Ontology for Robotics

Stefano Borgo

Laboratory for Applied Ontology
ISTC-CNR, Trento (IT)
http://www.loa.istc.cnr.it/

UTC, Sept 14, 2018

Table of Contents

Course Overview

First steps – 1

Leonardo Da Vinci (1452-1519) sketched many designs. One of them, drawn around 1495, was about a robot in the form of a medieval knight that could move its arms, head and open its jaws.

With the improvement of mechanics in 1700, a number of automatons ad automatic mechanisms started to appear. These automatons could draw, move, play music and even fly.

First steps – 1

Leonardo Da Vinci (1452-1519) sketched many designs. One of them, drawn around 1495, was about a robot in the form of a medieval knight that could move its arms, head and open its jaws.

With the improvement of mechanics in 1700, a number of automatons ad automatic mechanisms started to appear. These automatons could draw, move, play music and even fly.

The term automaton was the standard one until the publication of “Rossum’s Universal Robots” by Karel Capek (an influent book about replicants, not mechanical devices as we understand robots today) introduced the term **robot**. Robot comes from the Czech word “robota” which roughly means slave, forced labour.

First steps – 2

In 1956 business investor Joseph Engelberger and inventor George Devol started working together leading to the construciton of the Unimate, the very first industrial robot (a robotic arm).

First steps – 2

In 1956 business investor Joseph Engelberger and inventor George Devol started working together leading to the construciton of the Unimate, the very first industrial robot (a robotic arm).

Devol's patent for "Programmed Article Transfer" (1961) says:
The present invention relates to the automatic operation of machinery, particularly the handling apparatus, and to automatic control apparatus suited for such machinery. [wikipedia]

General Motors used Unimate in a die-casting plant.
Unimate undertook the job of transporting die castings from an assembly line and welding these parts on auto bodies, a dangerous task for workers due to toxic fumes and likely accidents.

First steps – 3

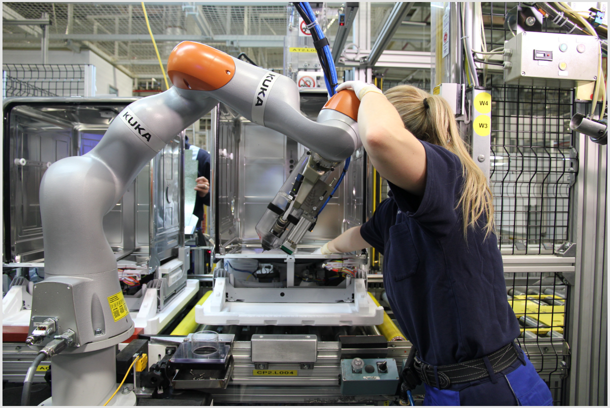
In those years, William Grey Walter constructed some of the first electronic autonomous robots. He wanted to prove that rich connections between a small number of brain cells could give rise to very complex behaviors.

First steps – 3

In those years, William Grey Walter constructed some of the first electronic autonomous robots. He wanted to prove that rich connections between a small number of brain cells could give rise to very complex behaviors.

A significant moment in robotics is when robots moved from the factory area to our everyday spaces.
Between 1966 and 1972 in Stanford a general-purpose mobile robot, called Shakey, was developed.
Shakey is the first robot able to reason about its own actions. It was the first project that integrated logical reasoning and physical action.

Scenario: Robot + Worker

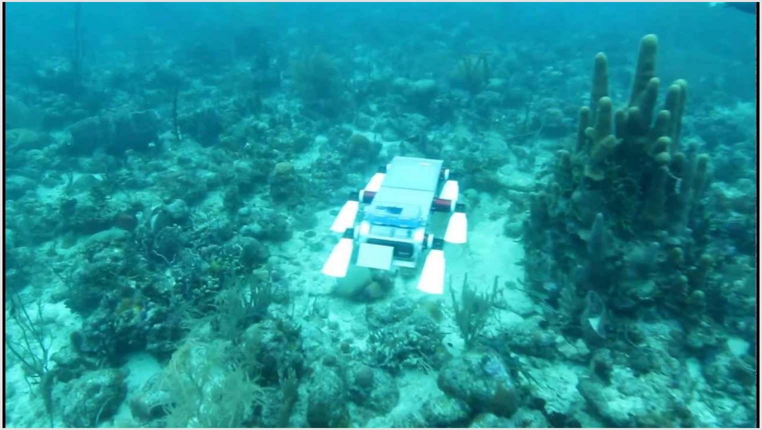


http://www.all-electronics.de

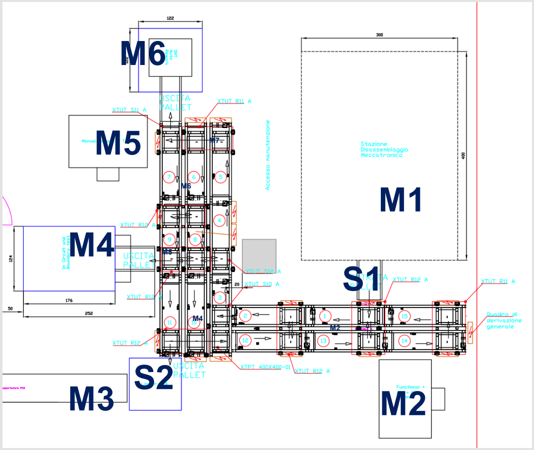
Scenario: Robot + Human

http://www.riken.jp

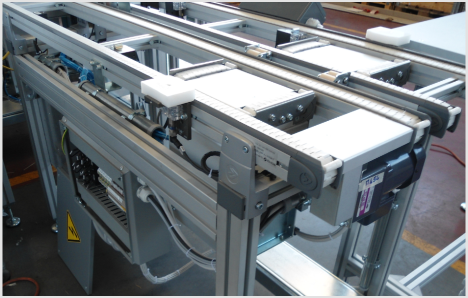
Scenario: Robot + Environment

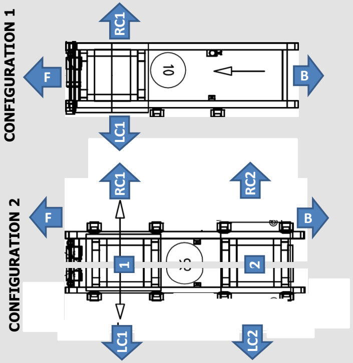

https://i.ytimg.com/vi/jC-AmPfInwU/maxresdefault.jpg

Scenario: Robot + Controlled environment



Scenario: Robot + Controlled environment /2





Agents

There are three prototypical types of (embodied) agents:

- human
- animal
- artificial

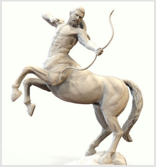
Agents

There are three prototypical types of (embodied) agents:

- human
- animal
- artificial

and then there are the mix-up, e.g.,

– cyborg

– centaur 

– and weaker candidates (e.g. lower biological systems).

What is an agent?

For human and animal agents (strong biological systems), the answer is simple:

> An agent is the offspring of an agent.

What is an agent?

For human and animal agents (strong biological systems), the answer is simple:

An agent is the offspring of an agent.

This is like to say:

An Italian is the offspring of an Italian.

What is an agent?

For human and animal agents (strong biological systems), the answer is simple:

An agent is the offspring of an agent.

This is like to say:

An Italian is the offspring of an Italian.

Nothing wrong with this, only that it is not telling us much and, even worse, it is not general:
it does not apply to artificial agents in general.

What is an agent?

We need to separate three problems:

- How can one identify agents?
- What can an agent do?
- What is an agent?

What is an agent?

We need to separate three problems:

- How can one identify agents?

What is an agent?

We need to separate three problems:

- How can one identify agents?

  Dennett's stances (physical, design, intentional)

What is an agent?

We need to separate three problems:

- How can one identify agents?

  Dennett's stances (physical, design, intentional)

- What can an agent do?

What is an agent?

We need to separate three problems:

- How can one identify agents?

  Dennett's stances (physical, design, intentional)

- What can an agent do?

  It discriminates, has preferences, decides, makes changes.

What is an agent?

We need to separate three problems:

- How can one identify agents?

  Dennett’s stances (physical, design, intentional)

- What can an agent do?

  It discriminates, has preferences, decides, makes changes.

- What is an agent?

What is an agent?

We need to separate three problems:

- How can one identify agents?

  Dennett's stances (physical, design, intentional)

- What can an agent do?

  It discriminates, has preferences, decides, makes changes.

- What is an agent?

  A perspectival physical entity that persists in time, discriminates, has preferences, decides and acts accordingly in the environment.

Definitions of agent /1

Some definitions of embodied agent from the literature.

(1) anything that is seen as “perceiving its environment through sensors and acting upon that environment through effectors.” (Russell and Norvig, 2010, p. 33)

Definitions of agent /1

Some definitions of embodied agent from the literature.

(1) anything that is seen as “perceiving its environment through sensors and acting upon that environment through effectors.” (Russell and Norvig, 2010, p. 33)

(2) “a system that tries to fulfill a set of goals in a complex, dynamic environment” (Maes, 1994, p. 136)

Definitions of agent /1

Some definitions of embodied agent from the literature.

(1) anything that is seen as “perceiving its environment through sensors and acting upon that environment through effectors.” (Russell and Norvig, 2010, p. 33)

(2) “a system that tries to fulfill a set of goals in a complex, dynamic environment” (Maes, 1994, p. 136)

(3) “any embodied system [that pursues] internal or external goals by its own actions while in continuous long-term interaction with the environment in which it is situated” (Beer, 1995, p. 173)

Definitions of agent /2

Some definitions of embodied agent from the literature.

(4) “entities which engage in normatively constrained, goal-directed, interaction with their environment” (Christensen and Hooker, 2000, p. 133)

Definitions of agent /2

Some definitions of embodied agent from the literature.

(4) "entities which engage in normatively constrained, goal-directed, interaction with their environment" (Christensen and Hooker, 2000, p. 133)

(5) (autonomous agent) "a system situated within and apart of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future." (Franklin and Graesser, 1996, p. 25).

Commonalities:
it is a system distinguishable from the environment,
able to sense/perceive that environment,
able to act/interact in pursuit of a goal.

Definitions of agent /2

Some definitions of embodied agent from the literature.

(4) "entities which engage in normatively constrained, goal-directed, interaction with their environment" (Christensen and Hooker, 2000, p. 133)

(5) (autonomous agent) "a system situated within and apart of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future." (Franklin and Graesser, 1996, p. 25).

Commonalities:
it is a system distinguishable from the environment,
able to sense/perceive that environment,
able to act/interact in pursuit of a goal.

Note: the definitions do not refer to intentionality.

A more interesting definition of agent

- An agent is “a system doing something by itself according to certain goals or norms within a specific environment.”

  Conditions:

  (1) the system is an individual;
  (2) the system is the active source of interaction; and
  (3) the interaction norm is generated by the system

  (Barandiaran, Di Paolo and Rohde, 2009, p. 374)

A more interesting definition of agent

- An agent is “a system doing something by itself according to certain goals or norms within a specific environment.”

  Conditions:

  (1) the system is an individual;
  (2) the system is the active source of interaction; and
  (3) the interaction norm is generated by the system

  (Barandiaran, Di Paolo and Rohde, 2009, p. 374)

Basics: system, distinguishable (the rest is environment), interactive, regulating.

(Again, intentionality is not an issue.)

Desired properties?

Which properties are characterizing agents?

- reactivity (maintain an ongoing relationship with the environment and respond to changes),
- proactiveness (take initiative, recognize opportunities),
- social ability (interact and cooperate with other agents),
- rationality,
- adaptability,
- . . .

What is a robot according to roboticists?

Robots can have different forms and functions but the scientific and engineering principles and algorithms that control them remain the same.

Although the term is used commonly and we have clear intuitions about it, it is hard to give a precise definition of what a robot is.
Generally people start from two core ideas:

- Carrying out actions automatically
- Being programmable (by a computer)

What is a robot according to roboticists?

Robots can have different forms and functions but the scientific and engineering principles and algorithms that control them remain the same.

Although the term is used commonly and we have clear intuitions about it, it is hard to give a precise definition of what a robot is.
Generally people start from two core ideas:

- Carrying out actions automatically
  (airplane autopilot? washing machine?)
- Being programmable (by a computer)

What is a robot according to roboticists?

Robots can have different forms and functions but the scientific and engineering principles and algorithms that control them remain the same.

Although the term is used commonly and we have clear intuitions about it, it is hard to give a precise definition of what a robot is.
Generally people start from two core ideas:

- Carrying out actions automatically
  (airplane autopilot? washing machine?)
- Being programmable (by a computer)
  (heating system? teller machine?)

Defining robots

There is no accepted definition of robot even though several proposals have been made.

“A robot is a machine –especially one programmable by a computer– capable of carrying out a complex series of actions automatically.” [Wikipedia]

A robot is “a machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer.” [Oxford English Dict]

Defining robots

There is no accepted definition of robot even though several proposals have been made.

“A robot is a machine –especially one programmable by a computer– capable of carrying out a complex series of actions automatically.” [Wikipedia]

A robot is “a machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer.”
[Oxford English Dict]

Adaptability is a crucial element and requires the use of sensors. Sensors enable a robot to verify the ongoing execution of complex tasks in a changing environment.

Defining robots

a more specialized attempt...

Robotics Institute of America (RIA):

A robot is a reprogrammable, multifunctional, manipulator designed to move material, parts, tools or specialised devices through variable programmed motions for the performance of a variety of tasks. The robot is automatically operating equipment, adaptable to complex conditions of the environment in which it operates, by means of reprogramming managing to prolong, amplify and replace one or more human functions in its interactions with the environment.

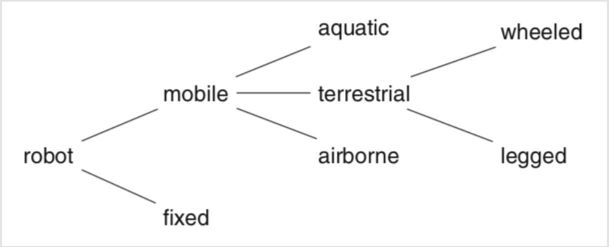
Defining robots

and another...

IEEE Standard for Ontologies for Robotics and Automation

An agentive device [...] in a broad sense, purposed to act in the physical world in order to accomplish one or more tasks. In some cases, the actions of a robot might be subordinated to actions of other agents [...], such as software agents (bots) or humans. A robot is composed of suitable mechanical and electronic parts.
Robots might form social groups, where they interact to achieve a common goal. A robot (or a group of robots) can form robotic systems together with special environments geared to facilitate their work.

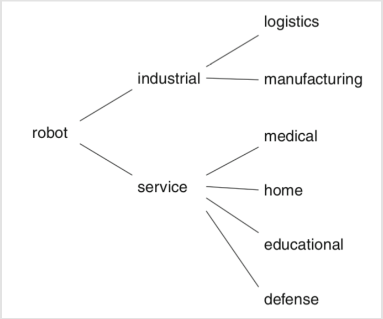
Classifying robots – 1



Classification of robots by environment and mechanism of interaction

Fixed robots are mostly industrial robotic manipulators. They are attached to a stable mount on the ground, so they can compute their position based on their internal state.
Mobile robots need to rely on their perception of the environment. Mobile robots need to deal with situations that are not precisely known in advance and that change over time (robotic vacuum cleaner, self-driving cars).
Different environments require significantly different design principles.

Classifying robots – 2



Classification of robots by intended application field and tasks they perform.

Industrial robots work in well-defined environments. Additional flexibility is required when industrial robots interact with humans and this introduces strong safety requirements, both for robotic arms and for mobile robots. The advantage of humans working with robots is that each can perform what they do best.

Important topics: purpose

Research in robotics comprises many aspects, in terms of purpose the most important are:

- mechanical manipulation (functionality)
- locomotion (functionality)
- computer vision (sensor)
- artificial intelligence (information extraction and management)

Important topics: structure and control

Research in robotics comprises many aspects, in terms of robot's structure and control the most important are:

- Joints and Links
- Sensors and Actuators
- Kinematics and Dynamics
- Planning and Control
- Artificial Intelligence

Applications

The predominant use still remains in the field of automation in manufacturing.
Automation replaces the worker with intelligent control systems, thereby contributing to increase in productivity, speed, and repeatability.

- Manufacturing robots
- Space robots
- Service robots
- Medical Robots
- Rehabilitation and assistive robots
- Entertainment Robotics

Course Overview

Ontological analysis vs ontologies

Ontological Analysis:
this refers to the study, guided by ontological principles, of a topic or a problem. The goal of the study is the understanding of the types of entities involved, the types of relations involved, the situations that are considered possible.

Ontological analysis vs ontologies

Ontological Analysis:
this refers to the study, guided by ontological principles, of a topic or a problem. The goal of the study is the understanding of the types of entities involved, the types of relations involved, the situations that are considered possible.
O.A. is the hard part in ontology research and determines the quality of the resulting ontological system.

Ontological analysis vs ontologies

Ontological Analysis:
this refers to the study, guided by ontological principles, of a topic or a problem. The goal of the study is the understanding of the types of entities involved, the types of relations involved, the situations that are considered possible.
O.A. is the hard part in ontology research and determines the quality of the resulting ontological system.

Ontologies:
these are “specifications of a conceptual system” concerned with the understanding of entities of interest and their relationships. Briefly, a set of explicit constraints on a domain. An ontology states the way we see (or our focus in) the world and is often written in a machine-readable language. An ontology should be built according to the ontological analysis of the domain/scenario.

Ontological analysis vs domain analysis and requirement analysis

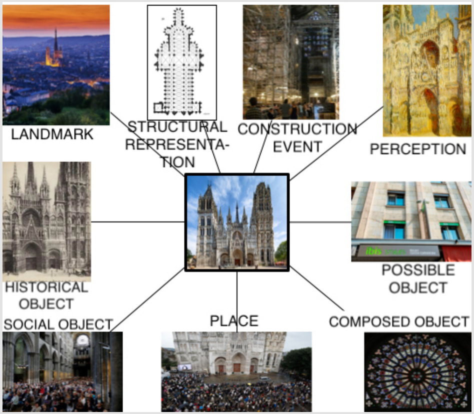
Ontological analysis should not be confused with domain and requirement analyses (*broadly understood*):

Domain analysis is the study of the topic or problem from the viewpoint of the expert. It starts from the traditional and consolidated view of the domain with the goal of classifying the topic or problem within the known knowledge system.

Requirement analysis is the study of the needs that make the understanding of the topic or a solution to the problem relevant. Requirement analysis elicits the constraints that should be satisfied by the understanding (e.g. the capacity to make certain predictions) or by the solution (e.g. avoiding certain situations).

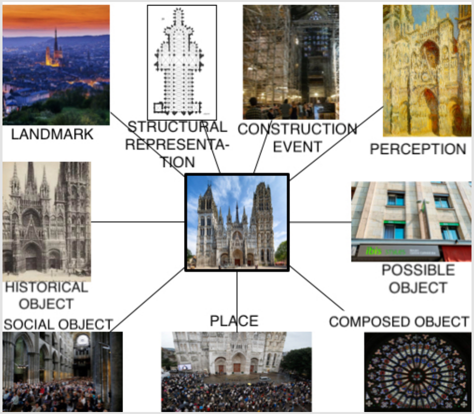
Different perspectives

Our knowledge of things is a cluster of different perspectives...



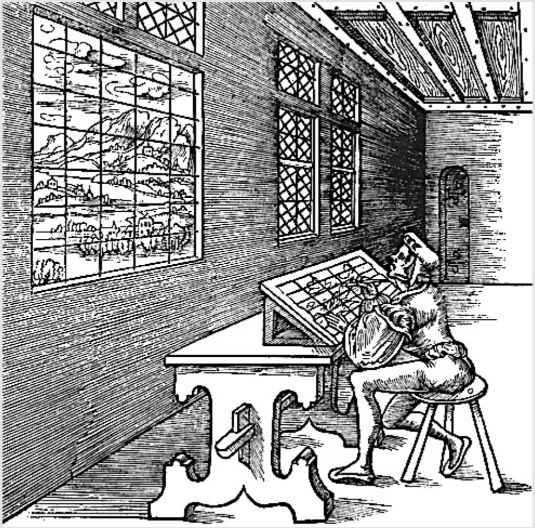
Different perspectives

Our knowledge of things is a cluster of different perspectives...

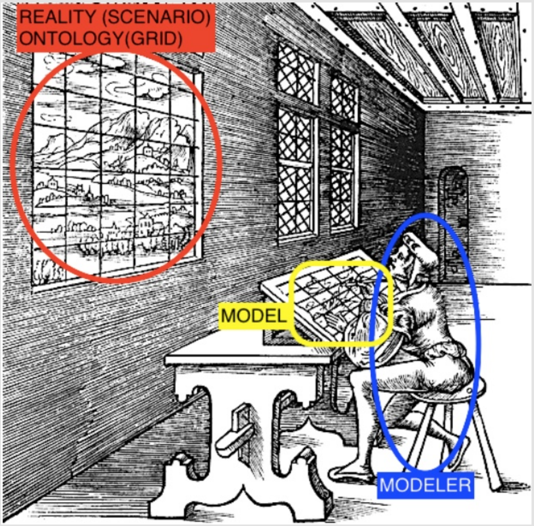


How can we make sense of and integrate this variety of perspectives?

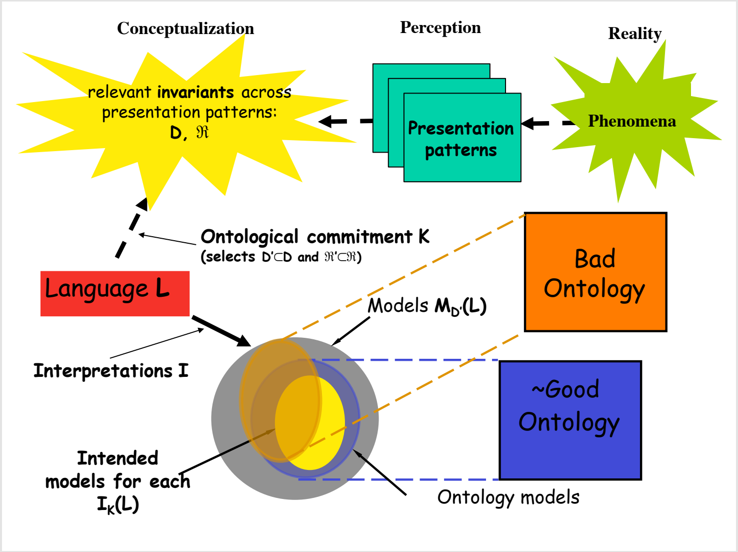
Leon Battista Alberti



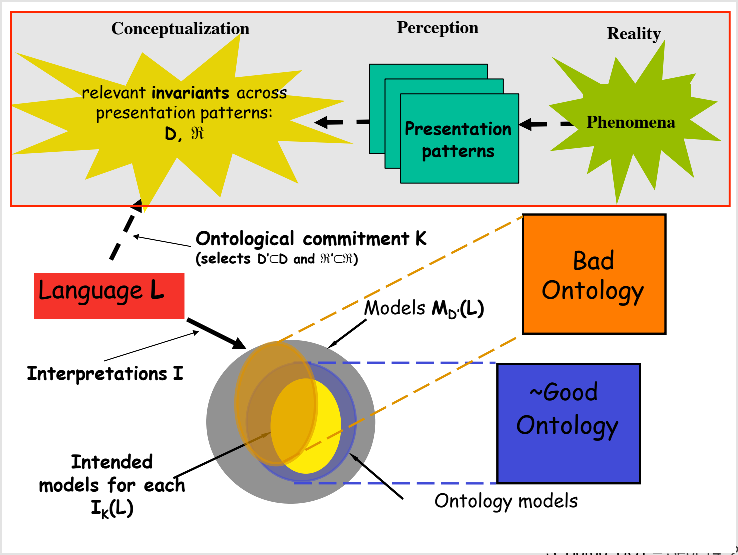
Leon Battista Alberti



The information flow from reality to models

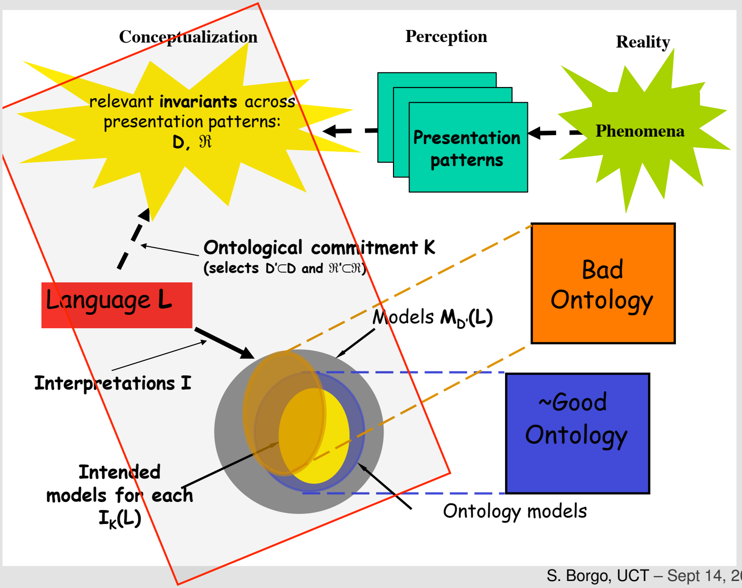


Flow step 1

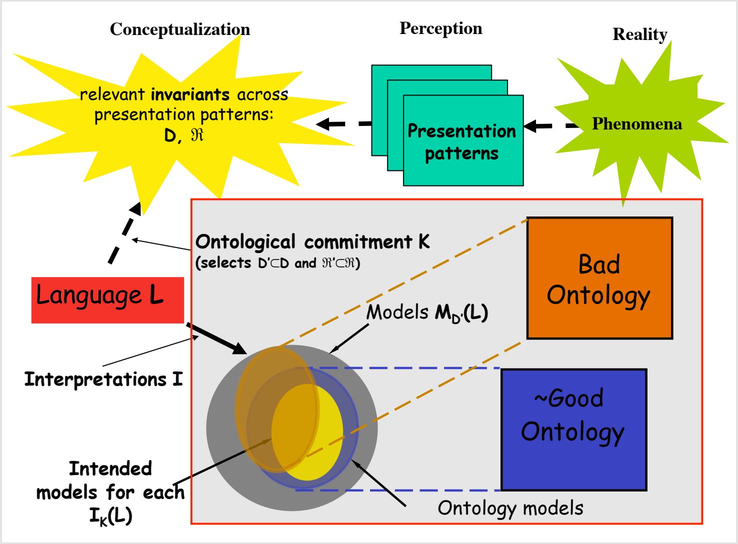


Conceptualization
Perception
Reality
relevant invariants across presentation patterns: D, $\mathfrak{R}$
Presentation patterns
Phenomena
Ontological commitment K
(selects D'⊂D and $\mathfrak{R}$'⊂$\mathfrak{R}$)
Language L
Models $M_{D'}(L)$
Bad Ontology
Interpretations I
~Good Ontology
Intended models for each $I_K(L)$
Ontology models

Flow step 2


Conceptualization
Perception
Reality
relevant invariants across presentation patterns: D, ℜ
Presentation patterns
Phenomena
Ontological commitment K
(selects D'⊂D and ℜ'⊂ℜ)
Language L
Bad Ontology
Models $M_{D'}(L)$
Interpretations I
~Good Ontology
Intended models for each $I_K(L)$
Ontology models

Role of ontology



What is in a foundational ontology?

Foundational ontologies are the most general formal ontologies.

They characterize general terms like
entity, event, process, spatial and *temporal location*...

and basic relations like
parthood, participation, dependence, identity...

The purpose is:

(1) to provide a formal description of entities and relationships that are common in all domains/perspectives

(2) to provide a consistent and unifying view

The ontology toolkit

Formal and ontological tools for ontology construction:

- Basic distinctions:

  Entities
  Properties, Qualities, Attributes
  Relations
  . . .

- Basic techniques:

  Stacking (co-location)
  Reification
  Modularity
  . . .

Understanding properties

Unfortunately, formal logic lacks suitable property constructs.

Let's discuss the following:

- "being a screwdriver"
- "being a disassembly tool"
- "being material"
- "having 1kg mass"

Understanding properties

Unfortunately, formal logic lacks suitable property constructs.

Let’s discuss the following:

- “being a screwdriver”

  candidate for a category: $Screwdriver(x)$

- “being a disassembly tool”

- “being material”

- “having 1kg mass”

Understanding properties

Unfortunately, formal logic lacks suitable property constructs.

Let's discuss the following:

- "being a screwdriver"

  candidate for a category: $Screwdriver(x)$

- "being a disassembly tool"

  candidate for a role: $\mathrm{CF}(DisassemblyTool, x, t)$

- "being material"

- "having 1kg mass"

Understanding properties

Unfortunately, formal logic lacks suitable property constructs.

Let's discuss the following:

- "being a screwdriver"

  candidate for a category: $Screwdriver(x)$

- "being a disassembly tool"

  candidate for a role: $\mathrm{CF}(DisassemblyTool, x, t)$

- "being material"

  candidate for a essential quality: $Material(x)$

- "having 1kg mass"

Understanding properties

Unfortunately, formal logic lacks suitable property constructs.

Let's discuss the following:

- "being a screwdriver"

  candidate for a category: $Screwdriver(x)$

- "being a disassembly tool"

  candidate for a role: $\mathrm{CF}(DisassemblyTool, x, t)$

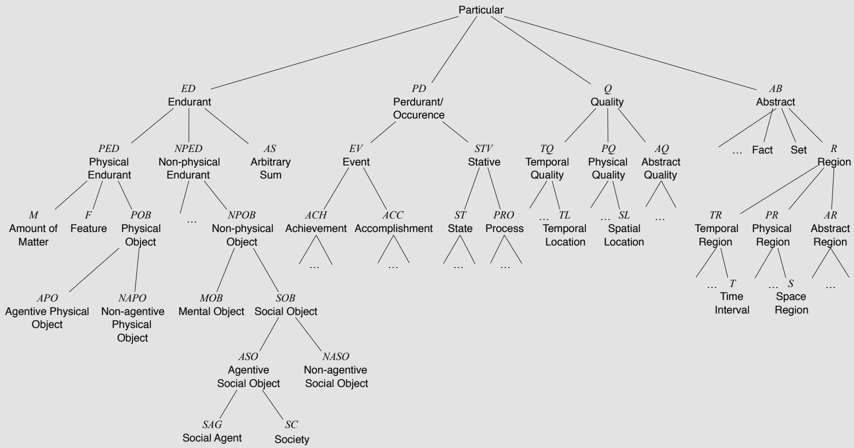
- "being material"

  candidate for a essential quality: $Material(x)$

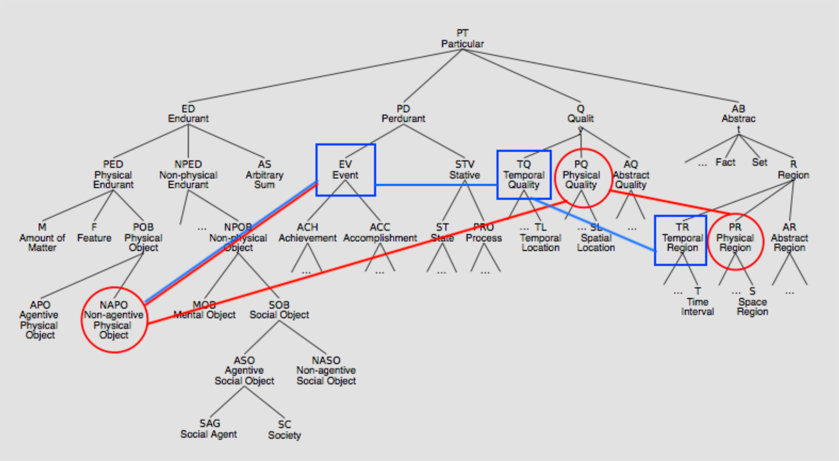
- "having 1kg mass"

  candidate for an individual quality: $\mathcal{I}(1kgMass, x)$

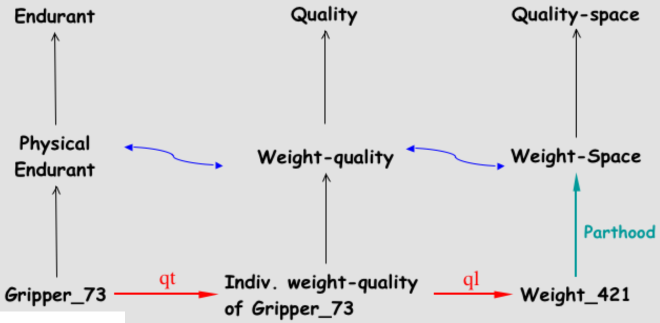
The DOLCE Taxonomy

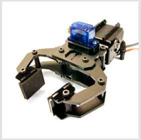


The relations across the taxonomy



The properties of a gripper





Course Overview

Device (ontologically speaking)

A robot needs to understand not just the physical reality but also our way to understand it and to differentiate between objects and roles.

Device (ontologically speaking)

A robot needs to understand not just the physical reality but also our way to understand it and to differentiate between objects and roles.

A device is an artefact that satisfies certain (structural and functional) constraints so that in certain environments and situations it manifests a (typical) behavior as selected by the designer.

Definition (Ontological Device)

A device D is a physical object which an agent(s) creates by two, possibly concurrent, intentional acts:

- the selection of a material entity (the constituent of D); and
- the attribution to D of technical qualities characterizing D's type.

Means (ontologically speaking)

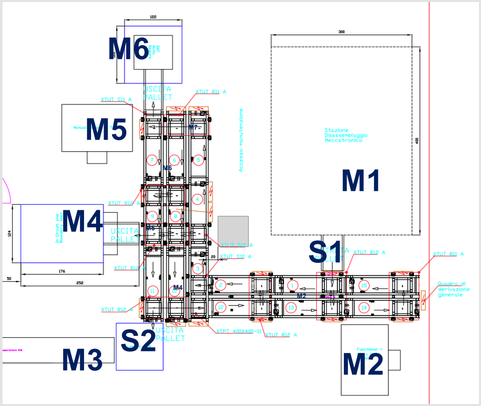
The means is an entity that manifests the behavior desired by the user in a given environment and situation.

Definition (Ontological Means)

An entity M is the means to an end if it is a physical object which plays the (functional) role assigned to it by the user in a given event.

Course Overview

Scenario: Robot in a controlled environment

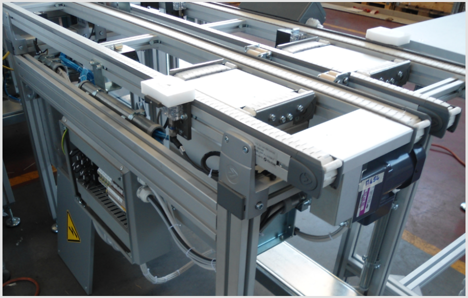


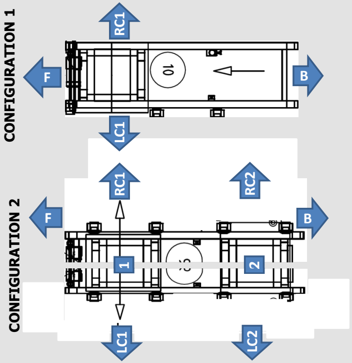
The plant is composed of automatic and manual machines devoted to perform loading/unloading, testing, repairing and shredding of PCBs and a reconfigurable transportation system connecting them.

The transportation system is composed by 15 reconfigurable mechatronic units, called transportation modules (TM).

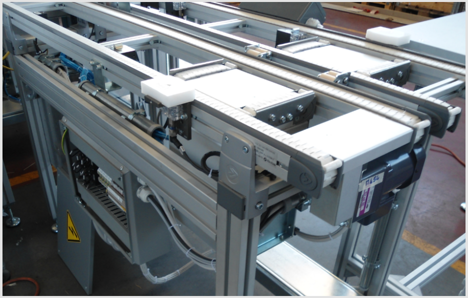
Printed Circuit Borad (PCB)

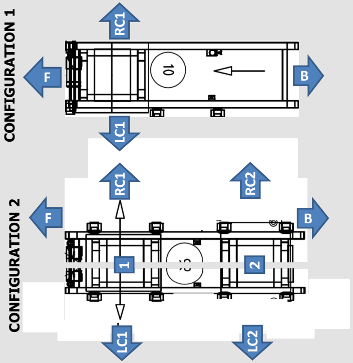
Scenario: Robot in a controlled environment /2





Scenario: Robot in a controlled environment /2





Considering the internal structure of this TM, it is possible to define three different types of component:

(1) the conveyor component;

(2) the port component; and

(3) the cross-transfer component.

Course Overview

Engineering functions (modeled by engineers)

In the treatment of engineering functions, the function is seen as an entity detached from the agent performing it:

- Functional representation: desired behavior of the device
- Functional basis: transformation of the input flows into output flows
- Functional concept: interpretation of a behavior of the device

Engineering functions (modeled by engineers)

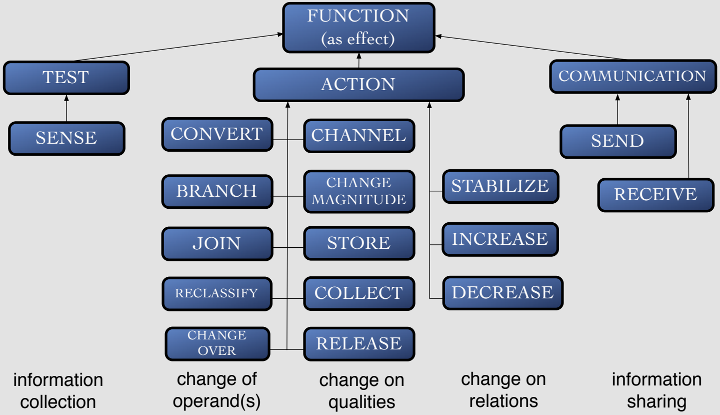
In the treatment of engineering functions, the function is seen as an entity detached from the agent performing it:

- Functional representation: desired behavior of the device
- Functional basis: transformation of the input flows into output flows
- Functional concept: interpretation of a behavior of the device

In all these cases, the user of the device is *irrelevant*.

This is not so when the robot has to reason about what to do.

Engineering functions from the robot's perspective



R. Mizoguchi et al. "A unifying definition for artifact and biological functions." App. Ont. 2016
S. Borgo et al. "A planning-based architecture for a reconfigurable manufacturing system." ICAPS 2016

Functions vs tasks

Engineering functions talk about homogeneous changes (or states) and are subdivided by types of change.

- To join is a change that riduces the number of (topological) objects by connecting two or more of them.
- To channel is a change in which an object changes location.
- To reclassify is a change in which an object changes status.
- To store is a state in which an object or a quantity is maintained in a certain position.
- . . .

Often these functions correspond only to a fragment of the change the agent has to realize.

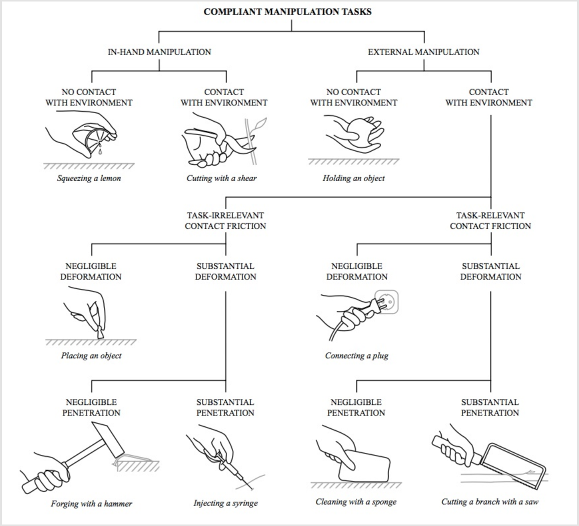
Tasks vs functions

Tasks are descriptions of changes in the world that are relevant for the agent. Here relevant means that the task's completion marks an important step toward the realization of the goal.

- To cut with a shear is a task that integrates a function (to branch) and a way of execution (cut with the shear).
- To squeeze a lemon is a task that integrates a function (to change magnitude) and the object (the lemon).
- To connect a plug is a task that integrates several functions (to channel, to join, to stabilize) and two objects (the plugh and the outlet).
- . . .

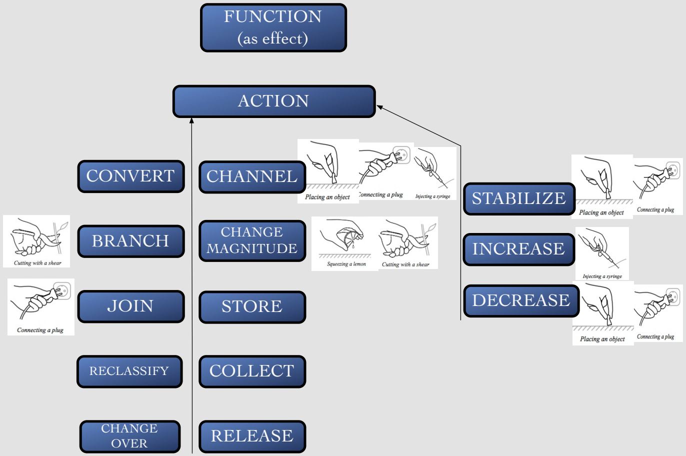
Most relevant tasks are <u>combinations</u> of several engineering functions and specific object types.

Manipulation tasks

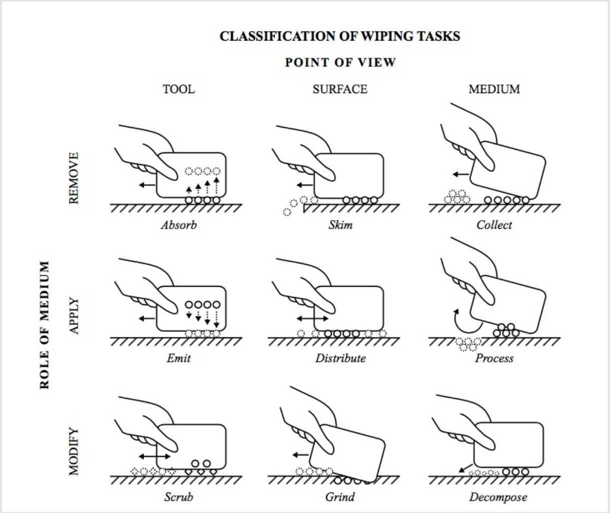


From “Classifying Compliant Manipulation Tasks for Automated Planning in Robotics”,D. Leidner et al., IROS 2015

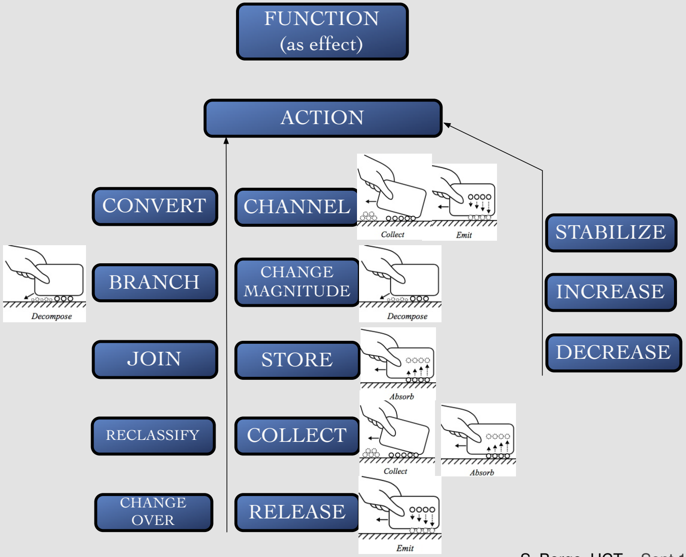
Aligning functions and manipulation tasks



Tasks can be very specific: wiping asks

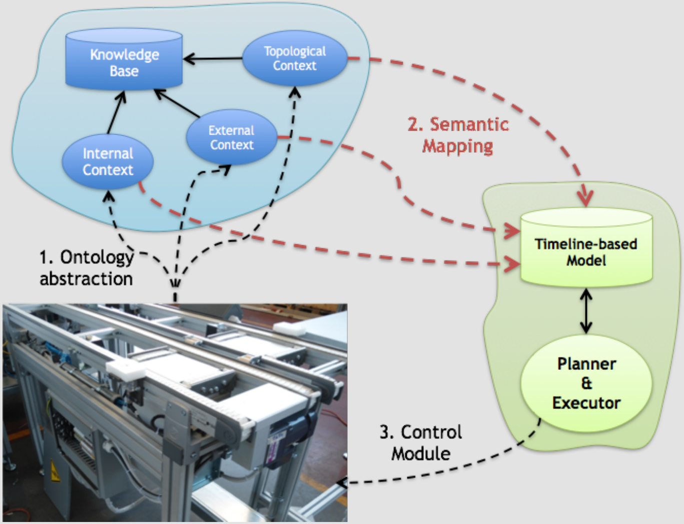


Aligning functions and wiping tasks

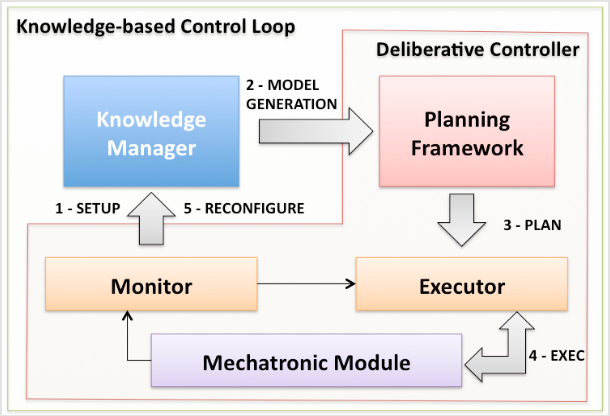


Course Overview

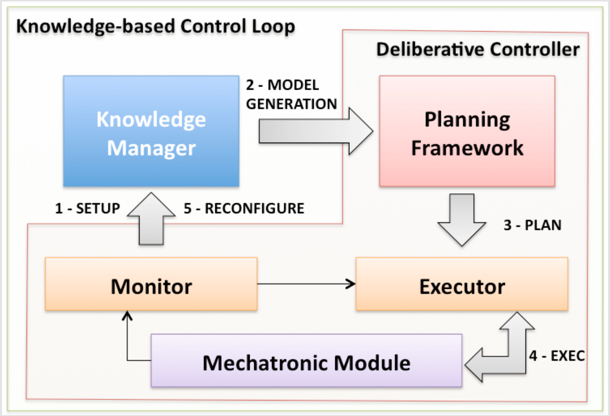
Physical robot, Knowledge module, Execution module



Knowledge module (KBCL)



Knowledge module (KBCL)



The overall cognitive architecture: a Knowledge Manager (top left), which contains the built-in know-how of the agent; and a Deliberative Controller, which constitutes a “classical” a plan-based control architecture.

Deliberative Controller

A plan-based control architecture (deliberative controller) has three layers:

(1) a deliberative layer which provides the agent with the capability of synthesizing the actions needed to achieve a goal (i.e., the Planning Framework);

(2) an executive layer which executes actions of a plan and continuously monitor their actual outcome with respect to the expected status of the system and the environment (i.e., the Monitor and the Executor); and

(3) the mechatronic system (and its functional processes) which represents the system and the environment to be controlled (i.e., in the scenario, the Mechatronic Module and the related transportation system).

Deliberative Controller

A plan-based control architecture (deliberative controller) has three layers:

(1) a deliberative layer which provides the agent with the capability of synthesizing the actions needed to achieve a goal (i.e., the Planning Framework);

(2) an executive layer which executes actions of a plan and continuously monitor their actual outcome with respect to the expected status of the system and the environment (i.e., the Monitor and the Executor); and

(3) the mechatronic system (and its functional processes) which represents the system and the environment to be controlled (i.e., in the scenario, the Mechatronic Module and the related transportation system).

The deliberative controller realizes a sense-plan-act cycle.

Planning

The Deliberative Controller relies on a static planning model which completely characterizes the capabilities of a TM and the associated working environment.

However, such a model is not capable of dynamically capturing changes in the configuration of the transportation system such as, e.g., changes concerning the local topology of a TM or changes concerning the internal configuration of a TM.
These changes affect the agent's capabilities.

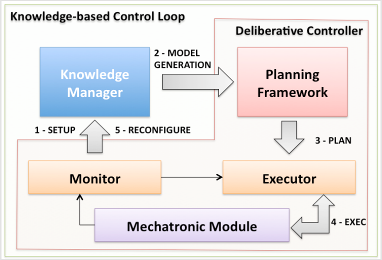
Planning

The Deliberative Controller relies on a static planning model which completely characterizes the capabilities of a TM and the associated working environment.

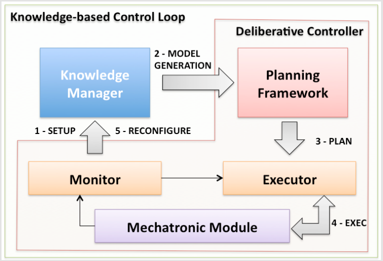
However, such a model is not capable of dynamically capturing changes in the configuration of the transportation system such as, e.g., changes concerning the local topology of a TM or changes concerning the internal configuration of a TM.
These changes affect the agent's capabilities.

The Knowledge Manager enhances the flexibility of the Deliberative Controller by dynamically generating planning models.

Knowledge module (KBCL): the initial steps

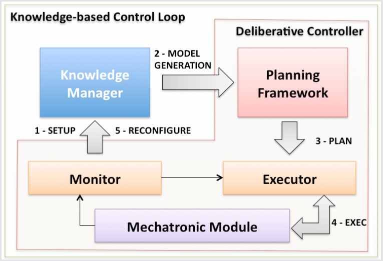


Knowledge module (KBCL): the initial steps



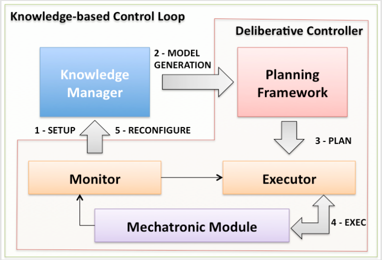
When the TM is activated, the Monitor collects the raw data from the Mechatronic Module with which a knowledge processing mechanism (1) initializes the KB (it adds the instances that represent the actual TM's state) (Point 1)

Knowledge module (KBCL): the initial steps



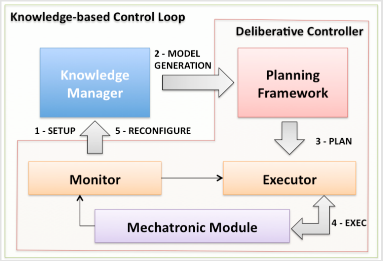
When the TM is activated, the Monitor collects the raw data from the Mechatronic Module with which a knowledge processing mechanism (1) initializes the KB (it adds the instances that represent the actual TM's state) (Point 1) and (2) dynamically generates the control model providing a first planning specification (Point 2).

Knowledge module (KBCL): the initial steps



When the TM is activated, the Monitor collects the raw data from the Mechatronic Module with which a knowledge processing mechanism (1) initializes the KB (it adds the instances that represent the actual TM's state) (Point 1) and (2) dynamically generates the control model providing a first planning specification (Point 2). Then the planning system generates a production plan (Point 3) and the plan execution is performed through the executive system (Point 4).

Knowledge module (KBCL): the initial steps



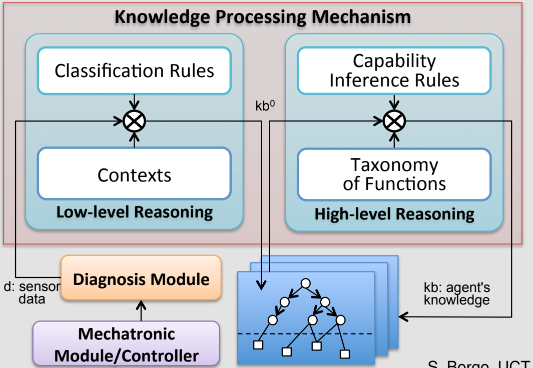
When the Monitor detects a change in the structure of the agent and/or its collaborators (e.g. a total or partial failure of a sensor/actuator or of a neighbor), the KBCL process starts a reconfiguration phase (Point 5) entailing the update of the KB, and starting a new iteration of the overall loop.

Knowledge processing mechanism

Each KB is specific to the agent. The management of such a KB relies on a knowledge processing mechanism implemented by means of a *Rule-based Inference Engine* which leverages a set of inference rules to generated and updated a KB of an agent.

Knowledge processing mechanism

Each KB is specific to the agent. The management of such a KB relies on a knowledge processing mechanism implemented by means of a *Rule-based Inference Engine* which leverages a set of inference rules to generated and updated a KB of an agent.



Knowledge mechanism

This mechanism involves two reasoning steps:

Knowledge mechanism

This mechanism involves two reasoning steps:

(1) the low-level reasoning step ($\rightarrow$ *local working environment*)
it is about the components that actually compose the agent's structure (e.g., ports, conveyors, etc.), and the associated collaborators. It relies on the internal and local contexts of the ontology and a set of classification rules.

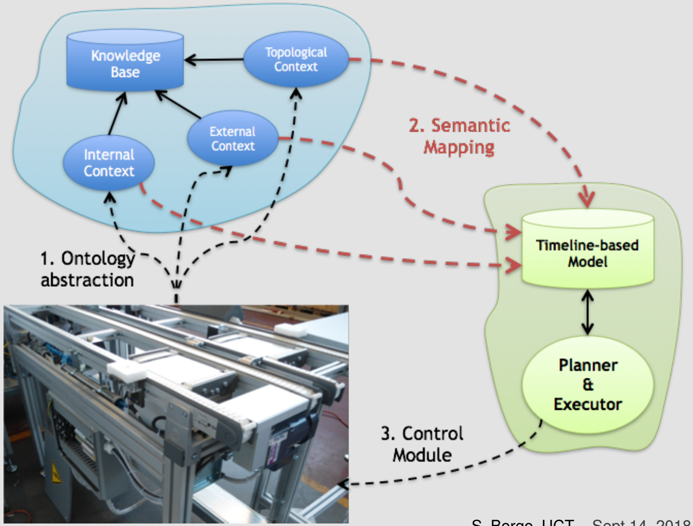
Knowledge mechanism

This mechanism involves two reasoning steps:

(1) the low-level reasoning step ($\rightarrow$ *local working environment*)
it is about the components that actually compose the agent's structure (e.g., ports, conveyors, etc.), and the associated collaborators. It relies on the internal and local contexts of the ontology and a set of classification rules.

(2) the high-level reasoning step ($\rightarrow$ *working environment and functional capabilities*)
it relies on the taxonomy of functions and the capability inference rules to complete the knowledge processing mechanism. It works on the KB developed by the previous step (internal and local context of the agent).

Towards cognition-based agents



Leveraging on the ontology and contexts

The local context provides:

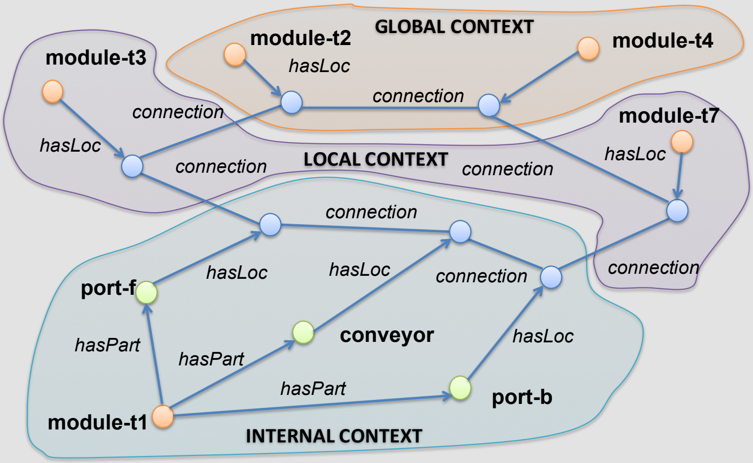
- list of ports/interfaces
- list of internal states
- list of engines/actuators

The ontology and the global context provide general information, e.g.:

- ports are locations
- conveyors are connectors of locations
- connection is transitive for the Channel (transportation) function

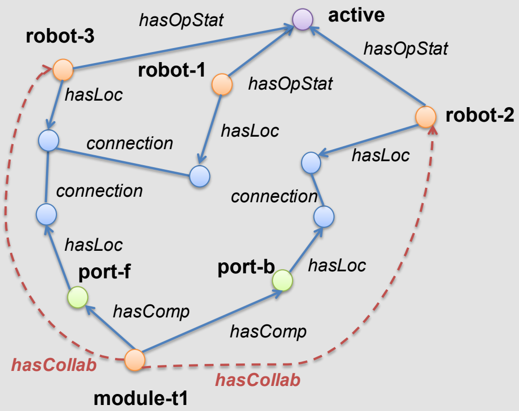
This suffices to generate all the possible ways for a transportation agent to execute a Channel function.

Contextual knowledge: internal, local, general



Elaboration of data received from the Diagnosis Module

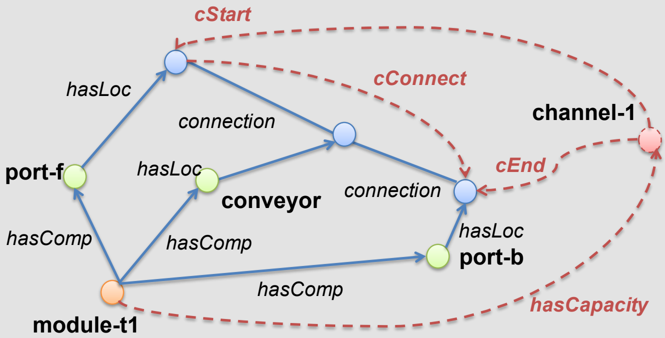
Building internal and local knowledge /1



Inferring collaborators of a TM (low-level reasoning)

$$ROBOT(r) \wedge PORT(p) \wedge hasLoc(p, l_p, t) \wedge ROBOTPART(p, r, t) \wedge hasOpStat(p, active, t) \wedge ROBOT(c) \wedge hasLoc(c, l_c, t) \wedge connection(l_p, l_c, t) \to hasCollab(r, c, t)$$

Building internal and local knowledge /2



Inferring collaborators of a TM (low-level reasoning)

$$ROBOT(r) \wedge CONVEYOR(c_1) \wedge hasOpStat(c_1, active, t) \wedge COMPONENT(c_2) \wedge COMPONENT(c_3) \wedge hasLoc(c_1, l_1, t) \wedge hasLoc(c_2, l_2, t) \wedge hasLoc(c_3, l_3, t) \wedge connection(l_2, l_1, t) \wedge connection(l_1, l_3, t) \wedge \rightarrow hasCapacity(r, f) \wedge CHANNEL(f) \wedge cStart(f, l_2) \wedge cEnd(f, l_3) \wedge cConnect(l_2, l_3)$$

The rational of the rule

$ROBOT(r) \wedge CONVEYOR(c_1) \wedge hasOpStat(c_1, active, t) \wedge$
$COMPONENT(c_2) \wedge COMPONENT(c_3) \wedge hasLoc(c_1, l_1, t) \wedge$
$hasLoc(c_2, l_2, t) \wedge hasLoc(c_3, l_3, t) \wedge connection(l_2, l_1, t) \wedge$
$connection(l_1, l_3, t) \wedge \rightarrow hasCapacity(r, f) \wedge CHANNEL(f) \wedge cStart(f, l_2) \wedge$
$cEnd(f, l_3) \wedge cConnect(l_2, l_3)$

This rule takes the functional interpretation of the *CONVEYOR* category as the set of components that can perform channel functions.

The rational of the rule

$ROBOT(r) \wedge CONVEYOR(c_1) \wedge hasOpStat(c_1, active, t) \wedge COMPONENT(c_2) \wedge COMPONENT(c_3) \wedge hasLoc(c_1, l_1, t) \wedge hasLoc(c_2, l_2, t) \wedge hasLoc(c_3, l_3, t) \wedge connection(l_2, l_1, t) \wedge connection(l_1, l_3, t) \wedge \rightarrow hasCapacity(r, f) \wedge CHANNEL(f) \wedge cStart(f, l_2) \wedge cEnd(f, l_3) \wedge cConnect(l_2, l_3)$

This rule takes the functional interpretation of the $CONVEYOR$ category as the set of components that can perform channel functions.

If a conveyor component connects two components of the TM through its spatial location (clause $connection(l_2, l_1, t) \wedge connection(l_1, l_3, t)$), then the conveyor can perform a primitive channel function between the components' locations.

The rational of the rule

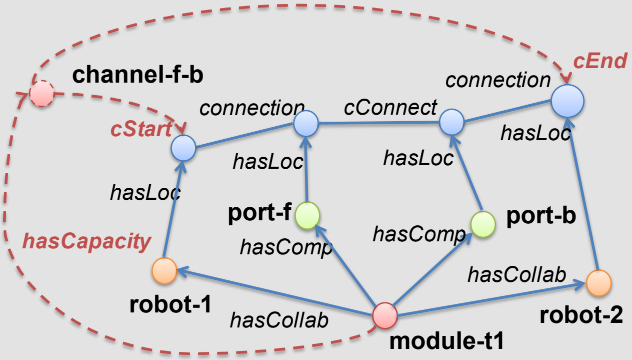
$ROBOT(r) \wedge CONVEYOR(c_1) \wedge hasOpStat(c_1, active, t) \wedge COMPONENT(c_2) \wedge COMPONENT(c_3) \wedge hasLoc(c_1, l_1, t) \wedge hasLoc(c_2, l_2, t) \wedge hasLoc(c_3, l_3, t) \wedge connection(l_2, l_1, t) \wedge connection(l_1, l_3, t) \wedge \rightarrow hasCapacity(r, f) \wedge CHANNEL(f) \wedge cStart(f, l_2) \wedge cEnd(f, l_3) \wedge cConnect(l_2, l_3)$

This rule takes the functional interpretation of the $CONVEYOR$ category as the set of components that can perform channel functions.

If a conveyor component connects two components of the TM through its spatial location (clause $connection(l_2, l_1, t) \wedge connection(l_1, l_3, t)$), then the conveyor can perform a primitive channel function between the components' locations.

Moreover, the $cConnect(l_2, l_3)$ (complex channel function) is a transitive predicate which allows to connect different channel functions. If two spatial locations are connected through the $cConnect$ predicate then there exists a composition of primitive channel functions that “connects” them.

Building internal and local knowledge /3



$$\begin{gathered} ROBOT(r) \wedge ROBOT(rc_1) \wedge ROBOT(rc_2) \wedge hasCollab(r, rc_1, t) \wedge \\ hasLoc(rc_1, rl_1, t) \wedge hasCollab(r, rc_2, t) \wedge hasLoc(rc_2, rl_2, t) \wedge PORT(c_1) \wedge \\ hasOpState(c_1, active, t) \wedge hasLoc(c_1, l_1, t) \wedge PORT(c_2) \wedge \\ hasOpState(c_2, active, t) \wedge hasLoc(c_2, l_2, t) \wedge connection(l_1, rl_1, t) \wedge \\ connection(l_2, rl_2, t) \wedge cConnect(l_1, l_2) \rightarrow \\ hasCapacity(r, f) \wedge CHANNEL(f) \wedge cStart(f, rl_1) \wedge cEnd(f, rl_2) \end{gathered}$$

Variables for the timeline-based planner

From the control perspective, it is possible to identify three different classes of state variables:

(1) functional state variables;

(2) primitive state variables; and

(3) external state variables.

Variables for the timeline-based planner

From the control perspective, it is possible to identify three different classes of state variables:

(1) functional state variables;

(2) primitive state variables; and

(3) external state variables.

Functional state variables model a physical system as a whole in terms of the high-level functions it can perform.

Variables for the timeline-based planner

From the control perspective, it is possible to identify three different classes of state variables:

(1) functional state variables;

(2) primitive state variables; and

(3) external state variables.

Functional state variables model a physical system as a whole in terms of the high-level functions it can perform.

Primitive state variables model the physical and/or logical elements that compose a physical system. In particular, the state variables model the elements needed to control the execution of high-level functions.

Variables for the timeline-based planner

From the control perspective, it is possible to identify three different classes of state variables:

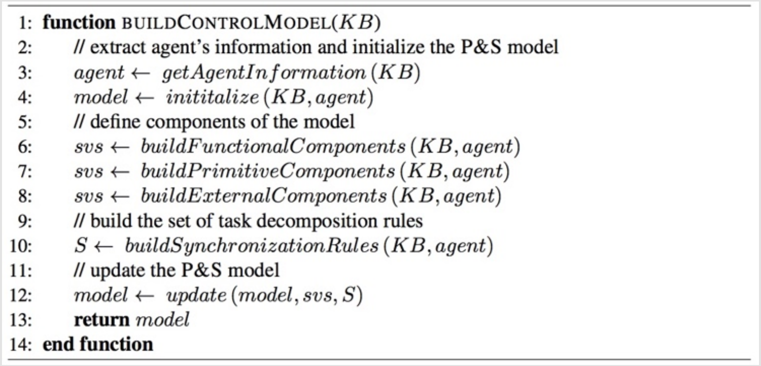
(1) functional state variables;
(2) primitive state variables; and
(3) external state variables.

Functional state variables model a physical system as a whole in terms of the high-level functions it can perform.

Primitive state variables model the physical and/or logical elements that compose a physical system. In particular, the state variables model the elements needed to control the execution of high-level functions.

External state variables model elements of the domain whose behavior is not directly under the control of the system like conditions that must hold to successfully perform operations.

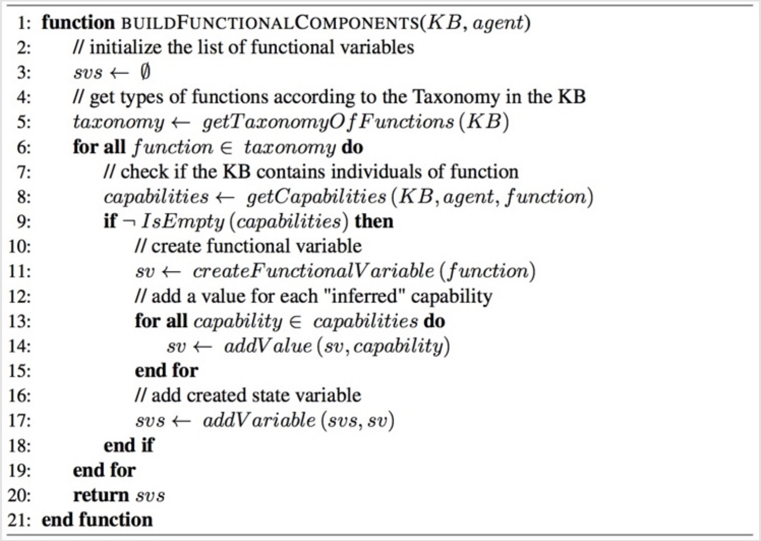
Generating the model



```
**function** BUILDCONTROLMODEL($KB$)
    // extract agent's information and initialize the P&S model
    $agent \leftarrow getAgentInformation(KB)$
    $model \leftarrow inititalize(KB, agent)$
    // define components of the model
    $svs \leftarrow buildFunctionalComponents(KB, agent)$
    $svs \leftarrow buildPrimitiveComponents(KB, agent)$
    $svs \leftarrow buildExternalComponents(KB, agent)$
    // build the set of task decomposition rules
    $S \leftarrow buildSynchronizationRules(KB, agent)$
    // update the P&S model
    $model \leftarrow update(model, svs, S)$
    **return** $model$
**end function**
```

The model generation procedure

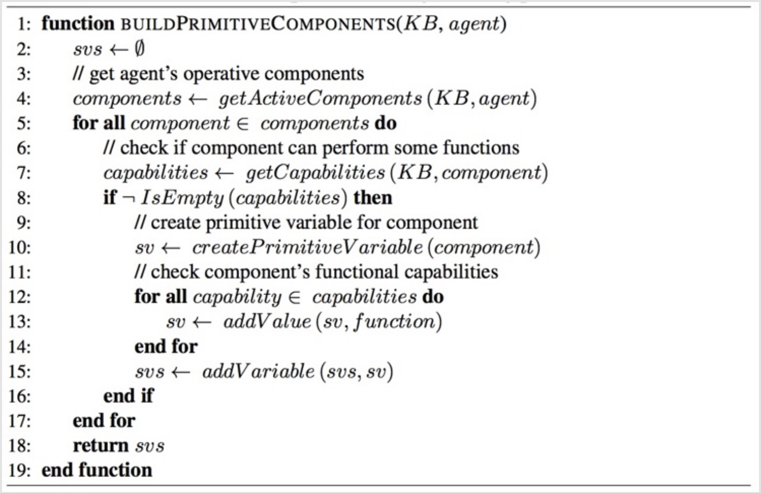
Generating functional information



```
function BUILDFUNCTIONALCOMPONENTS($KB, agent$)
    // initialize the list of functional variables
    $svs \leftarrow \emptyset$
    // get types of functions according to the Taxonomy in the KB
    $taxonomy \leftarrow getTaxonomyOfFunctions(KB)$
    for all $function \in taxonomy$ do
        // check if the KB contains individuals of function
        $capabilities \leftarrow getCapabilities(KB, agent, function)$
        if $\neg IsEmpty(capabilities)$ then
            // create functional variable
            $sv \leftarrow createFunctionalVariable(function)$
            // add a value for each "inferred" capability
            for all $capability \in capabilities$ do
                $sv \leftarrow addValue(sv, capability)$
            end for
            // add created state variable
            $svs \leftarrow addVariable(svs, sv)$
        end if
    end for
    return $svs$
end function
```

The functional variable generation procedure

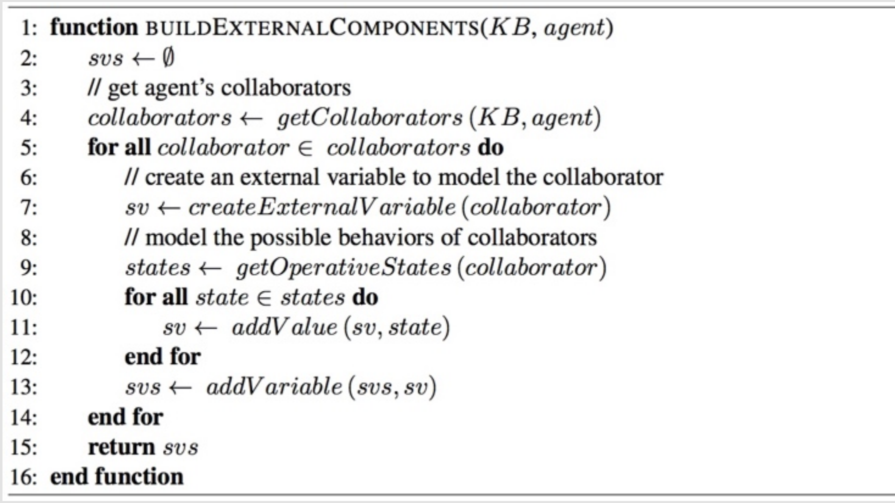
Generating primitive variables



```
function BUILDPRIMITIVECOMPONENTS($KB$, $agent$)
    $svs \leftarrow \emptyset$
    // get agent's operative components
    $components \leftarrow getActiveComponents(KB, agent)$
    for all $component \in components$ do
        // check if component can perform some functions
        $capabilities \leftarrow getCapabilities(KB, component)$
        if $\neg IsEmpty(capabilities)$ then
            // create primitive variable for component
            $sv \leftarrow createPrimitiveVariable(component)$
            // check component's functional capabilities
            for all $capability \in capabilities$ do
                $sv \leftarrow addValue(sv, function)$
            end for
            $svs \leftarrow addVariable(svs, sv)$
        end if
    end for
    return $svs$
end function
```

The primitive variable generation procedure

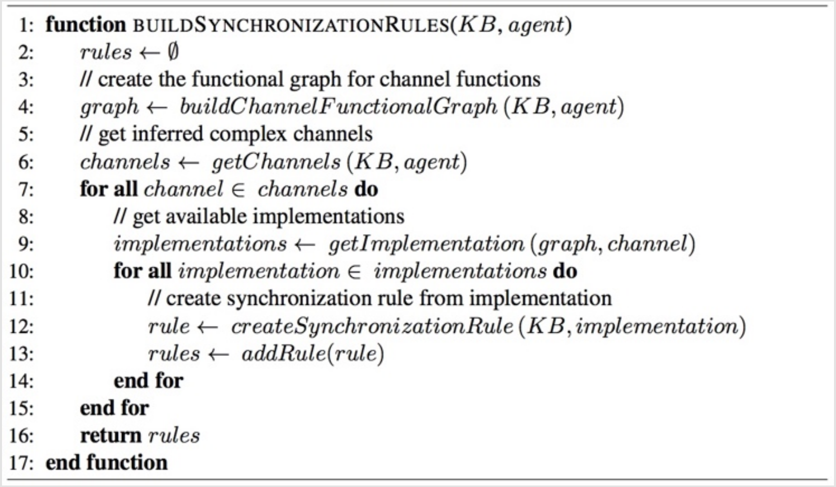
Generating external variables



```
function BUILDEXTERNALCOMPONENTS($KB$, $agent$)
    $svs \leftarrow \emptyset$
    // get agent's collaborators
    $collaborators \leftarrow getCollaborators(KB, agent)$
    for all $collaborator \in collaborators$ do
        // create an external variable to model the collaborator
        $sv \leftarrow createExternalVariable(collaborator)$
        // model the possible behaviors of collaborators
        $states \leftarrow getOperativeStates(collaborator)$
        for all $state \in states$ do
            $sv \leftarrow addValue(sv, state)$
        end for
        $svs \leftarrow addVariable(svs, sv)$
    end for
    return $svs$
end function
```

The external variable generation procedure

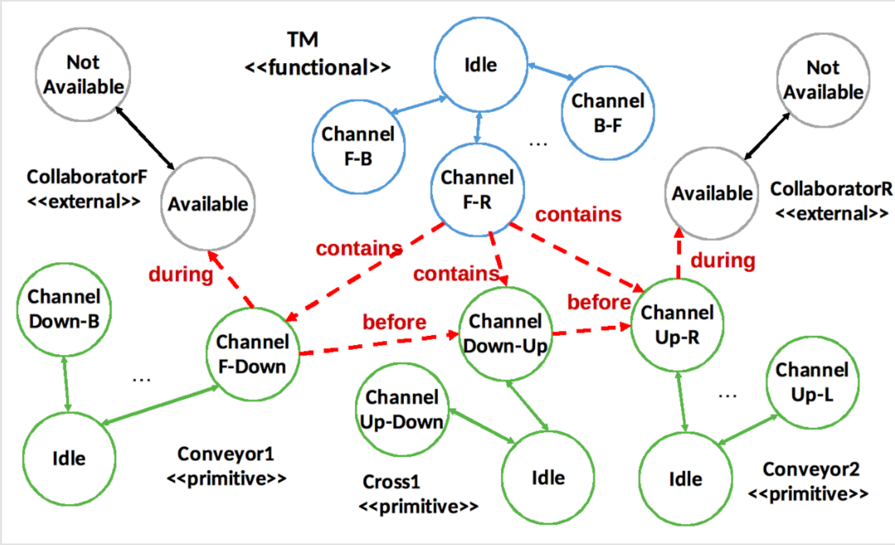
Generating synchronization rules



```
function BUILDSYNCHRONIZATIONRULES($KB, agent$)
    $rules \leftarrow \emptyset$
    // create the functional graph for channel functions
    $graph \leftarrow buildChannelFunctionalGraph\,(KB, agent)$
    // get inferred complex channels
    $channels \leftarrow getChannels\,(KB, agent)$
    for all $channel \in channels$ do
        // get available implementations
        $implementations \leftarrow getImplementation\,(graph, channel)$
        for all $implementation \in implementations$ do
            // create synchronization rule from implementation
            $rule \leftarrow createSynchronizationRule\,(KB, implementation)$
            $rules \leftarrow addRule(rule)$
        end for
    end for
    return $rules$
end function
```

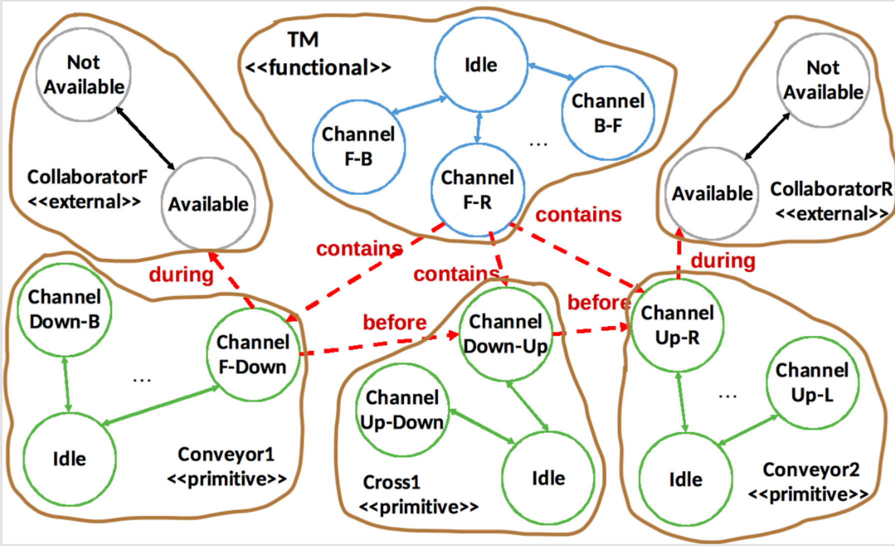
The synchronization rule generation procedure (inter-component causal and temporal requirements for the plan to be successful, they describe dependencies between the variables of a planning domain and may determine a hierarchy among them.)

The generated timeline-based model



A partial timeline-based model (TM equipped with one cross-transfer unit)

The generated timeline-based model



A partial timeline-based model (TM equipped with one cross-transfer unit)

Implementation details

The ontology is provided in FOL but FOL is used only for preprocessing (primarily to ensure conceptual consistency).

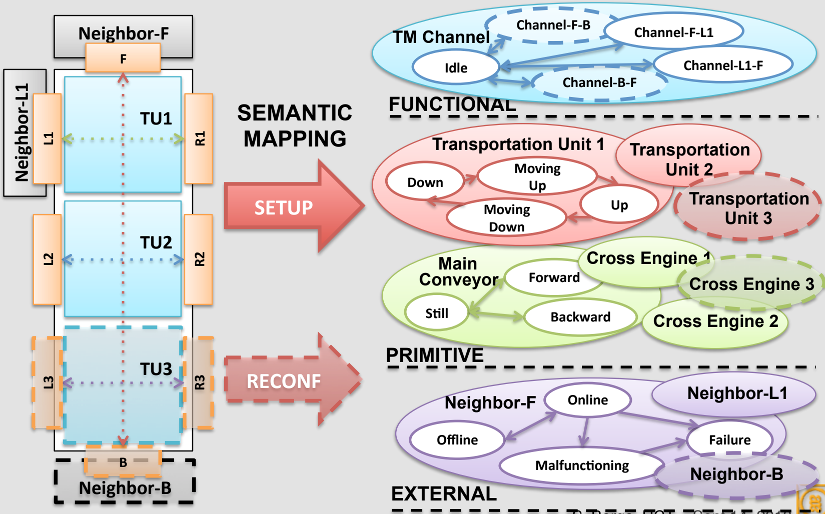
Most of the inferences at runtime are done in the OWL version of the KB (we exploit primarily the contextual classification and relationships).

The ontology editor Protégé has been used for KB design and testing.

For runtime reasoning in the Knowledge Manager, we have used the Ontology and RDF APIs and Inference API provided by the Apache Jena Software Library.

Finally, the Deliberative Controller has been realized by means of the GOAC architecture whose deliberative features are implemented by means of APSI-TRF.

Modeling an adaptive autonomous agent aware of its capabilities



Conclusions

Ontology applied to robotics faces new challenges since it has to deal with perspectival knowledge.

Conclusions

Ontology applied to robotics faces new challenges since it has to deal with perspectival knowledge.

A robot needs to understand what exists, what it can do and how.

Conclusions

Ontology applied to robotics faces new challenges since it has to deal with perspectival knowledge.

A robot needs to understand what exists, what it can do and how.

A robot needs to deal with contextual information and changing environments.

Conclusions

Ontology applied to robotics faces new challenges since it has to deal with perspectival knowledge.

A robot needs to understand what exists, what it can do and how.

A robot needs to deal with contextual information and changing environments.

A robot needs to understand other agents (desires, capacities, attitude).

Conclusions

Ontology applied to robotics faces new challenges since it has to deal with perspectival knowledge.

A robot needs to understand what exists, what it can do and how.

A robot needs to deal with contextual information and changing environments.

A robot needs to understand other agents (desires, capacities, attitude).

Ontology is one necessary component for coherently integrating all this data and knowledge.

Conclusions

Ontology applied to robotics faces new challenges since it has to deal with perspectival knowledge.

A robot needs to understand what exists, what it can do and how.

A robot needs to deal with contextual information and changing environments.

A robot needs to understand other agents (desires, capacities, attitude).

Ontology is one necessary component for coherently integrating all this data and knowledge.

Thank you