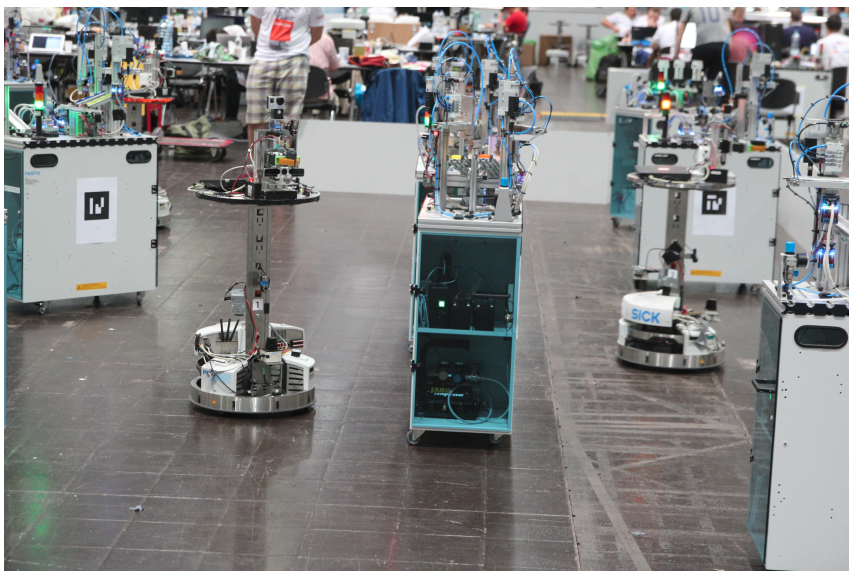
A Document-Oriented Robot Memory for Knowledge Sharing and Hybrid Reasoning on Mobile Robots

Frederik Zwilling
Adviser: Tim Niemueller
Supervisors: Prof. Lakemeyer, PhD., Prof. Dr. Jarke

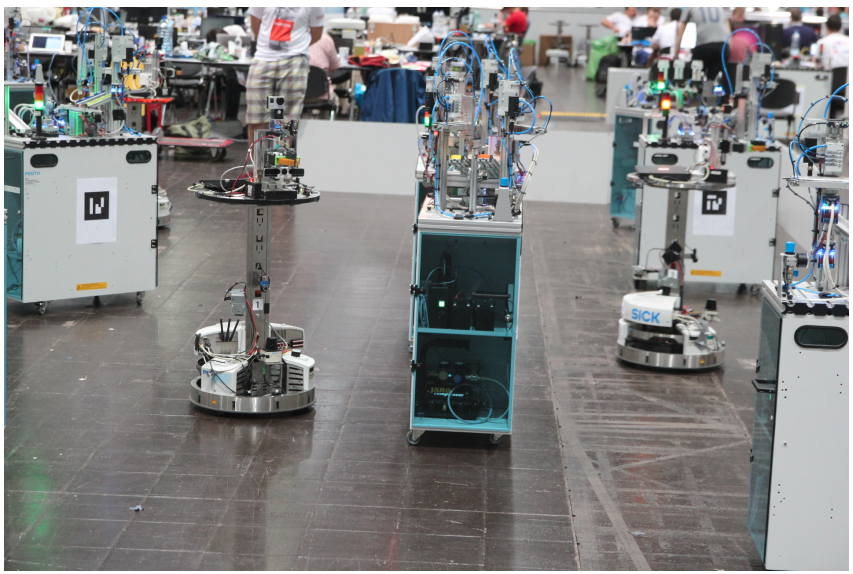
Motivation

Why do robots need a memory?



Motivation

Why do robots need a memory?



Robot Memory: Component for knowledge storing and querying

Problem Description - Proposed Solution

Problems of existing approaches

- Static data structure
- Memory intertwined in single Knowledge Based System
- Locality
- Multiple state estimations, inconsistencies
- Spatio-Temporal vs. Symbolic
- Not scalable, efficient
- Not persistent

Problem Description - Proposed Solution

Problems of existing approaches

- Static data structure
- Memory intertwined in single Knowledge Based System
- Locality
- Multiple state estimations, inconsistencies
- Spatio-Temporal vs. Symbolic
- Not scalable, efficient
- Not persistent

Proposed Robot Memory

- Generic representation
- Decoupled Memory from sources and consumers
- Distributed for robot teams
- Consistent storage

- On demand computation
- Scalable basis
- Persistent

Related Work: Knowledge Processing Systems

KnowRob/OpenRobots Ontology (ORO)

- Common sense reasoning with ontologies
- Virtual knowledge base to interface perception/ Events notifying about changes
- Based on Prolog/Java

Related Work: Knowledge Processing Systems

KnowRob/OpenRobots Ontology (ORO)

- Common sense reasoning with ontologies
- Virtual knowledge base to interface perception/ Events notifying about changes
- Based on Prolog/Java

- Not applicable for multi-robot systems
- Scalability, efficiency concerns
- Missing events/knowledge computation on demand

Related Work: Knowledge Processing Systems

KnowRob/OpenRobots Ontology (ORO)

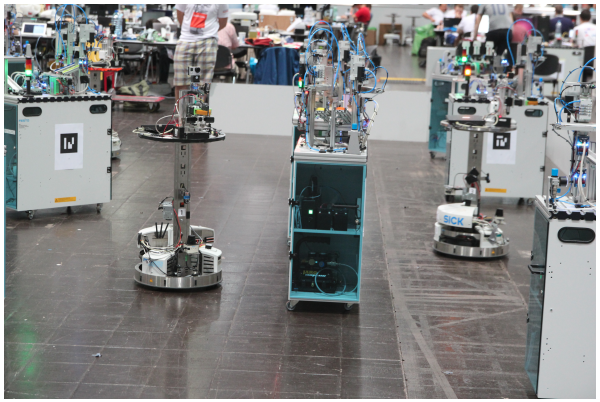
- Common sense reasoning with ontologies
- Virtual knowledge base to interface perception/ Events notifying about changes
- Based on Prolog/Java

- Not applicable for multi-robot systems
- Scalability, efficiency concerns
- Missing events/knowledge computation on demand

Generic Robot Database with MongoDB

- Data logging for evaluation and fault analysis
- Generic and scalable storage with MongoDB
- Integration in Fawkes and ROS

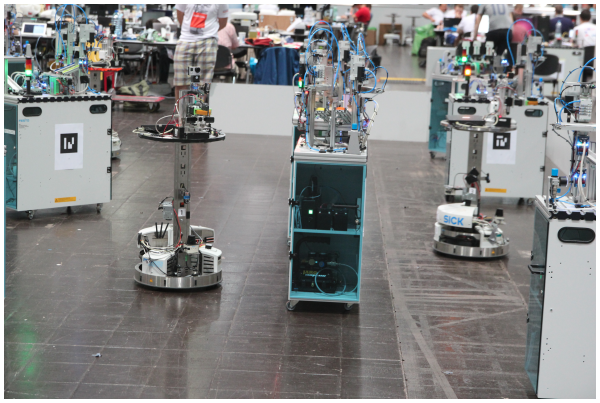
Background: Application Domains



RoboCup Logistics League

- Production logistics in smart factory
- Share world model
  - between robots
  - between global planner and reasoner executive

Background: Application Domains



RoboCup Logistics League

- Production logistics in smart factory
- Share world model
  - between robots
  - between global planner and reasoner executive

RoboCup@Home

- Domestic service robots
- Collect knowledge about concrete environment
- Hybrid reasoning with spatio-temporal knowledge
- Persistent storage

Background: Planners and Reasoners

CLIPS Rules Engine

- First-Order Logic forward chaining systems
- Fact base and condition-action rules

$\Rightarrow$ Good for world model reasoning and execution monitoring

Background: Planners and Reasoners

CLIPS Rules Engine

- First-Order Logic forward chaining systems
- Fact base and condition-action rules

$\Rightarrow$ Good for world model reasoning and execution monitoring

Planning Domain Definition Language (PDDL)

- Standardized language for planning problems
- Find action sequence through heuristic search

$\Rightarrow$ Good for finding global plans

Background: Planners and Reasoners

CLIPS Rules Engine

- First-Order Logic forward chaining systems
- Fact base and condition-action rules

$\Rightarrow$ Good for world model reasoning and execution monitoring

Planning Domain Definition Language (PDDL)

- Standardized language for planning problems
- Find action sequence through heuristic search

$\Rightarrow$ Good for finding global plans

Motion Planners

- Robot arm, locomotion collision avoidance
- Depend on geometric data

Background: Fawkes

Fawkes

- Robot Software Framework
- Component-based software design
- Blackboard communication infrastructure with specific interfaces

$\Rightarrow$ Has no generic memory for knowledge sharing

Background: MongoDB

- Document-oriented database
- Scalable, powerful and widely used
- Generic, schema-less data structure with key-value pairs

```
{
  type: "position",
  name: "robot1",
  translation: {x:2.5, y:1.0, z:0.0},
  rotation: {x:0.0, y:0.0, z:0.0, w:1.0},
  timestamp : ISODate("2016-05-19T15:26:34")
}
```

```
db.positions.find(
  {
    type: "position",
    name: "robot1",
    timestamp : {"$gt":
      ISODate("2016-05-19T15:26:34")}
  })
```

Background: MongoDB

- Document-oriented database
- Scalable, powerful and widely used
- Generic, schema-less data structure with key-value pairs

```
{
  type: "position",
  name: "robot1",
  translation: {x:2.5, y:1.0, z:0.0},
  rotation: {x:0.0, y:0.0, z:0.0, w:1.0},
  timestamp : ISODate("2016-05-19T15:26:34")
}
```

```
db.positions.find(
  {
    type: "position",
    name: "robot1",
    timestamp : {"$gt":
      ISODate("2016-05-19T15:26:34")}
  })
```

- Distributable with Replica Sets
- Expressive querying with JavaScript, MapReduce

$\Rightarrow$ Allows generic, scalable robot memory

Goals

What should the robot memory be capable of?

- Generic storage and retrieval
- Knowledge sharing between knowledge based systems
- *Computables:* Knowledge computation on demand for hybrid reasoning
- Shared knowledge for multi-robot systems
- *Event Triggers:* Update notifications
- Persistent storage

Theoretical Foundation

Definition of documents representing knowledge

e.g. $\{(\text{"}\textit{object}\text{"}, \text{"}\textit{cup}\text{"}), (\text{"}\textit{room}\text{"}, \text{"}\textit{kitchen}\text{"}), (\text{"}\textit{position}\text{"}, \{(\text{"}x\text{"}, 8), (\text{"}y\text{"}, 4)\})\}$

Theoretical Foundation

Definition of documents representing knowledge

e.g. $\{(\text{"}\mathit{object}\text{"}, \text{"}\mathit{cup}\text{"}), (\text{"}\mathit{room}\text{"}, \text{"}\mathit{kitchen}\text{"}), (\text{"}\mathit{position}\text{"}, \{(\text{"}x\text{"}, 8), (\text{"}y\text{"}, 4)\})\}$

1. **Keys:** $\mathcal{K} := \Sigma^*$
2. **Atomic values:** $\mathcal{V}_0$ are constants
3. **Unnested key-value pairs:** $\mathcal{P}_0 := \mathcal{K} \times \mathcal{V}_0$
4. **Unnested documents:**
$$\mathcal{D}_0 := \{d \in \mathbb{P}(\mathcal{P}_0) | \forall (k, v), (k', v') \in d, k \neq k' \vee (k, v) = (k', v')\}$$

With nesting

1. **Values:** $\mathcal{V}_n := \mathcal{V}_{n-1} \cup \mathcal{D}_{n-1}$
2. **Key-Value Pairs:** $\mathcal{P}_n := \mathcal{K} \times \mathcal{V}_n$
3. **Documents:**
$$\mathcal{D}_n := \{d \in \mathbb{P}(\mathcal{P}_n) | \forall (k, v), (k', v') \in d, k \neq k' \vee (k, v) = (k', v')\}$$

Theoretical Foundation

Definition of documents representing knowledge

e.g. $\{(\text{"}object\text{"}, \text{"}cup\text{"}), (\text{"}room\text{"}, \text{"}kitchen\text{"}), (\text{"}position\text{"}, \{(\text{"}x\text{"}, 8), (\text{"}y\text{"}, 4)\})\}$

1. **Keys:** $\mathcal{K} := \Sigma^*$
2. **Atomic values:** $\mathcal{V}_0$ are constants
3. **Unnested key-value pairs:** $\mathcal{P}_0 := \mathcal{K} \times \mathcal{V}_0$
4. **Unnested documents:**
$$\mathcal{D}_0 := \{d \in \mathbb{P}(\mathcal{P}_0) | \forall (k, v), (k', v') \in d, k \neq k' \vee (k, v) = (k', v')\}$$

With nesting

1. **Values:** $\mathcal{V}_n := \mathcal{V}_{n-1} \cup \mathcal{D}_{n-1}$
2. **Key-Value Pairs:** $\mathcal{P}_n := \mathcal{K} \times \mathcal{V}_n$
3. **Documents:**
$$\mathcal{D}_n := \{d \in \mathbb{P}(\mathcal{P}_n) | \forall (k, v), (k', v') \in d, k \neq k' \vee (k, v) = (k', v')\}$$

Finitely nested documents: $\mathcal{D} = \cup_{n \in \{0..b\}} \mathcal{D}_n$
Values: $\mathcal{V} = \cup_{n \in \{0..b\}} \mathcal{V}_n$, with nesting bound b

Definition Robot Memory

1. **Database:** finite set $\mathcal{DB} \subset \mathcal{D}$
2. **Query:** represented by document $q \in \mathcal{D}$
   yields set of documents $r \subset \mathcal{DB}$ as result
   e.g. $q = \{(\text{"}\mathit{object}\text{"}, \text{"}\mathit{cup}\text{"}), (\text{"}\mathit{room}\text{"}, \text{"}\mathit{kitchen}\text{"})\}$
3. **Computable:** $f : \mathcal{D} \to \mathbb{P}(\mathcal{D})$
4. **Set of Computables:** $\mathcal{C}$
5. **Robot Memory:** $\mathcal{RM} = (\mathcal{DB}, \mathcal{C})$
6. **Memorized Documents:** $mem(\mathcal{RM}) = \mathcal{DB} \cup \bigcup_{f \in \mathcal{C}} f(\mathcal{D})$

Mapping into PDDL

e.g. $map_p(\{(\text{"}predicate\text{"}, \text{"}at\text{"}), (\text{"}object\text{"}, \text{"}cup\text{"}), (\text{"}room\text{"}, \text{"}kitchen\text{"})\})$
$= at(map_f(\text{"}cup\text{"}), map_f(\text{"}kitchen\text{"})) = at(cup, kitchen)$.

Mapping into PDDL

e.g. $map_p(\{(\text{"}predicate\text{"}, \text{"}at\text{"}), (\text{"}object\text{"}, \text{"}cup\text{"}), (\text{"}room\text{"}, \text{"}kitchen\text{"})\})$
$= at(map_f(\text{"}cup\text{"}), map_f(\text{"}kitchen\text{"})) = at(cup, kitchen).$

1. **Predicate symbols $\mathcal{R}$, Function symbols $\mathcal{F}$**
2. **Name mapping:**

$$name_{pred} : \mathcal{R} \to \Sigma^* \text{ and } name_{func} : \mathcal{F} \to \Sigma^*$$
$$name_{pred-atr} : \mathcal{R} \times \mathbb{N} \to \mathcal{K} \text{ and } name_{func-atr} : \mathcal{F} \times \mathbb{N} \to \mathcal{K}$$

3. **Map to predicate:**

$map_p(d) = p(map_f(v_1), ..., map_f(v_n)),$ iff p is a n-array predicate in $\mathcal{R}$,
$(\text{"}predicate\text{"}, name_{pred}(p)) \in d, \forall i \in \{1..n\}(name_{pred-atr}(p, i), v_i) \in d$
$map_p(d) = nil_p, otherwise$

Mapping into PDDL

e.g. $map_p(\{(\text{"}predicate\text{"}, \text{"}at\text{"}), (\text{"}object\text{"}, \text{"}cup\text{"}), (\text{"}room\text{"}, \text{"}kitchen\text{"})\})$
$= at(map_f(\text{"}cup\text{"}), map_f(\text{"}kitchen\text{"})) = at(cup, kitchen).$

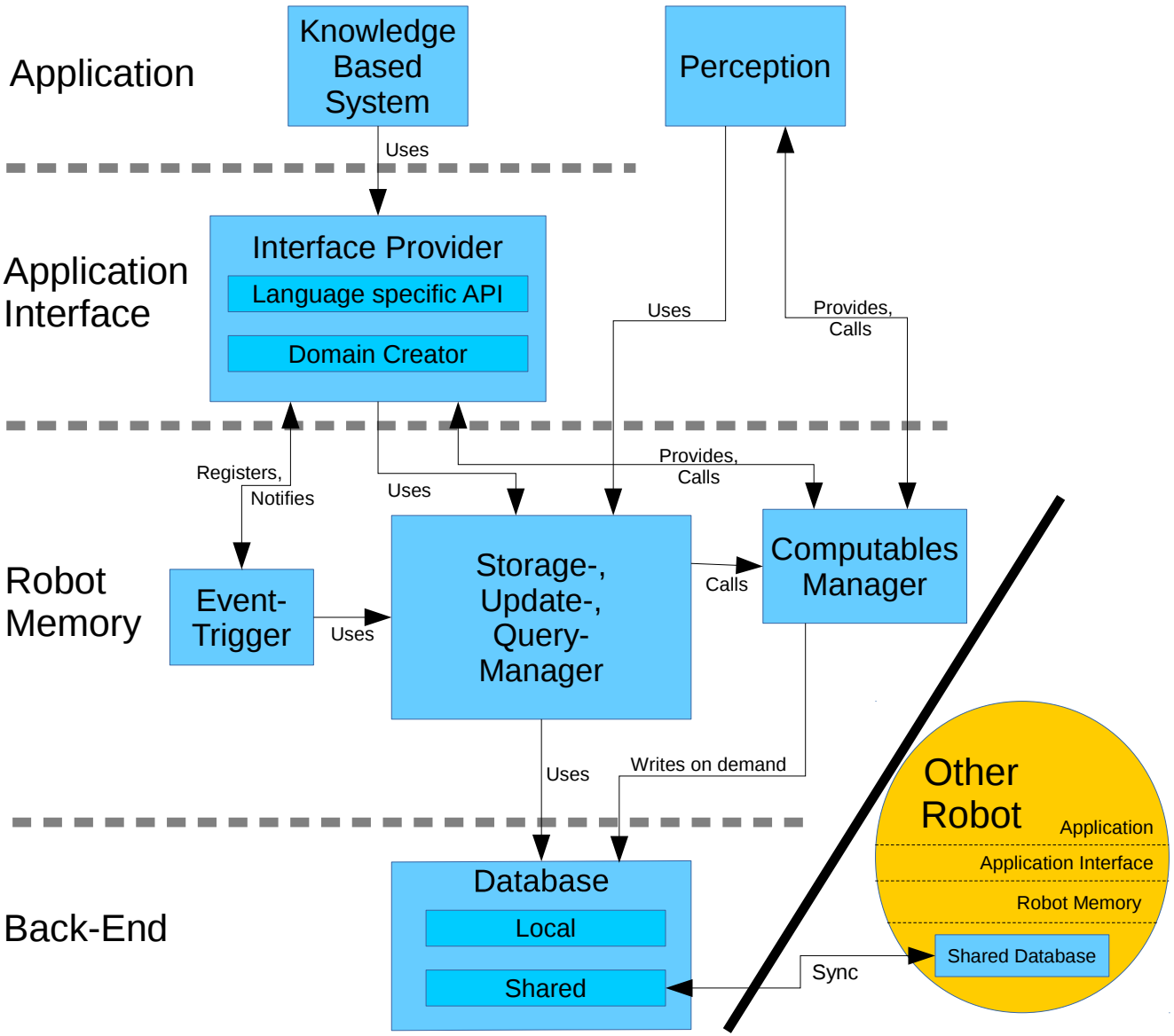
1. **Predicate symbols $\mathcal{R}$, Function symbols $\mathcal{F}$**
2. **Name mapping:**

$$name_{pred} : \mathcal{R} \to \Sigma^* \text{ and } name_{func} : \mathcal{F} \to \Sigma^*$$
$$name_{pred-atr} : \mathcal{R} \times \mathbb{N} \to \mathcal{K} \text{ and } name_{func-atr} : \mathcal{F} \times \mathbb{N} \to \mathcal{K}$$

3. **Map to predicate**
4. **Map to function term:**

$map_f(v) = v, v \in \mathcal{V}_0$
$map_f(d) = f(map_f(v_1), ..., map_f(v_n)),$ iff f is a n-array function in $\mathcal{F}$,
$(\text{"}function\text{"}, name_{func}(f)) \in d, \forall i \in \{1..n\}(name_{func-atr}(f, i), v_i) \in d$
$map_f(d) = nil_f, otherwise$

Architecture



Implementation

Back-End

- Realized with MongoDB
- Replication for multi-robot distribution

Implementation

Back-End

- Realized with MongoDB
- Replication for multi-robot distribution

Robot Memory

- Query enhancing, adding meta information (e.g. decay time)
- Dispatching raw DB queries and queries for computables
- Computables: forward query $\rightarrow$ compute results $\rightarrow$ cache results in DB $\rightarrow$ execute query on results
- Event-trigger with MongoDB Oplog

Implementation

Back-End

- Realized with MongoDB
- Replication for multi-robot distribution

Robot Memory

- Query enhancing, adding meta information (e.g. decay time)
- Dispatching raw DB queries and queries for computables
- Computables: forward query $\rightarrow$ compute results $\rightarrow$ cache results in DB $\rightarrow$ execute query on results
- Event-trigger with MongoDB Oplog

Planner/Reasoner

- Provide interface to robot memory (store/query features in CLIPS, domain creator in PDDL)

Evaluation

Prototype	Evaluation
• RCLL CLIPS agent with world model sync	• Representing and querying • Reasoner integration • Network throughput
• RCLL PDDL domain creation	• Planner integration • Knowledge Sharing Planner-Reasoner
• @Home Computables	• Knowledge Sharing Reasoner-Motion Planner • Hybrid Reasoning
• @Home Tidy up scenario	• Scalability • Expressiveness vs. Computation Speed

Conclusion and Questions

The generic robot memory is capable of efficiently storing and querying arbitrary symbolic or geometric information and allows consistent knowledge sharing.

- Document-oriented representation and querying
- Theoretical and architectural design
- Interfaces for knowledge based systems
- Computables for on demand knowledge computation
- RCLL and @Home as application and evaluation domains
- Foundation for future projects