Dialogue Management with VOnDA

Bernd Kiefer // DFKI

June 20, 2018

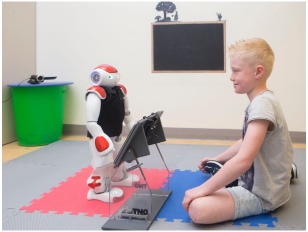
PAL

PERSONAL ASSISTANT FOR HEALTHY LIFESTYLE

HORIZON 2020

Co-funded by the Horizon 2020 Framework Programme of the European Union under grant agreement no. 643783

Talking Robots @ MLT



Dialogue Systems for Autonomous Agents

Scenario Requirements

- Delicate Application Areas
- User and Situation Adaptivity
- Long Term use / Multiple Sessions

Dialogue Systems for Autonomous Agents

Scenario Requirements

- Delicate Application Areas
- User and Situation Adaptivity
- Long Term use / Multiple Sessions

Application Requirements

- High reliability
- Long-Term Memory
- World knowledge / reasoning about the situation

What’s that to do with Delph-IN?

Reliability

- Grammaticality of NL generation
- Fine-grained NL analysis

What’s that to do with Delph-IN?

Reliability

- Grammaticality of NL generation
- Fine-grained NL analysis

Symbolic representations for reasoning

- NLU that delivers (general) semantic structures
- . . . and as input to generation

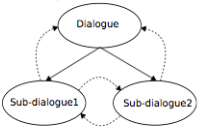
Approaches to Dialogue Management

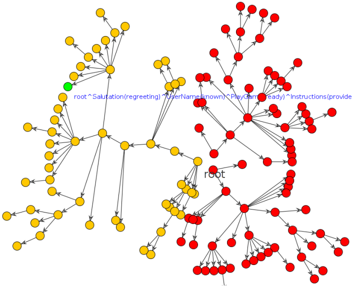
- (Hierarchical) State Machines
  - SceneMaker, DialogOS
  - + Easy to use, sufficient for many applications
  - – Limited scalability and flexibility, bad at generalization
- Machine Learning, mostly Hierarchical POMDP
  - PyDial
  - + Adaptive, flexible
  - – Hard to enforce behaviours or inhibit unwanted behaviour
- Rule / Reasoning Based
  - OpenDIAL, RavenClaw, VOnDA
  - + declarative, more flexible, generalization is easy, transparent reasoning
  - – dependencies between rules, scalability(?), harder to implement

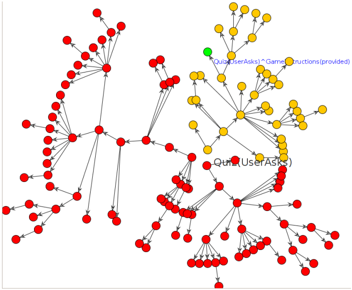
Dialogue Management with Bayes Nets

Flexible Hierarchical Control

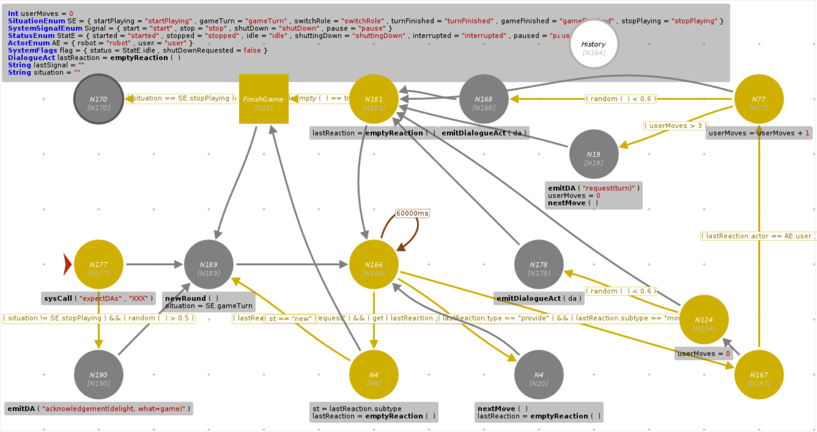
R: Do you want to ask first?
U: OK.
R: OK, you start, what is the first question?
U: What's the capital of Italy?




root^Salutation(regreeting)^UserName(known)^PlayGame(ready)^Instructions(provided)
root



State Charts



```
Int userMoves = 0
SituationEnum SE = { startPlaying = "startPlaying" , gameTurn = "gameTurn" , switchRole = "switchRole" , turnFinished = "turnFinished" , gameFinished = "gameFinished" , stopPlaying = "stopPlaying" }
SystemSignalEnum Signal = { start = "start" , stop = "stop" , shutDown = "shutDown" , pause = "pause" }
StatusEnum StatE = { started = "started" , stopped = "stopped" , idle = "idle" , shuttingDown = "shuttingDown" , interrupted = "interrupted" , paused = "paus
ActorEnum AE = { robot = "robot" , user = "user" }
SystemFlags flag = { status = StatE.idle , shutDownRequested = false }
DialogueAct lastReaction = emptyReaction ( )
String lastSignal = ""
String situation = ""
```

History
[N164]
N170
[N170]
(situation == SE.stopPlaying)
FinishGame
[S25]
mpty ( ) == tr
N181
[N181]
N168
[N168]
(random ( ) < 0.6)
N77
[N77]
lastReaction = emptyReaction ( )
emitDialogueAct (da)
(userMoves > 3)
userMoves = userMoves + 1
N19
[N19]
emitDA ("request(turn)")
userMoves = 0
nextMove ( )
60000ms
(lastReaction.actor == AE.user)
N177
[N177]
N189
[N189]
N166
[N166]
N178
[N178]
(random ( ) < 0.6)
sysCall ("expectDAs" , "XXX")
newRound ( )
situation = SE.gameTurn
emitDialogueAct (da)
(situation != SE.stopPlaying) && (random ( ) > 0.5)
(lastRea(st == "new")equest") && (get (lastReaction (lastReaction.type == "provide") && (lastReaction.subtype == "mov
N124
[N124]
userMoves = 0
N190
[N190]
N4
[N4]
N4
[N20]
N167
[N167]
emitDA ("acknowledgement(delight, what=game)")
st = lastReaction.subtype
lastReaction = emptyReaction ( )
nextMove ( )
lastReaction = emptyReaction ( )

Requirements in PAL

- Flexile dialogue strategies
- Predicable behaviour
- Long-term memory to be used in dialogue
- User-adaptive behaviour (dialogue/generation)

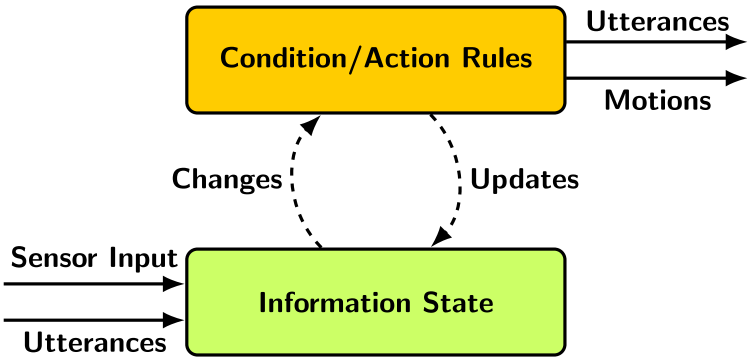
Requirements in PAL

- Flexile dialogue strategies
- Predicable behaviour
- Long-term memory to be used in dialogue
- User-adaptive behaviour (dialogue/generation)

$\Longrightarrow$

Rule-based approach with transparent access to the memory

Information State – Update



Design Decisions

Information State

- Favour ontologies over (unflexible) database schemata
- Tagging all information with time → memory
- Also: RDF objects can be used like Java objects
- Allow integration of arbitrary sensor data
- DL (and other) reasoning

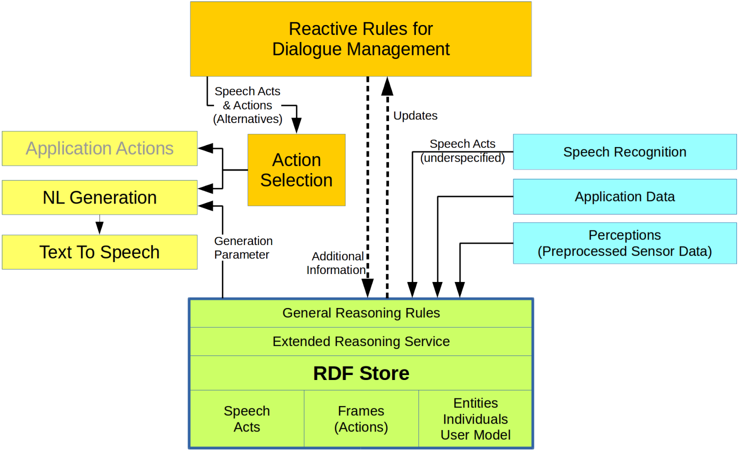
Rule Language

- Easy access to the database (with history)
- Concise specifications / short code
- Seamless integration of Java code

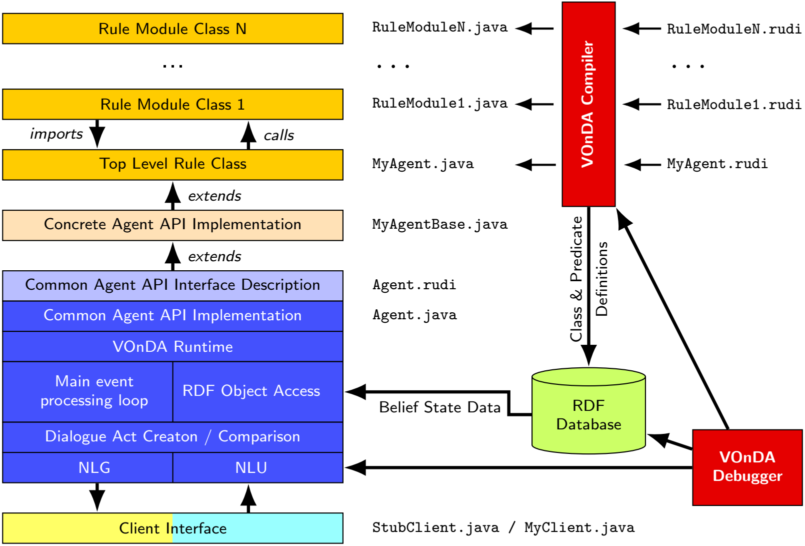
Uniform Representation on all Layers

- Favouring dynamic ontologies over unflexible database schemata
    - Easier to extend and change
    - Data structures of differing complexity
- Tagging all incoming and computed information with time → Going beyond RDF triples and standard entailment
- Automatically creates a history of events
- Makes it possible to use individuals as programming variables
- Rules that operate over time-stamped information drive the dialogue
- Alternative dialogue continuations are represented through *future branching time* (possible belief sets)

Architecture

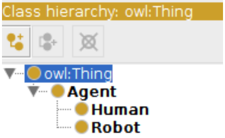


VOnDA Framework



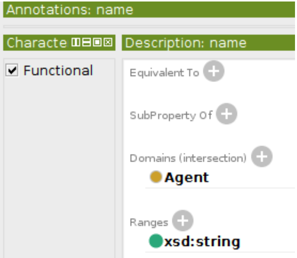
Ontology and Code

Ontology in Protégé



Class hierarchy: owl:Thing

- owl:Thing
  - Agent
    - Human
    - Robot



Annotations: name

Characte | Description: name

☑ Functional

Equivalent To

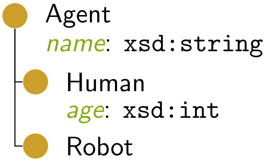
SubProperty Of

Domains (intersection)
Agent

Ranges
xsd:string

VOnDA Code and Ontology

```
user = new Human;
user.name = "Joe";
set_age:
if (user.age <= 0) {
  user.age = 15;
}
```



Agent
name: xsd:string

Human
age: xsd:int

Robot

Information State/Update

High-level programming language unifying rules, data access and temporal continuation

```
gameActive = Sorting
lastMove = {Actor:User, Correct=Yes, when=170}
lastSpeechact = {DialogueActType=Inform, Frame=Move,
                 Actor=Robot, when=180}
```

```
if (lastMove.Actor == user) doAction(nextMove)

if (lastMove.Actor == user
    && lastMove.when > lastDA().when
    && random() > .5) {
  emitDA(#Acknowledge(Move, Correct={lastMove.Correct}))
}
```

Rule Language: Central Aspects

- (Labeled) reactive rules, triggered by
  - incoming / changing data
  - timeouts
  - system events
- Organized in modules that can be reused
  - Rule modules are `imported` by others (any depth)
  - Variables are inherited to imported modules
  - Definition of functions (also inherited)
- Built-in timeouts (single / repeating):
  react to delays or silence
- Geared towards lean specifications

Rule Language II

- Special support for Dialogue Acts

```
forename = "John"
emitDA(#Inform(Name, value={forename}, sender={I_MYSELF}))

if (!da.value) da.value = forename
```

- Shortcuts for access to RDF objects
  - `user.forename = "John"`
  - `fullname = user.forename + " " + user.givenname`
  - `user.hasHobbies += Football`
  - `if (user.hasHobbies.contains((h) -> (h <= Football)) ...`

Rule Language III

- Types of variables or expressions are inferred where possible
  - Manual specification possible where necessary
  - Uses ontology for type inference of RDF objects / variables
  - Dialogue Acts are backed by ontology: Frame / argument checking
- Functional expressions
  - Java-like: `(h) -> (h.isFilled())`
  - to be used with `contains`, `all`, `filter`, `sort`

Rule Language IV

- Overloaded operators , e.g., <=
  - "ordinary" interpretation for Java data types
  - subsumption of semantic structures
  - subclass operation for RDF classes
- To end normal rule processing:
  - labeled `return` statements
  - `cancel` (local) and `cancel_all` (global)
- Seamless use of Java objects and methods

Rule Example I

```
interpretation_underspecification:
if ((myLastDA() <= #Request(top) || myLastDA() <= #YNQuestion(top))
     && (lastDA() <= #Confirm(top) || lastDA() <= #Disconfirm(top)))
    || (myLastDA() <= #WHQuestion(top) && lastDA() <= #Inform(top)) {

  // there is no explicit reference, fill it
  if (! lastDA().refersTo)
    lastDA().refersTo = myLastDA().id;

  // the topic is completely underspecified
  if (lastDA().getProposition() == top)
    lastDA().setProposition(myLastDA().getProposition());

  if (! lastDA().addressee)
    lastDA().addressee = myLastDA().sender;
}
```

Rule Example

A pending task with missing information, which is provided now.

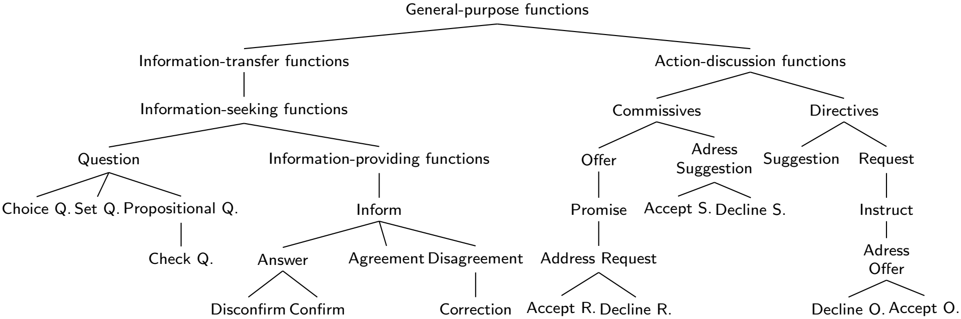
```
task_fill_argument:
if ((lastDA()<= #Inform({pendingTask.Frame})
    || lastDA() <= #Confirm({pendingTask.Frame})) {
    for (pair : lastDA().getSlots()) {
      if (!pendingTask.pair.arg) {
        pendingTask.pair.arg = pair.val;
      }
    }
    if (isFullySpecified(pendingTask)) {
      createTask(pendingTask);
      // possibly inform that the task will now be executed
      emitDA(#Inform({pendingTask.Frame}));
      pendingTask = null;
    }
  }
}
```

Rule Processing

- Fix-point computation of proposed actions (closures)
- Statistical module for selection of most appropriate action
- Support for synchronization with end of text-to-speech and / or motion for generated dialogue actions
- Detailed logging of rule conditions
  - all atomic parts of the boolean expression are logged
  - dynamic per-rule selection (by rule name)
- More debugging tools planned (dependency analysis, etc.)

Interfacing NL Components

- Goal: declarative high-level specification of possible *things-to-say* as parameterised dialogue acts
- Layer One: Taxonomy of dialogue acts along DIT++



General-purpose functions

Information-transfer functions

Action-discussion functions

Information-seeking functions

Commissives

Directives

Question

Information-providing functions

Offer

Adress Suggestion

Suggestion

Request

Choice Q.

Set Q.

Propositional Q.

Inform

Promise

Accept S.

Decline S.

Instruct

Check Q.

Answer

Agreement

Disagreement

Address Request

Adress Offer

Disconfirm

Confirm

Correction

Accept R.

Decline R.

Decline O.

Accept O.

Additional Parameters beyond Speech Acts

Employing FrameNet frames in shallow semantics

A: Can I offer you some coffee and chocolates?

```
offer(give, theme=coffee_and_chocolate, sender=I, ...)
```

B: Only coffee please.

```
acceptOffer(give, theme=coffee, ...)
```

Additional parameterisation from information state

- User model
- Sensor data
- dialogue history

Information Context

Data used by multi-modal processing

- User Model Information (including emotional state) for personalization
- Dialogue history (also across sessions), authored content, etc. for long-term interaction
- Updated during dialogue, text and other processing

Making it accessible

- Declarative specification as RDF subgraphs or queries
- Used for parameterising the (non)verbal generation
- Can be used for automated coverage tests
- Specification describes what is in the user model, long term memory, etc.

VOnDA Approach: Pros

- + Declarative!
- + Uniform representation and access to knowledge
- + Easier to generalize over different dialogue situations
- + Easier to create more flexible dialogues
- + Open to meta-reasoning
- + Better modularization and reusability
- + Self-Introspection and explanation of behaviour

VOnDA Approach: Potential Cons

- Hard to keep track of the dependencies between rules
- Rule sets might get quite big for large systems
  The same is true for state charts → Break-even point?
- Concept might be harder to grasp for unexperienced users

- ▸ Will be addressed by appropriate development tools
- ▸ Static and dynamic analysis of rules and behaviour
- ▸ Recorded history may help pinpointing problems

Try it out!

It's now an open-source project on github

`https://github.com/bkiefer/vonda`

Any comments welcome!

References

- Patrick Gebhard, Gregor Mehlmann, and Michael Kipp (2012). *Visual SceneMaker—a tool for authoring interactive virtual characters.* Journal on Multimodal User Interfaces, 6(1-2):3–11.
- H.-U. Krieger & G.-J. Kruijff (2011). *Combining Uncertainty and Description Logic Rule-Based Reasoning in Situation-Aware Robots.* Proceedings of the AAAI 2011 Spring Symposium on "Logical Formalizations of Commonsense Reasoning.
- I. Kruijff-Korbayová, G. Athanasopoulos, A. Beck, P. Cosi, H. Cuayahuitl, T. Dekens, V. Enescu, A. Hiolle, B. Kiefer, H. Sahli, M. Schröder, G. Sommavilla, F. Tesser, W. Verhelst (2011) *An event-based conversational system for the Nao robot* Workshop on Paralinguistic Information and its Integration in Spoken Dialogue Systems
- H.-U. Krieger (2012). *A Temporal Extension of the Hayes/ter Horst Entailment Rules and an Alternative to W3C's N-ary Relations.* Proc. of the 7th Int. Conf. on Formal Ontology in Information Systems (FOIS).
- H.-U. Krieger (2013). *An Efficient Implementation of Equivalence Relations in OWL via Rule and Query Rewriting .* Proceedings of the 7th IEEE International Conference on Semantic Computing (ICSC).

References

- Ivana Kruijff-Korbayová, Elettra Oleari, Ilaria Baroni, Bernd Kiefer, Mattia Coti Zelati, Clara Pozzi, Alberto Sanna (2014) *Effects of Off-Activity Talk in Human-Robot Interaction with Diabetic Children.* Ro-Man 2014
- H.-U. Krieger (2014). *A Detailed Comparison of Seven Approaches for the Annotation of Time-Dependent Factual Knowledge in RDF and OWL.* Proceedings of the 10th Joint ACL-ISO Workshop on Interoperable Semantic Annotation.
- Pierre Lison and Casey Kennington (2015). *Developing spoken dialogue systems with the OpenDial toolkit.* SEMDIAL 2015.
- Stefan Ultes, Lina M Rojas Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Inigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gasic, et al. (2017). *Pydial: A multi-domain statistical dialogue system toolkit.* Proceedings of ACL 2017, System Demonstrations.