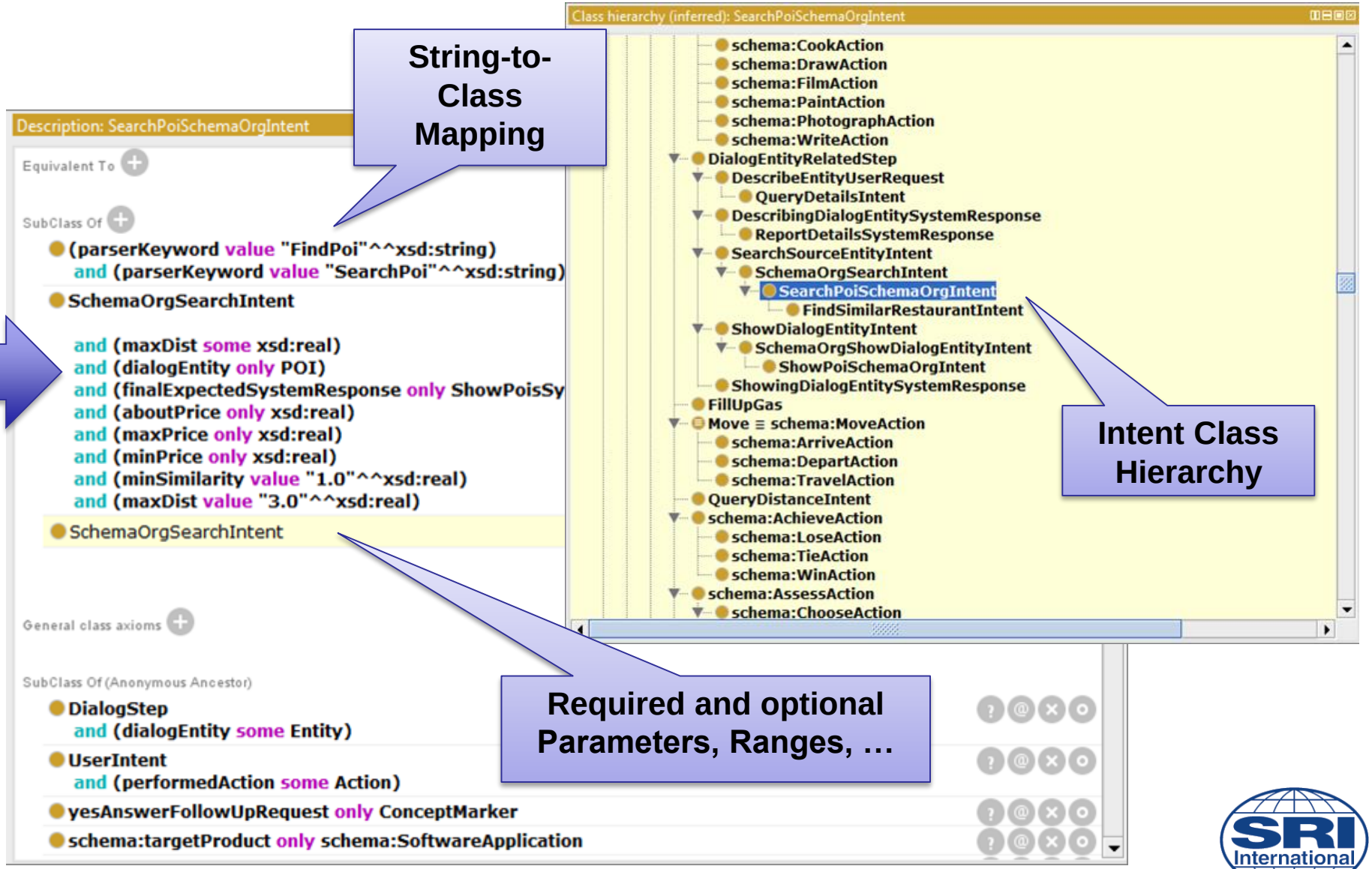
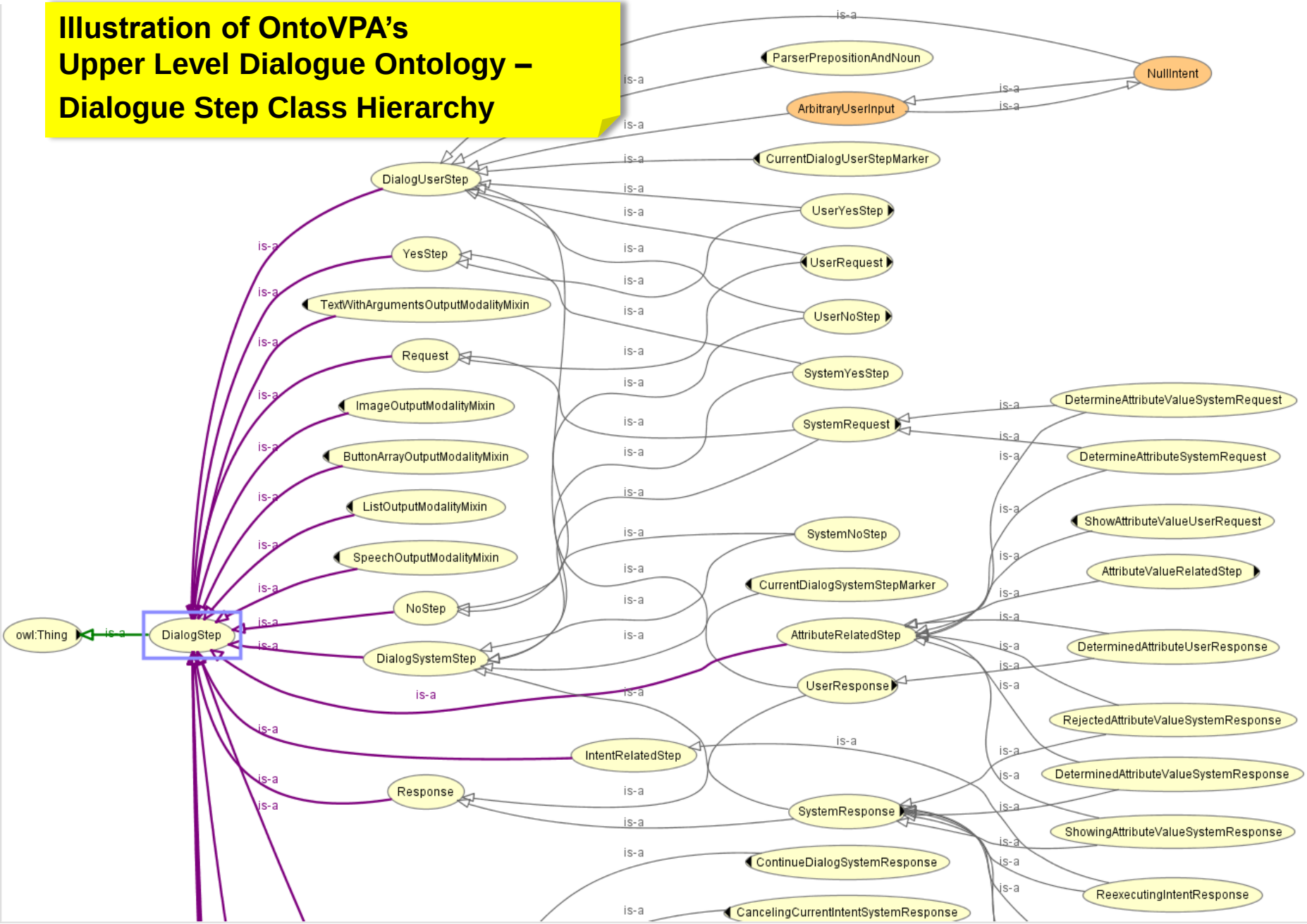


Illustration of OntoVPA's Upper Level Dialogue Ontology – Dialogue Step Class Hierarchy



Structure of OntoVPA Ontology

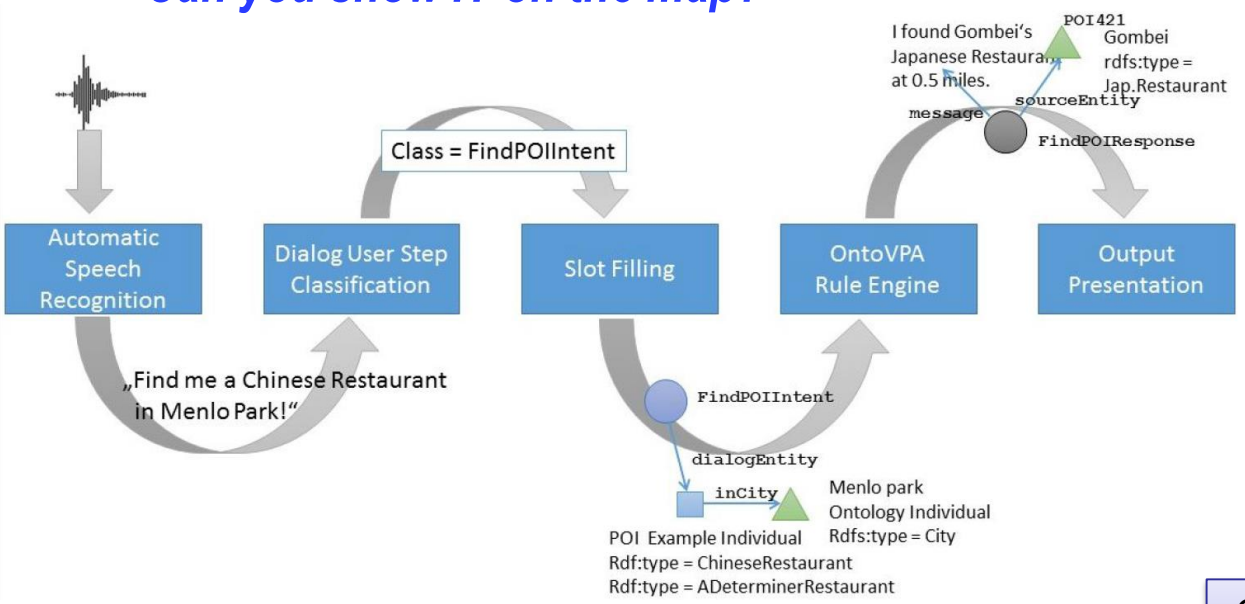
- **Domain Ontology (Static)**
  - Domain Vocabulary („TBox“)
    - Classes: POI, Restaurant, ChineseRestaurant, City, ...
    - Relations: inCity, nearBy, name, hasAddress, ...
  - Domain „Database“ („ABox“)
    - Instances & relationships: restaurant123, inCity(restaurant123, menloPark), ...
- **Dialogue Ontology (Static)**
  - Speech Act Theory Inspired
    - Request, Response, UserRequest, UserResponse, SystemRequest, SystemResponse, Yes/No, Greeting, ...
    - UserIntents are special UserRequests: FindPOIIntent
- **Actual Ontology-Based Runtime Dialogue Representation (Dynamic)**
  - Dialogue ABox as a „dialogue history graph“
  - Dynamically updated by SPARQL rules

OWL2 offers

- **Multiple inheritance for classes & properties (relations)**
- **Multiple types per individual**
- **Semi-structured “data” representation**
- **Optional & required properties / parameters**

Dialogue Dynamics – Runtime Dialogue Representation

Is there a Japanese Restaurant in Menlo Park?
Can you show IT on the map?



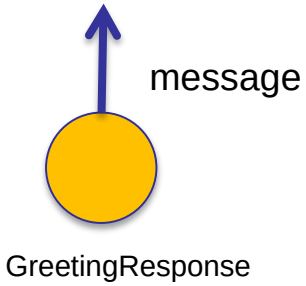
1. **Statistical intent classification -> FindPoiIntent**
2. **Parameter extraction & „slot filling“**
   dialogueEntity = { type : ChineseRestaurant,
   inCity : menloPark }
3. **Intent is asserted into dialogue engine**
4. **Dialogue engine: more interpretation, reasoning, output gen.**

Query-By-Example
Exemplar POI

Example Dialogue – Dialogue Representation – 1 / 5

VPA

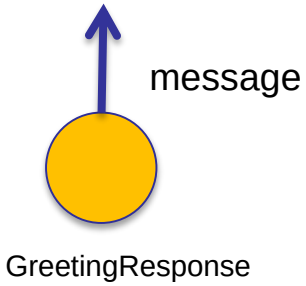
Hello, what can I do for you?

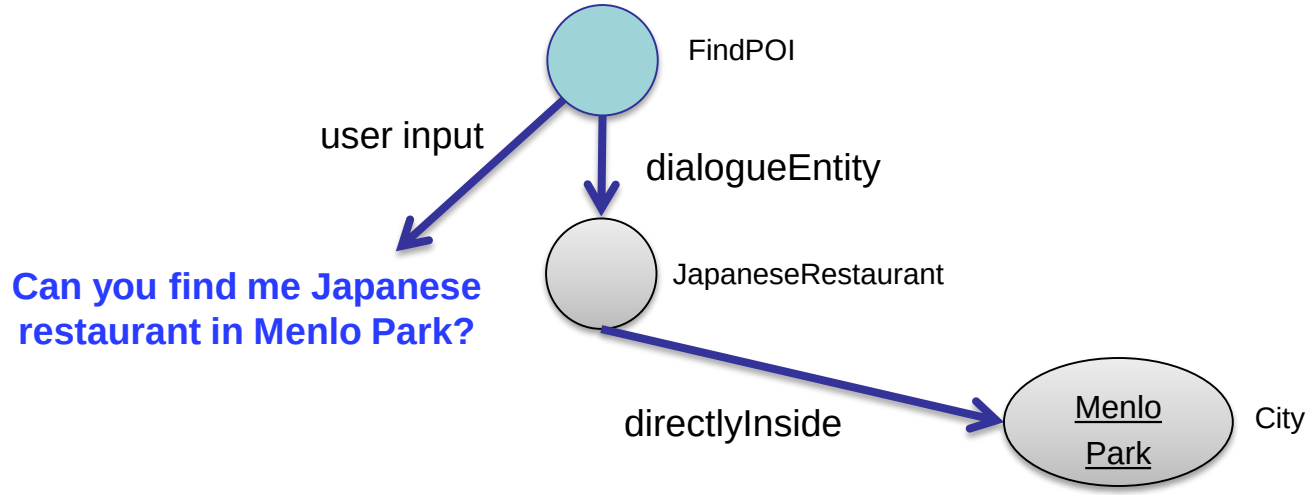


USER

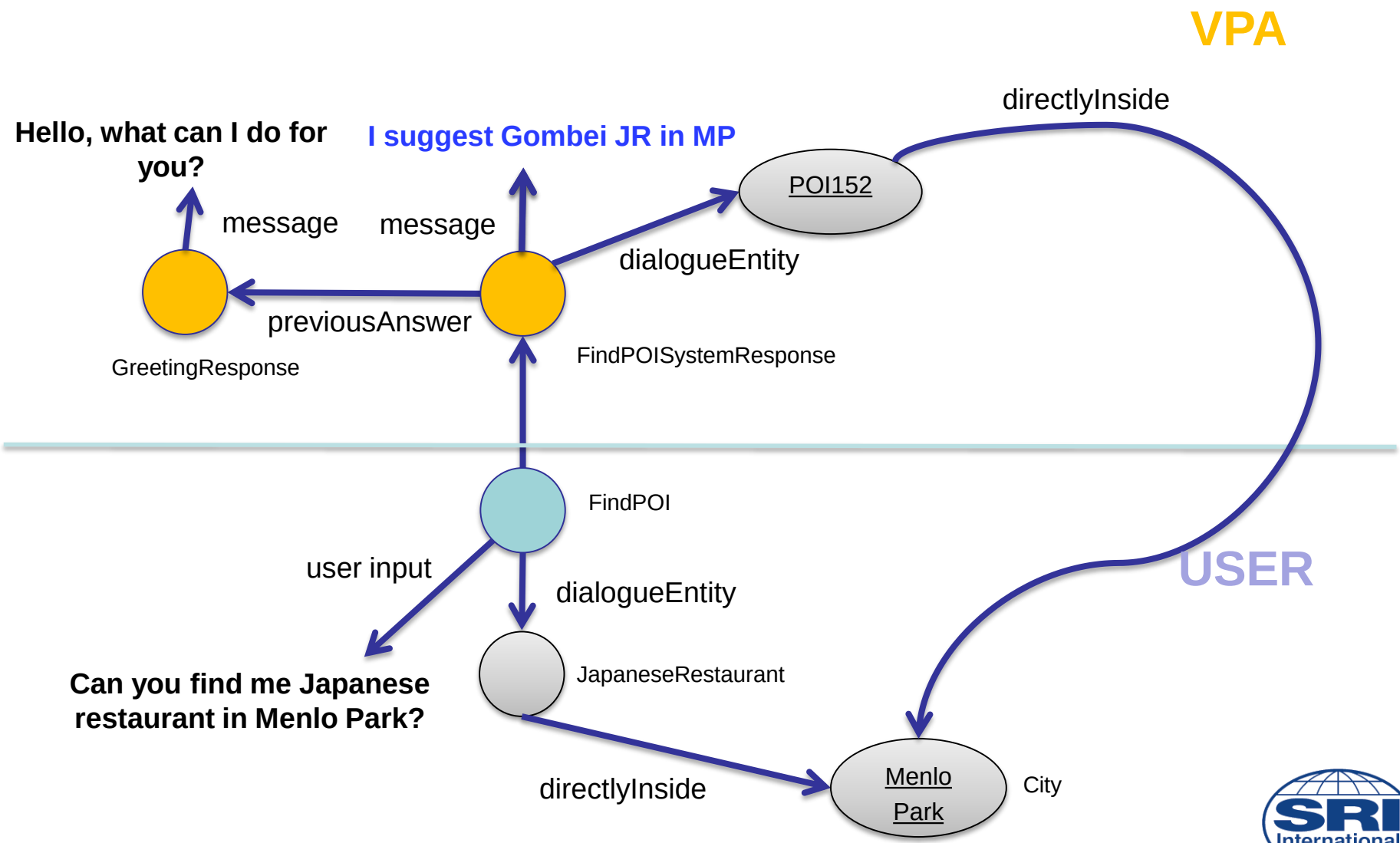
Example Dialogue – Dialogue Representation – 2 / 5

Hello, what can I do for you?

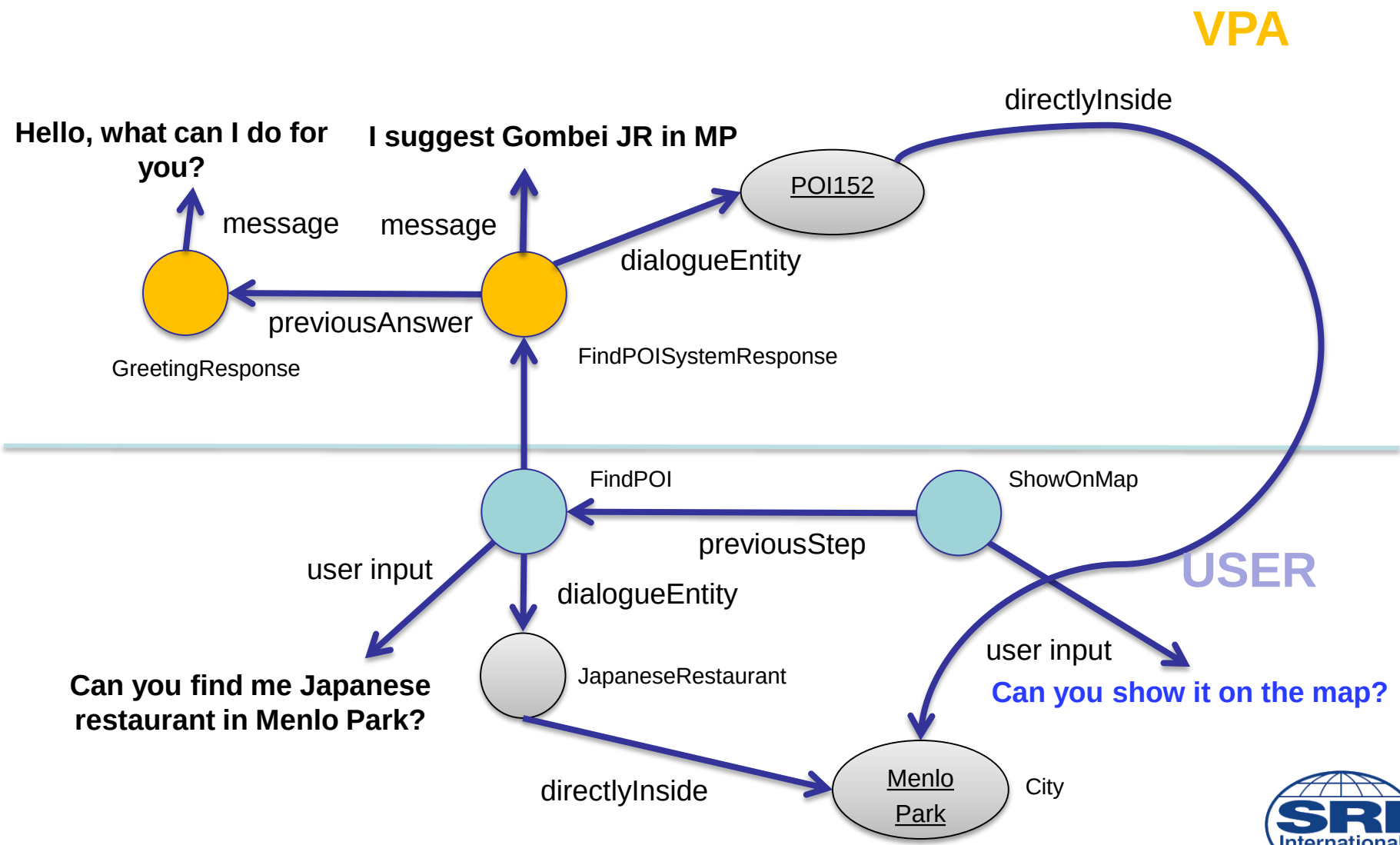




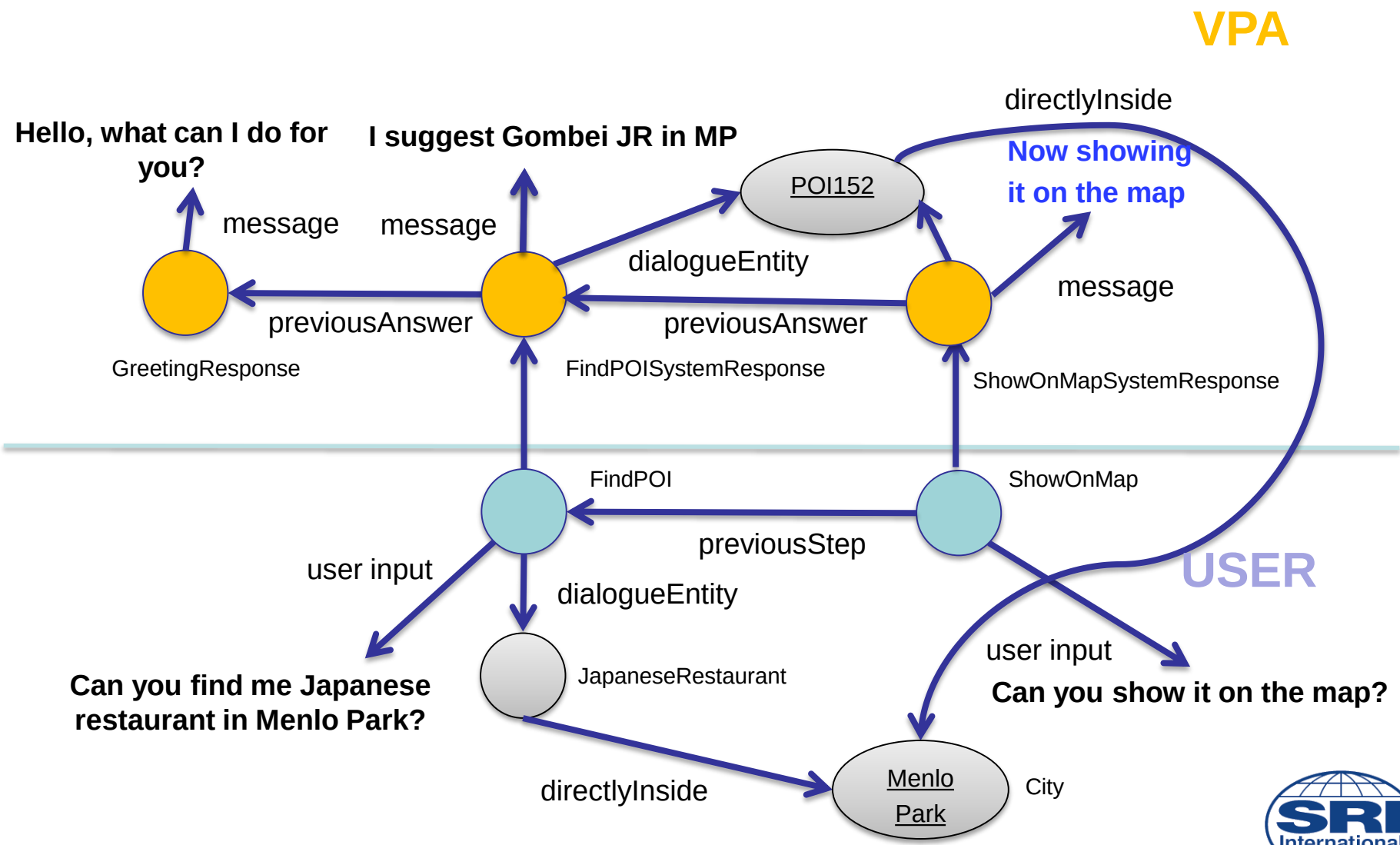
Example Dialogue – Dialogue Representation – 3 / 5



Example Dialogue – Dialogue Representation – 4 / 5



Example Dialogue – Dialogue Representation – 5 / 5



Structure of OntoVPA Ontology

- **Domain Ontology (Static)**
  - Domain Vocabulary („TBox“)
    - Classes: POI, Restaurant, ChineseRestaurant, City, ...
    - Relations: inCity, nearBy, name, hasAddress, ...
  - Domain „Database“ („ABox“)
    - Instances & relationships: restaurant123, inCity(restaurant123, menloPark), ...
- **Dialogue Ontology (Static)**
  - Speech Act Theory Inspired
    - Request, Response, UserRequest, UserResponse, SystemRequest, SystemResponse, Yes/No, Greeting, ...
    - UserIntents are special UserRequests: FindPOIIntent
- **Actual Runtime Dialogue Representation (Dynamic)**
  - Dialogue ABox as a „dialogue history graph“
  - Dynamically updated by SPARQL rules
- **Upper Level Ontologies & Generic Rule Layer**

OWL2 offers

- **Multiple inheritance for classes & properties (relations)**
- **Multiple types per individual**
- **Semi-structured “data” representation**
- **Optional & required properties / parameters**

OntoVPA Rule Engine (based on Apache Jena)

- **Why SPARQL? SPARQL is a RDF(s) query language…**
  - However, due to CONSTRUCT, it is also a “one rule” rule language
  - Can construct consequence triples
  - Add rule application strategy => rule engine
- **DMS-specific, but generic rule engine in OntoVPA**
  - Defeasibility / conflict resolution
  - Hierarchical layers of rule (augmentation, dialogue, recover)
  - Rules can fire or disable other rules, or loop, …
- **Useful features of SPARQL**
  - `CONSTRUCT`
  - User-defined SPARQL functions (interface to Java world)
  - “Second-order like” expressivity
  - Fast, W3C standard, mature implementations
  - Aware of ontology-consequences, customizable ***(Entailment Regimes)***

Query By Example Search – SPARQL Expressivity

- Candidate POI from data source has to fulfil all requirements stated by the exemplar POI
- An source POI `?x` is a match if it has (at least) all the properties P that the exemplar `dialogueEntity` has

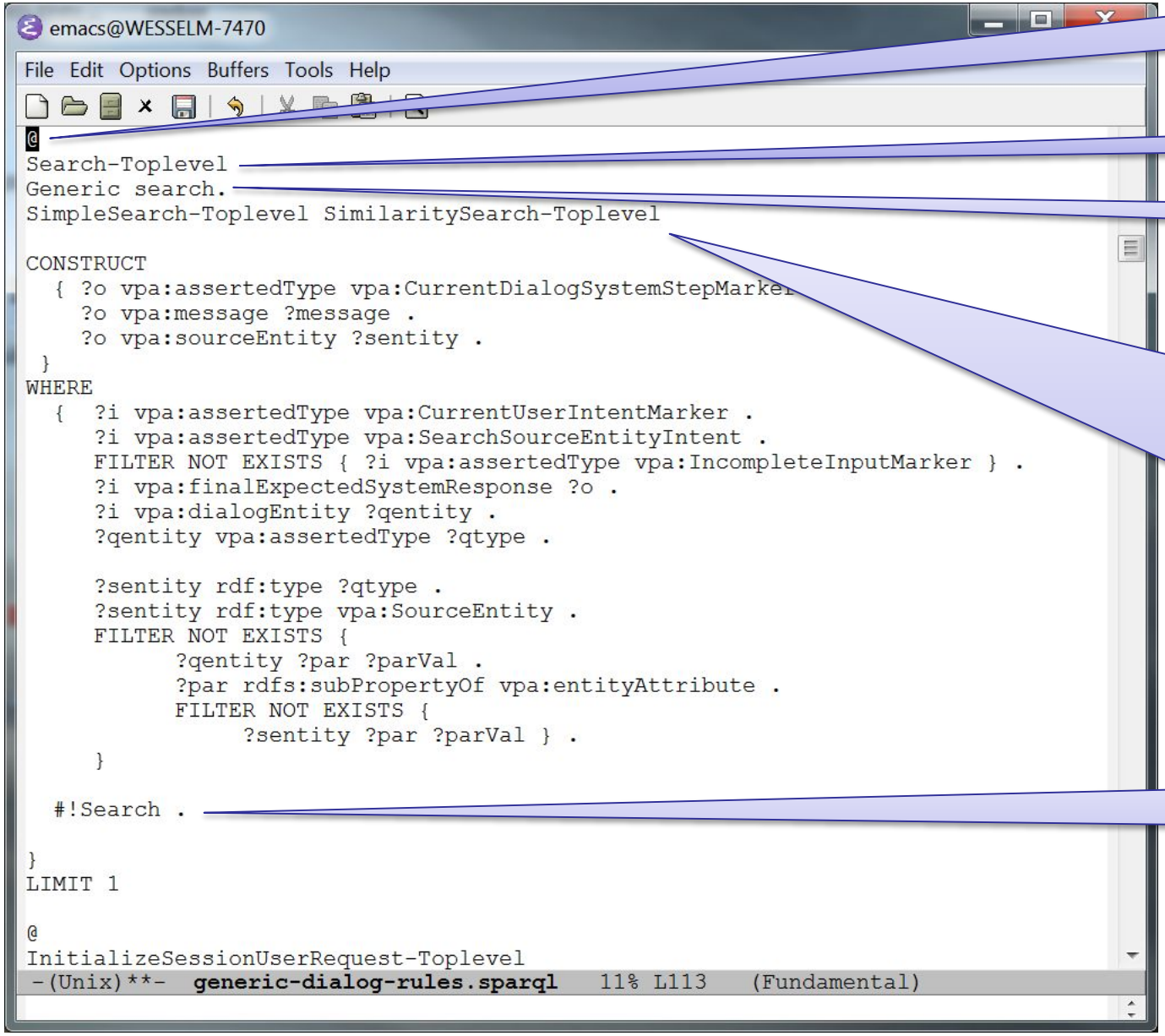
$$\forall P \forall val : P(dialogEntity, val) \rightarrow P(x, val)$$

- There is no property `?P` that `dialogueEntity` has, that `?x` does not have:

```
FILTER NOT EXISTS {
    dialogueEntity ?P ?val .
   FILTER NOT EXISTS {
       ?x ?P ?val
   }
}
```

Much of OntoVPA's generic behavior is specified using concise, expressive and generic "second-order like" rules in its upper level rule layer.

Search-Toplevel SPARQL Rule



Multiple rules in a .SPARQL file, separated by „@“

Rule Name

Documentation

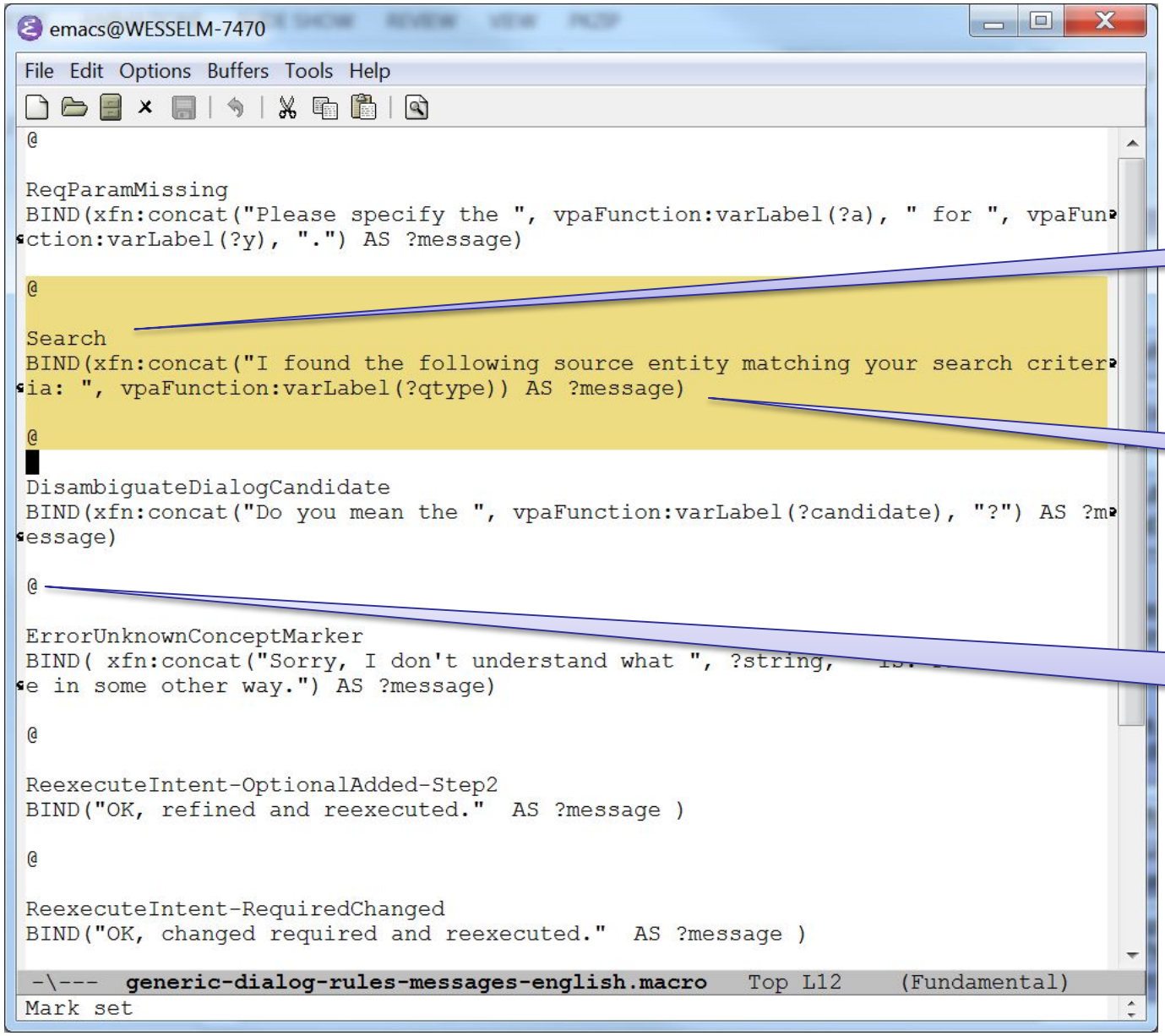
List of „defeated“ rules (inheritance)

Also numeric („priority“) values possible here.

Rules can also be disabled.

Our SPARQL Macros Internationalization!

Macro Files to Support Localization (here: English)



Macro Name

SPARQL Expression

Multiple macros in a .MACRO file, separated by „@“

Important SPARQL Features

- **`CONSTRUCT`**
  - blank nodes to create ans assert arbitrary (complex) new structure!
- **Universal and existential quantification, negation**
- **„Higher-order“ like quantification over properties (see search example)**
- **User-defined SPARQL functions**
- **`ASK` for quick rule applicability check**
- **Federated SPARQL queries (WikiData, DBPedia, ...)**
- **Human editable syntax (no XML)**

Important Apache Jena Features

- **Apache License**
- ***Fast* in-memory query engine**
  - Custom DMS-specific rule engine built on top:
    - check all `Toplevel ASK`s, select applicable (non-defeated), `CONSTRUCT`, ..
  - So far, OntoVPA is fast enough with up to a couple of 100 rules
- **Easy to write user-defined functions!**
  - `geoDistance`, `semanticSimilarity`, ...
- **Jena SPARQL is „ontology aware“ (Tbox axioms)**
  - Completeness (and performance!) of reasoning is configurable
  - `OwlMiniReasoner`, `OwlMicroReasoner`, ...
- **Support for federated SPARQL**
- **We encountered some serious bugs, but were able to fix them - thanks to OpenSource !**

Further Aspects of OntoVPA Implementation

- **Efficient Ontology representation for SPARQL queries?**
  - if an intent *misses* a required parameter, OntoVPA will ask about its value - it requires (expensive!) OPWL reasoning to figure that out!
  - an ***offline-compilation process*** rewrites the Tboxes and makes these inferences explicit ***(materialization)***
  - SPARQL can then read off the required info easily and very efficiently
- **Rules are organized in layers**
  - **Generic rule layer:** *preprocessing, agumentation, dialogue, recovery*
  - **Domain-specific rule layer:** *same*
  - VPA Init:
    - fire all applicable (generic & domain-specific) ***preprocessing rules***
  - For each dialog user step / intent („talk“):
    - fire all applicable (generic & domain-specific) ***augmentation rules*** exhaustively (defeasibility!)
    - fire one applicable ***dialogue rule*** (most specific usually, defeasibility!)
    - if none fired, fire one applicable ***recovery rule*** (most specific, ...)

Summary

- **Explicit, symbolic, declarative representations with formal semantics**
  - Full dialogue history, fully “reflexive”
  - Transparent, understandable, explainable, reliable
  - Data / knowledge driven
- **Multi-lingual, multi-domain, multi-modal**
- **Based on W3C® Standards (OWL, RDFS, SPARQL, …)**
  - Rich tool stack for authoring
  - Compatibility
  - No “vendor lock in” for customers
  - Less steep learning curve for customers and better acceptance (books)
- **Reusability & Core Functionalities**
  - Dialogue management specific upper-level ontologies & upper level rules
  - Ontologies (“off the shelf”) and services on the Web exploitable
  - SPARQL endpoints (federated SPARQL queries), Semantic Web Services, …

=> Flexible, Powerful, and Cost-Efficient VPA Development

Thank you!

Questions?