Applications of category theory to automated planning and program compilation in robotics

Angeline Aguinaldo

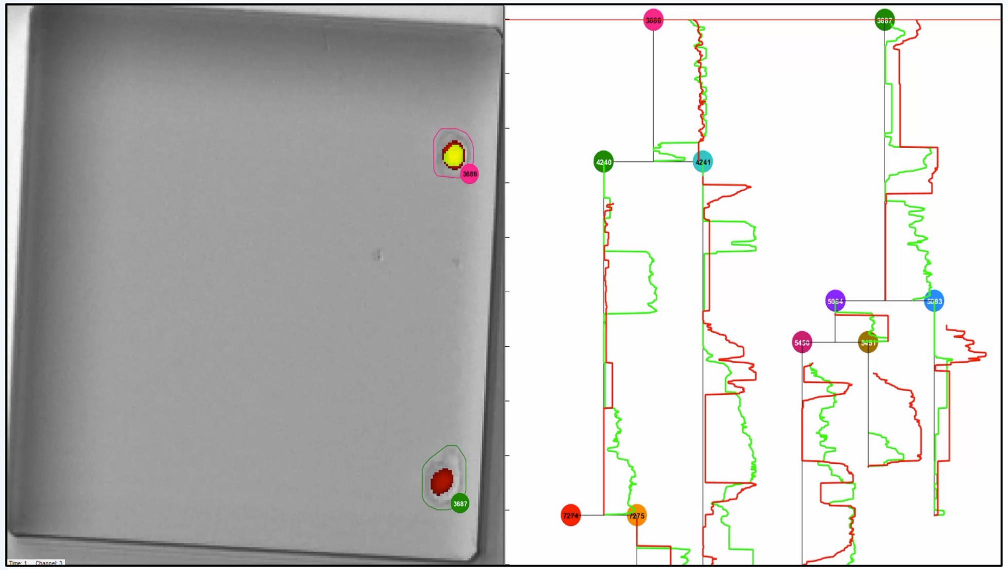
University of Maryland, College Park

Johns Hopkins University Applied Physics Laboratory

Xerox PARC, Design Seminar
August 12, 2022

Contents

LEVER: stem cell segmentation, tracking, and lineaging. Bioimage Lab. Drexel University.

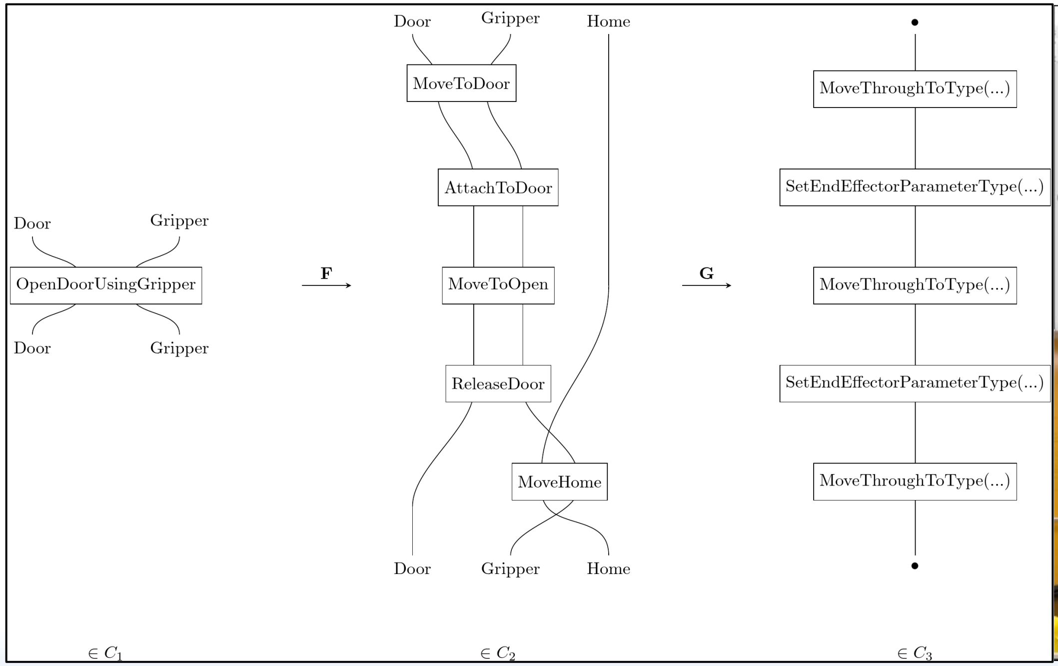


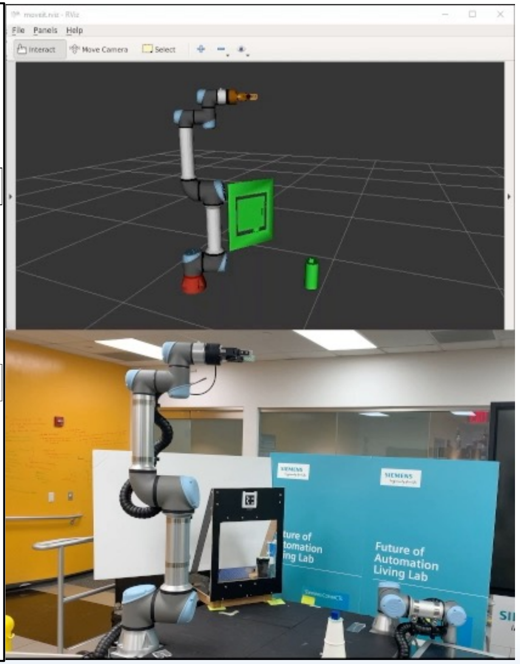
Humanitarian Assistance and Disaster Relief (HADR) Data Products for Hurricane Dorian 2019 Response

Drexel UNIVERSITY

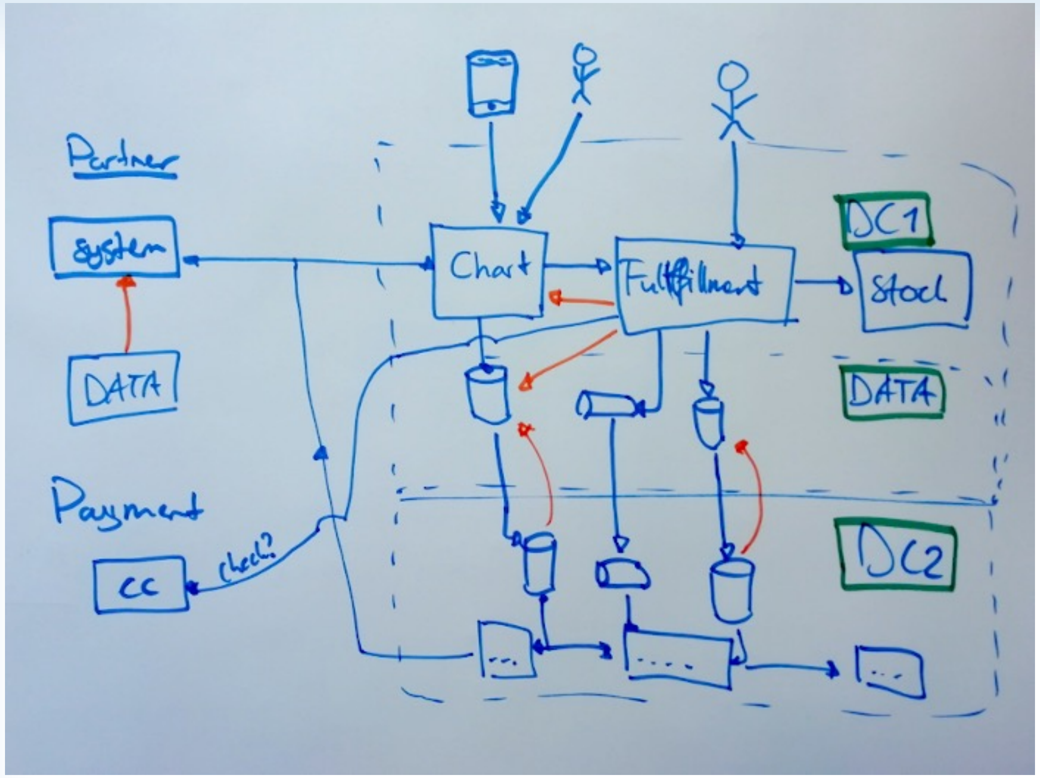
JOHNS HOPKINS APPLIED PHYSICS LABORATORY

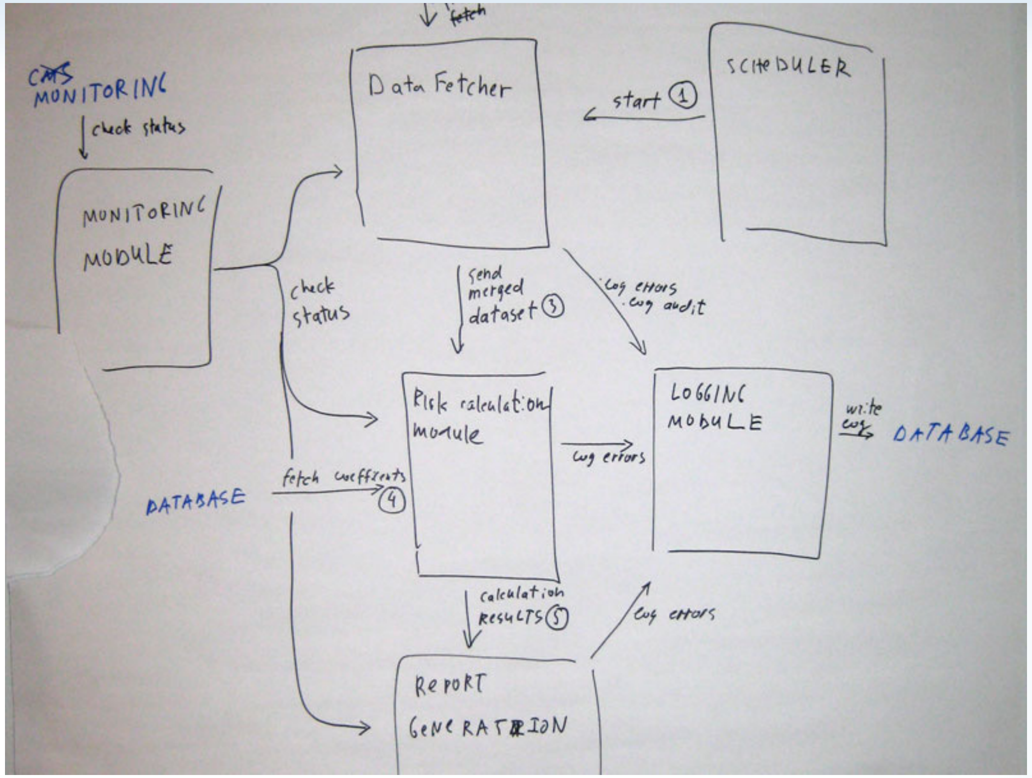
UNIVERSITY OF MARYLAND 1856

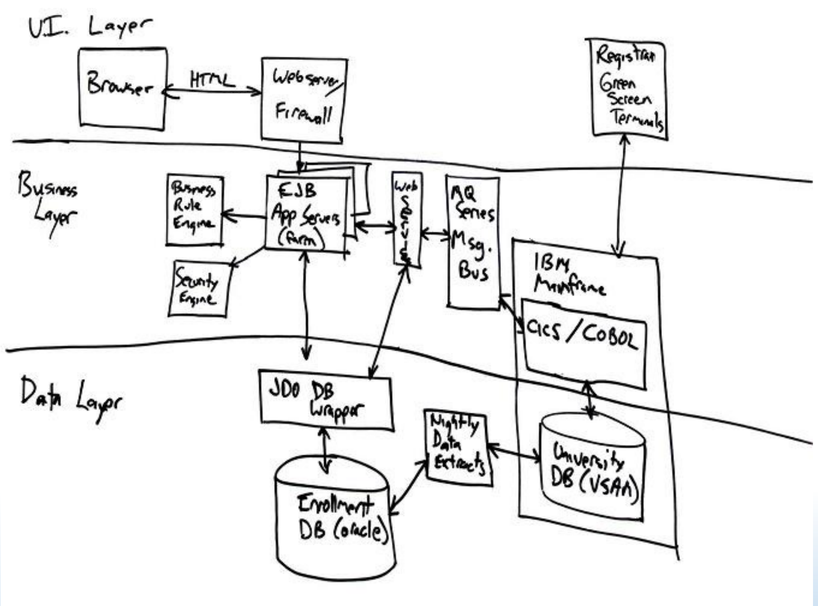




RoboCat: A category theoretic framework for robotic interoperability using goal-oriented programming

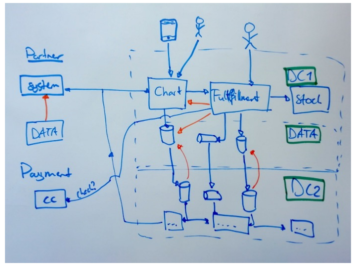

Partner
system
DATA
Payment
CC
check?
Chart
Fulfillment
Stock
DC1
DATA
DC2

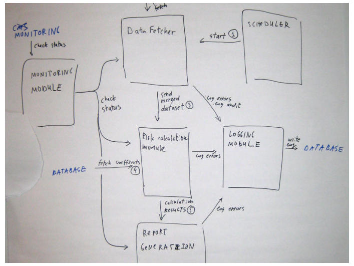

MONITORING
check status
MONITORING MODULE
Data Fetcher
SCHEDULER
start ①
check status
send merged dataset ③
log errors
log audit
Risk calculation module
LOGGING MODULE
write log
DATABASE
log errors
DATABASE
fetch coefficients ④
calculation results ⑤
log errors
REPORT GENERATION

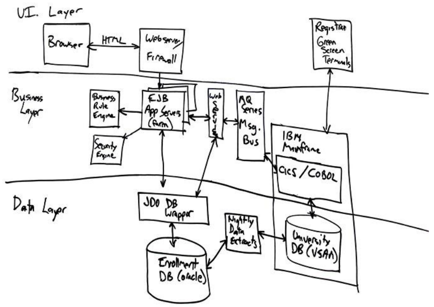

U.I. Layer
Browser
HTML
Webserver Firewall
Registrar Green Screen Terminals
Business Layer
Business Rule Engine
EJB App Servers (farm)
Web Services
MQ Series Msg. Bus
Security Engine
IBM Mainframe
CICS / COBOL
Data Layer
JDO DB Wrapper
Nightly Data Extracts
University DB (VSAM)
Enrollment DB (oracle)

Generalization

Mathematics







Engineering

Implementation

Contents

Context in decision-making

Example Software Project:
"I want to a program that can tell me how many animals are in this image"

"Oh, I want it to be able to tell me what type of animals are in it"

"And I want it to tell me the location of these animals on Earth"

Count animals

My animals

Object detection algorithm

Preprocess image

Image classification algorithm

Geolocate animal

Really hard question: How did a change in the problem (context) impact the space of designs that could address it?

Context in *automated* decision-making

”Engineers are not the only professional designers. Everyone [or thing] designs who devises **courses of action** aimed at changing existing situations into preferred ones.”

– Herbert Simon, *The Science of Design: Creating the Artificial*

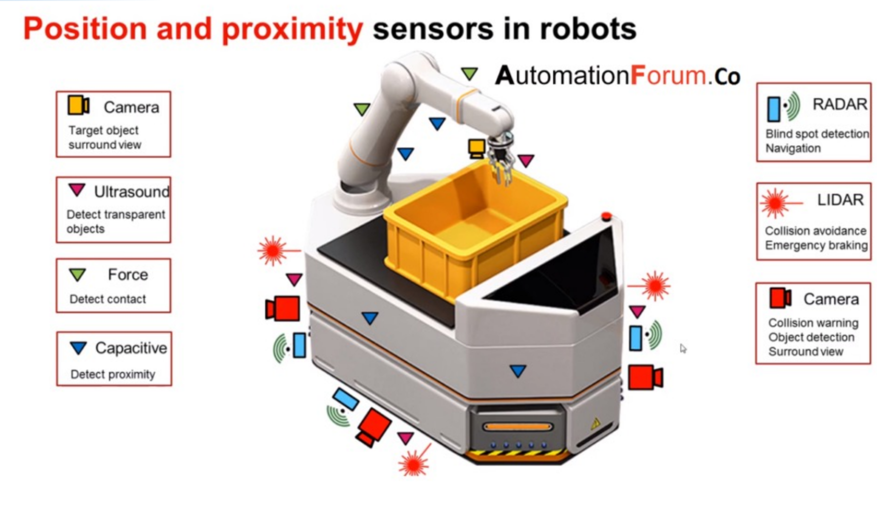
☞ This sounds a lot like planning…

Let’s scope it down to domains with more defined structures, e.g. knowledge representation and automated planning.

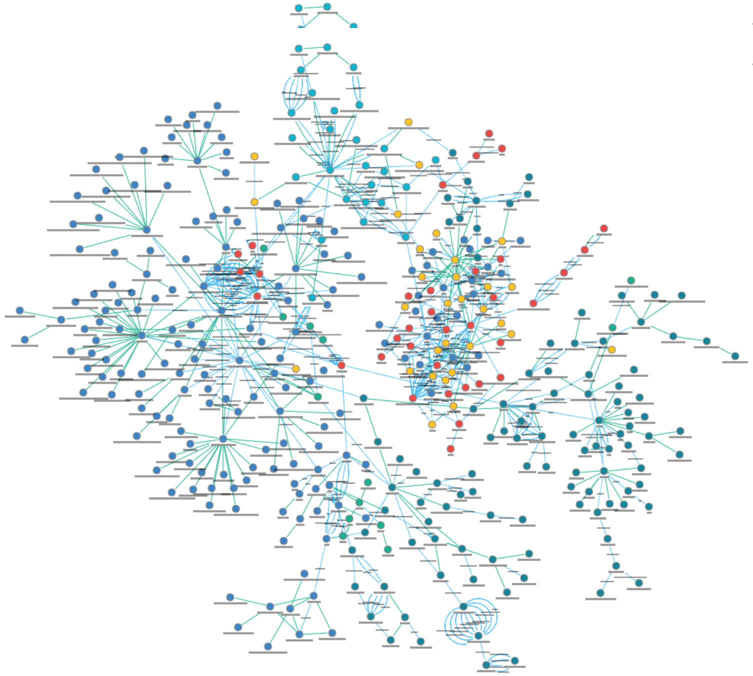
Release the **robots**!

Making use of context in robotics

Context awareness – mechanism that allows an agent to adjust its behavior in response to dynamic context information such as location and resources; traditionally for mobile and IoT devices



https://automationforum.co/what-are-sensors-on-a-robot-and-why-are-sensors-important-to-robots/



Increased availability and capability of sensors results in an increase of information.

This increases the computational demands as more algorithms that use the information get deployed.

Deciding what information is relevant makes knowledge interoperable between tasks.

A general framework for determining what is contextually important (*contextual attention*)

Terminology

Definition (Context). *Context* is a description of the characteristics of the environment an agent must act in.

Definition (Action). An *action* is an operation that changes the state of some or all characteristics of the environment.

Definition (Task plan). A *task plan* is sequence of actions that achieves a specified goal.

Definition (Contextual attention). *Contextual attention* is the identification of context entities that are most important in achieving the task. Importance means that some property of achievable tasks exceeds a given threshold when the context entity is removed or modified.

Related methods for contextual attention

In perception and sensor fusion

- Filling in gaps in images based on context
- Representation learning (find most concise state representation)
- Value of information

In knowledge representation and planning

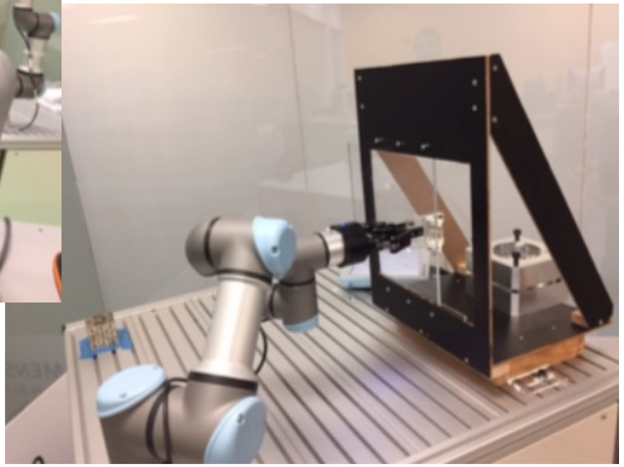
- Case-based reasoning (Schank 1982)
- Recommender systems
- Bayesian network, POMDPs, MDPs

Limitations

- Data-driven, requires learning
- Requires attention criteria a priori
- Focused on inferring high-level context from low-level context
- No denotational semantics
- Not tied to capability

Example Scenarios

Manufacturing

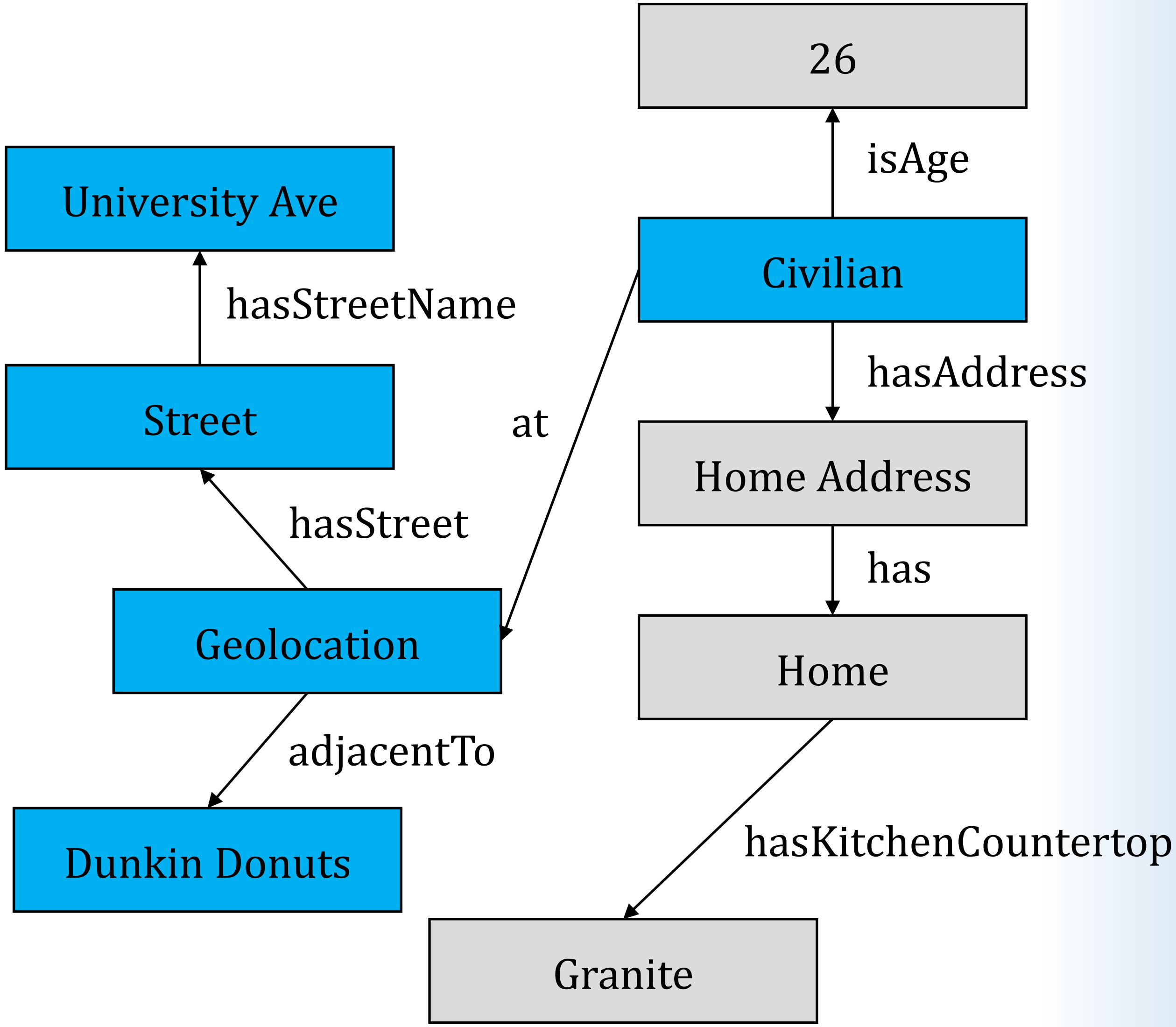
Context:
- ➢ Machine has a hinge door
- ➢ Item is rod of length 2 meters
- ➢ Rod is bronze metal

Disaster relief

Context:
- ➢ On University Ave
- ➢ Adjacent to Dunkin Donuts
- ➢ Has granite, not ceramic, kitchen countertops

Example Scenario

Disaster relief (path planning)



Task: Go to injured civilian

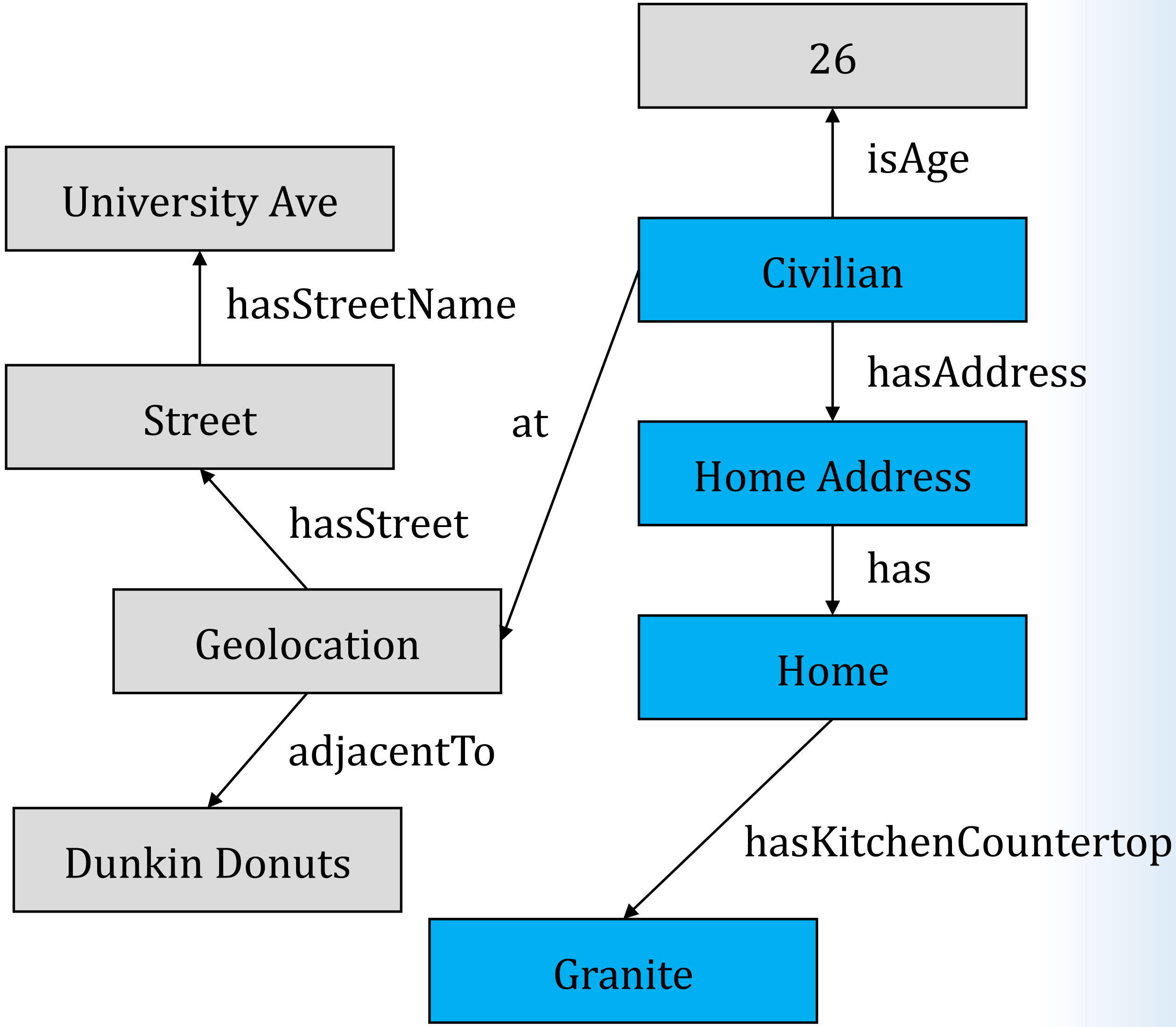
1. Move forward until you University Ave
2. Make a right at the Dunkin Donuts
3. Locate civilian on street

Example Scenario

Home renovation (demolition)

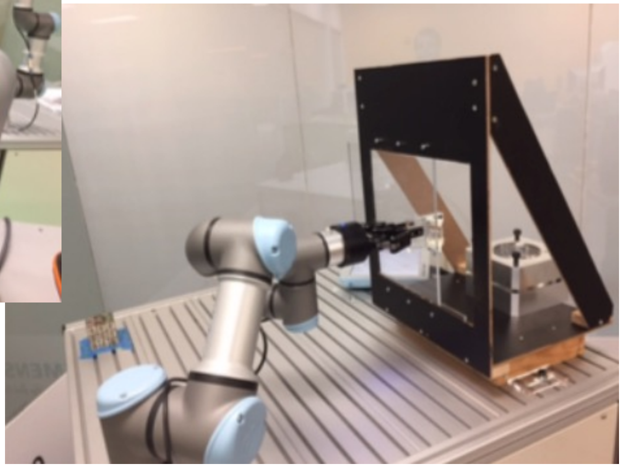
Task: Replace granite with ceramic in kitchen

1. Identify surfaces with granite
2. Measure surface
3. Remove surface
4. Add ceramic slate of correct size



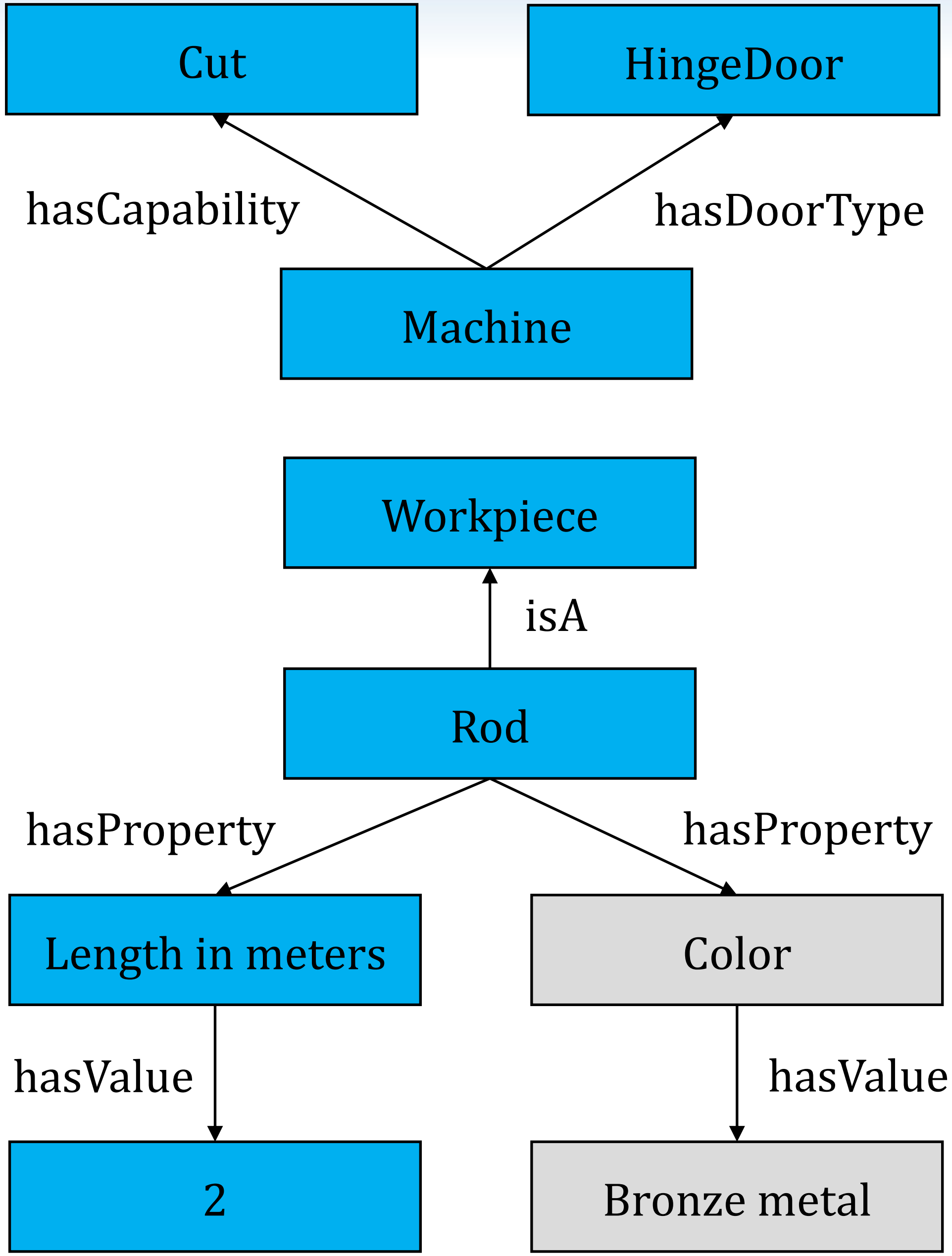
Example Scenario

Manufacturing (machine-tending)



Task: Cut rod to 1 meter

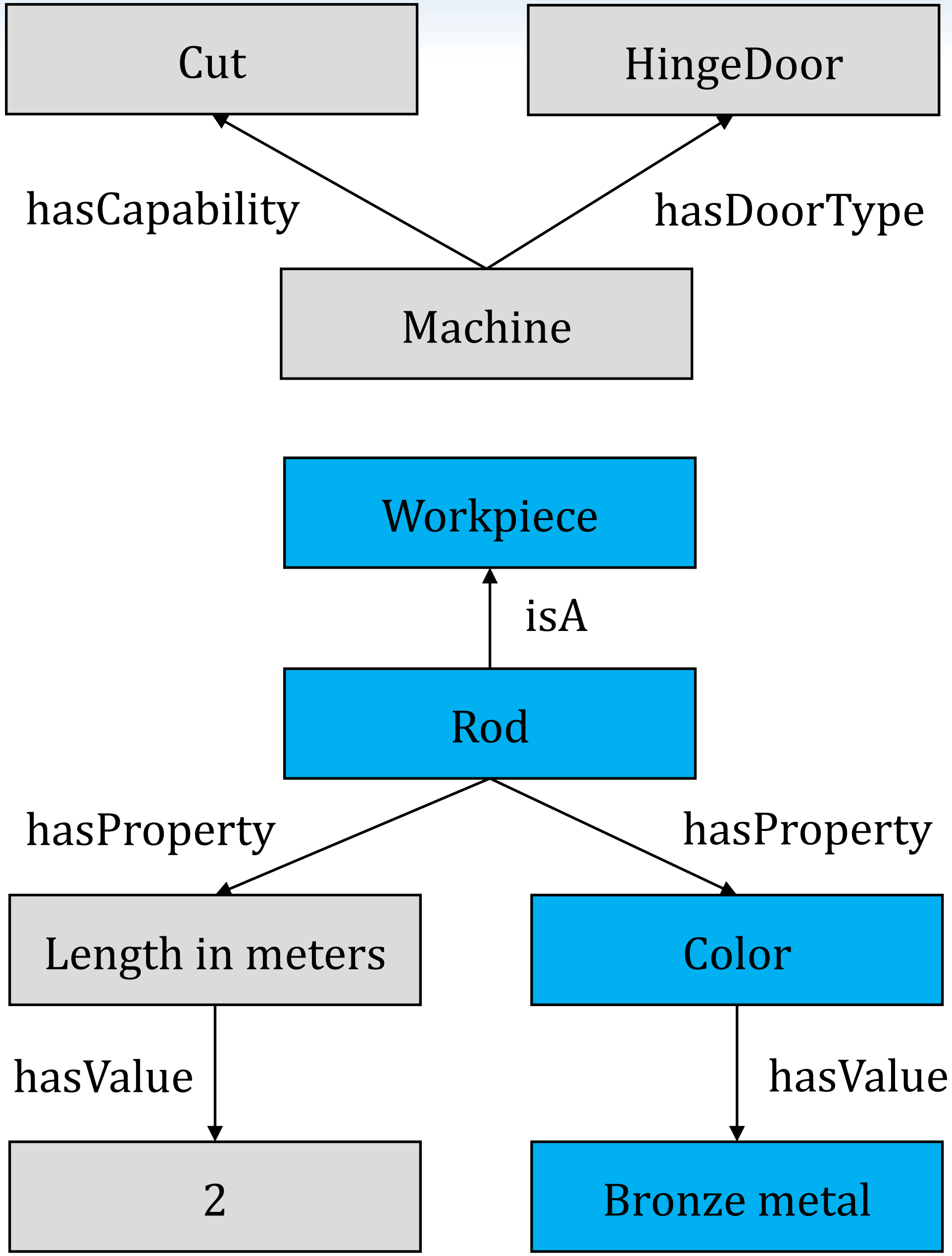
1. Open door
2. Pick rod greater than 1 meter
3. Place in machine
4. Close door



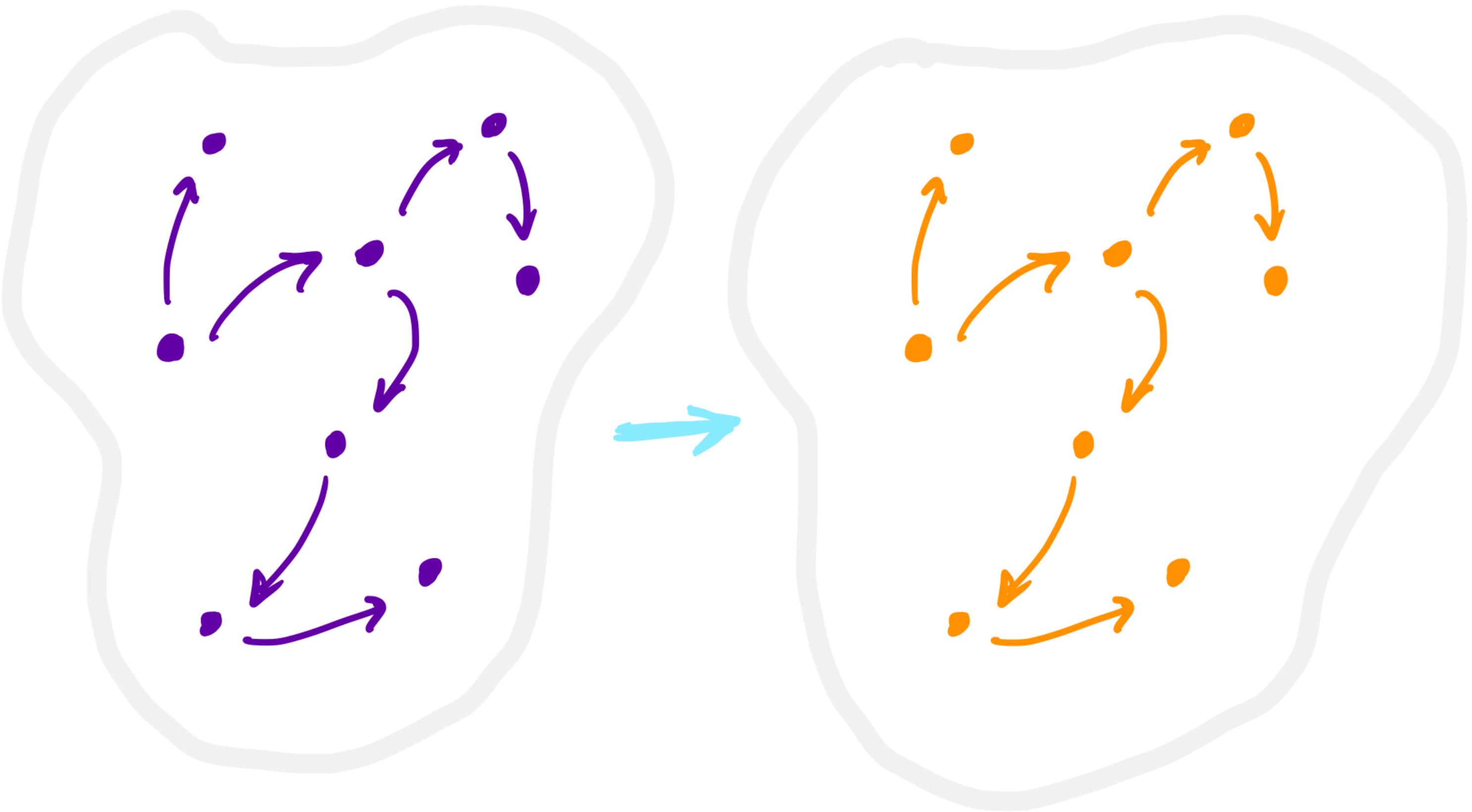
Example Scenario

Manufacturing (painting)

Task: Paint rod silver metal
1. Locate rod that is not silver metal
2. Paint rod silver metal



Informally speaking...



Some machine tracking **context**

Some machine tracking **achievable tasks**

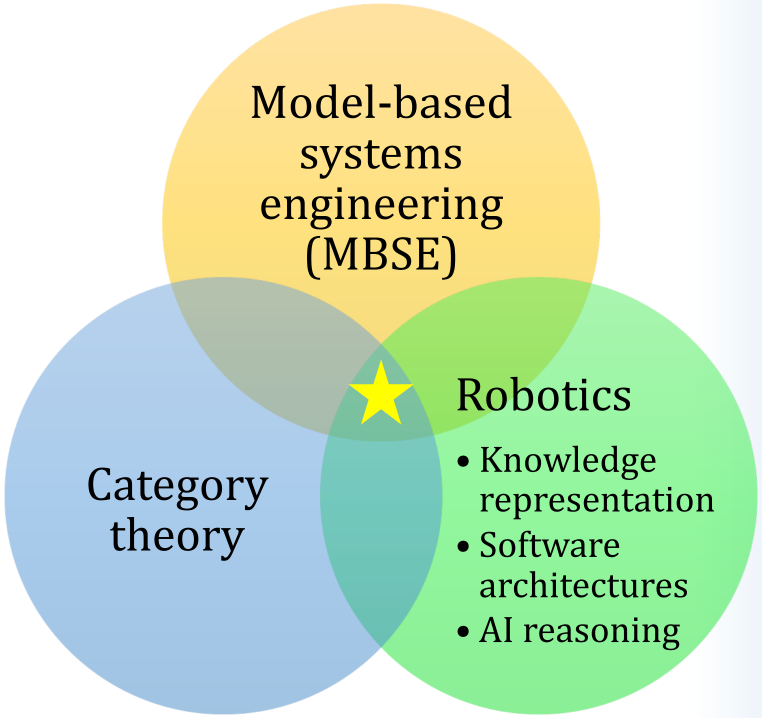
Some **structure-preserving relationship** between them

Formal semantic framework requirements

(a)	The ability to encode both procedural (task, motion, and control sequences) and declarative (knowledge) data.
(b)	The ability to track between abstraction levels (hierarchy).
(c)	The ability to encode composite (parts of a whole, decomposition, traceability) relationships and composition (merging, gluing, planning) relationships.
(d)	The ability to encode binary relations such as equivalence and inclusion.
(e)	The ability to adhere to constraints demanded by the internal syntax of knowledge, plans, and control.

Related work

- **MBSE** & **Robotics**
  - Platform independent model (PIM) and/or platform specific model (PSM) with model-to-model and model-to-text transformation methods to synthesize robotic implementations *(Heinzemann 2018, Bocciarelli 2019, Brugali 2016, Ruscio 2016, Bruyninckx 2013, Ringert 2015, Nordmann 2015, Wigand 2017, Steck 2011, Schlegel 2010, Hochgeschwender 2016)*
- **MBSE** & **Category theory**
  - Bidirectional model synchronization, state-based and delta-based lenses *(Diskin 2008, Diskin 2011, Diskin 2012)*
  - Model transformations with constraints *(Rutle 2010, Rutle 2012)*
  - Program synthesis using metamodels *(Batory 2008)*
- **Robotics** & **Category theory**
  - Symmetric monoidal categories for modeling robot program abstractions *(Aguinaldo 2020)*
  - Co-design applied to autonomous system design *(Zardini 2021 ECC, Zardini 2021 IROS)*

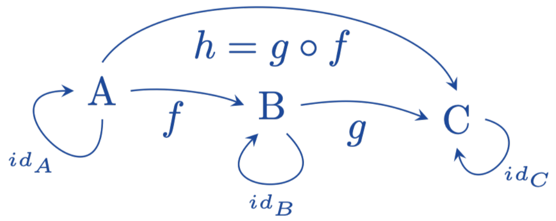


Contents

What is category theory?

Category theory is a branch of mathematics that provides mathematical structures whose properties are attentive to **composition of relationships**.

A category ($\mathbb{C}$) is:



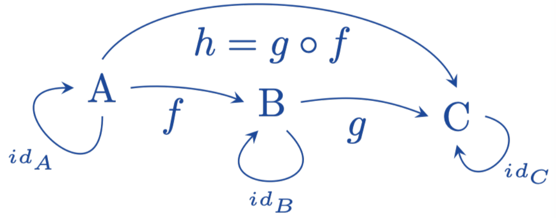
- A set of **objects** $\{A, B, C, \dots\}$
- A set of **morphisms** $\{f, g, h, \dots\}$ that map objects to objects
  - Where every object has an identity morphism, id_A
- **Composition operator**, $\circ$, between morphisms that is *associative* and has identity morphisms as *unitors*

What is category theory?

Category theory is a branch of mathematics that provides mathematical structures whose properties are attentive to **composition of relationships**.

A category ($\mathbb{C}$) is:

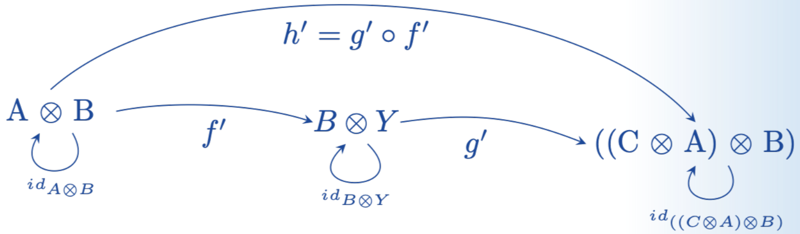
- A set of **objects** $\{A, B, C, \ldots\}$
- A set of **morphisms** $\{f, g, h, \ldots\}$ that map objects to objects
  - Where every object has an identity morphism, id_A
- **Composition operator**, $\circ$, between morphisms that is *associative* and has identity morphisms as *unitors*



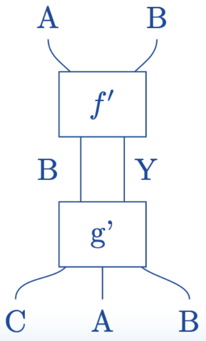
A symmetric monoidal category ($\mathbb{M}$), adds:

\+ **Tensor product**, $\otimes$, which is the product of $\mathbb{M}$ (objects and morphisms) with itself that is *associative* and has *unitor isomorphisms*

\+ Braiding isomorphism where $B_{\{X,Y\}}: X \otimes Y \rightarrow Y \otimes X$



A string diagram is the graphical syntax for symmetric monoidal categories, where **boxes are morphisms** and **strings are objects**.

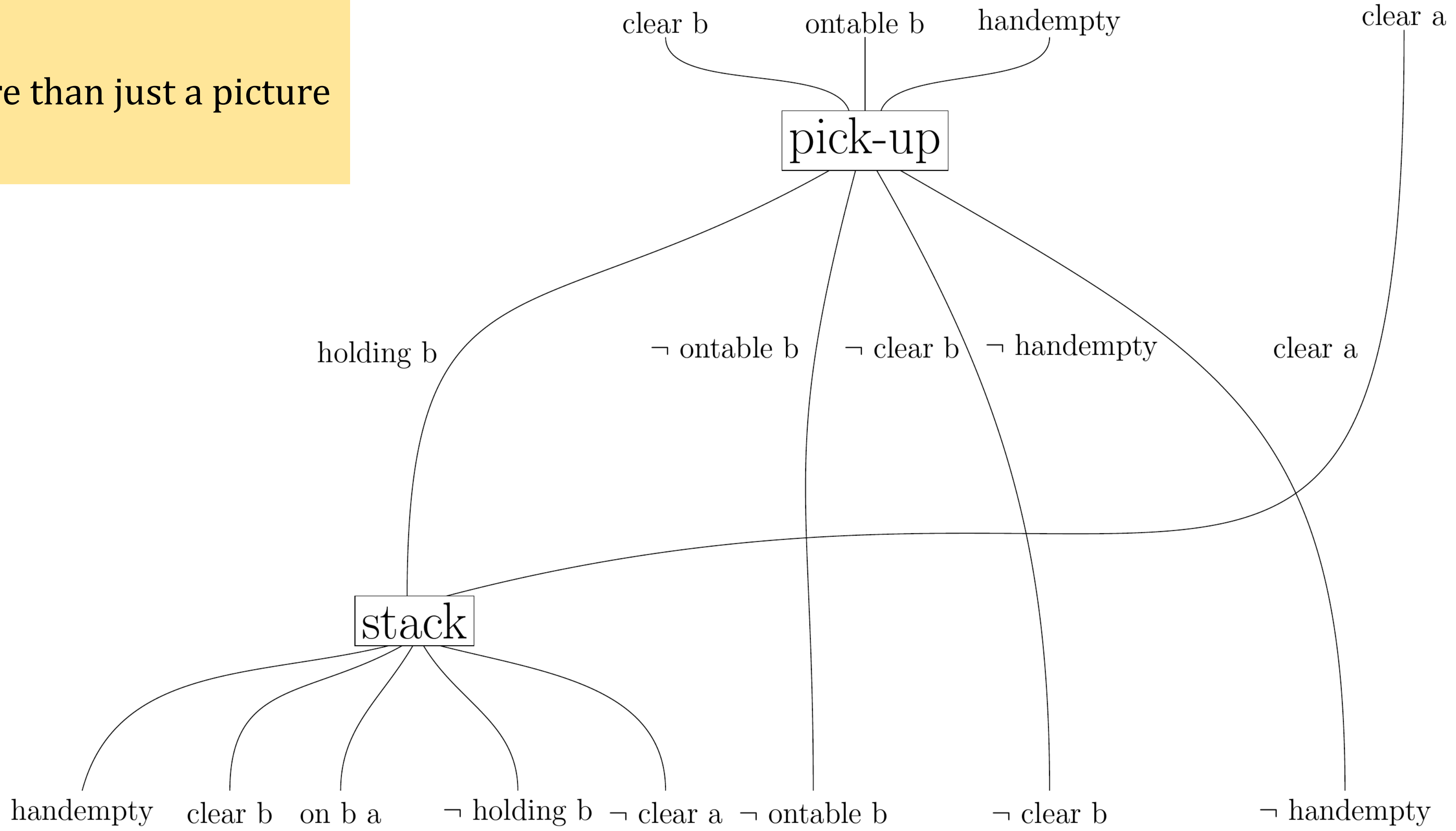


$$(id_C \otimes id_B \otimes id_A) \circ \\ g' \circ \\ (id_B \otimes id_Y) \circ \\ f' \circ \\ (id_A \otimes id_B)$$

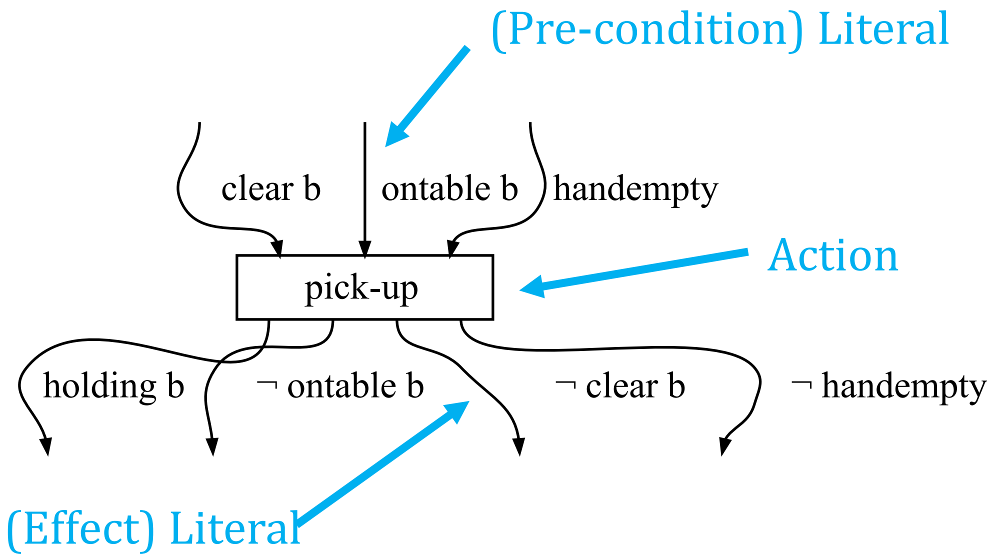
Contents

String Diagrams from Category Theory

! More than just a picture



String Diagrams for PDDL

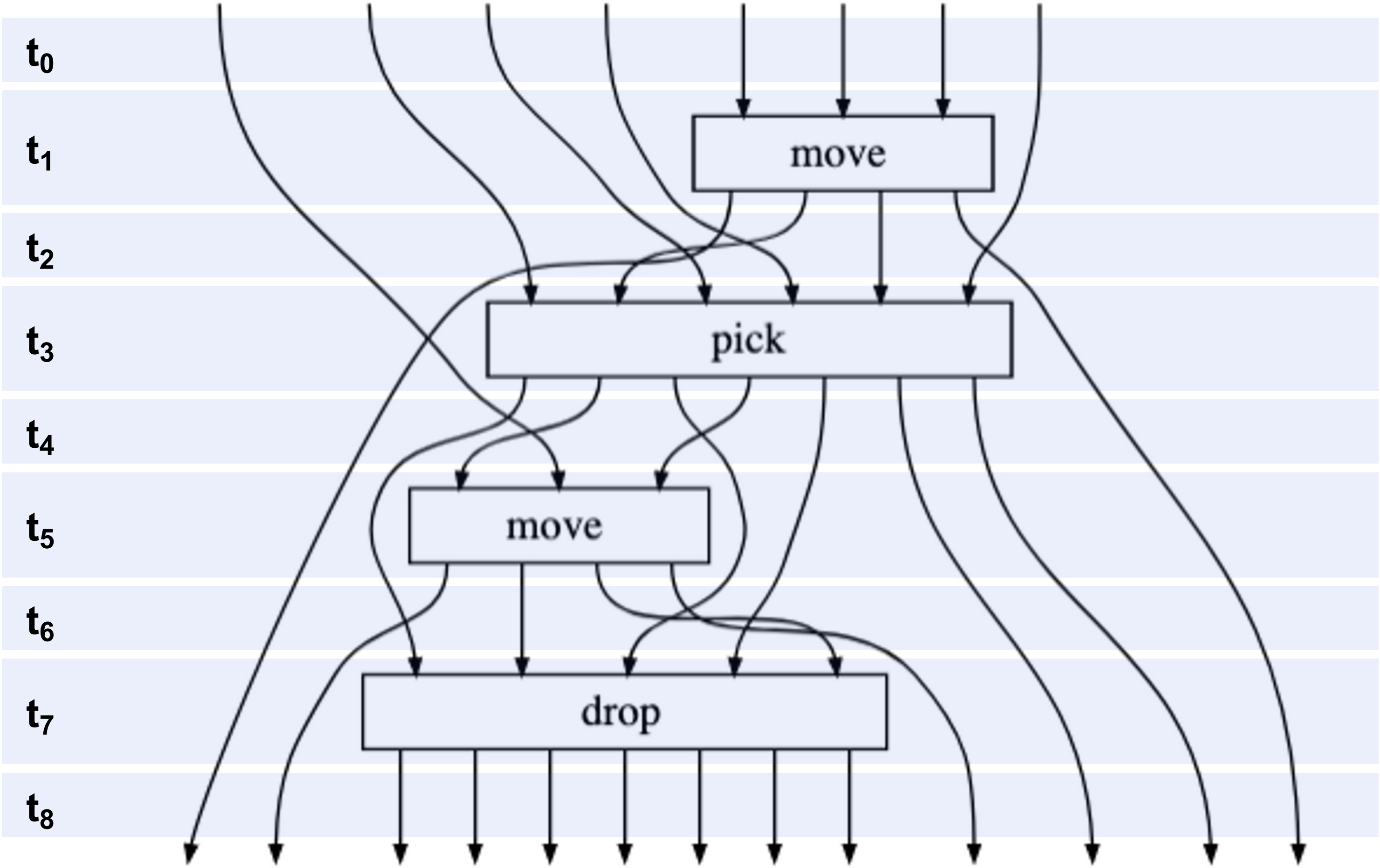


Categorification of Planning Solution

- Objects are **literals**
- Morphisms are **actions**
- Composition (∘) **chains actions**
- Tensor product (⊗) implies **parallel actions** or **conjunction of literals**

PDDL – Planning Domain Definition Language (McDermott 1998)

String Diagrams for Resource Tracking



String diagram with arbitrary time slices (t_0 - t_8) overlayed. At every time slice, we have complete knowledge of the data resources and/or function(s) running. Each slice can be re-interpreted in a linear mathematical syntax (not shown). Note, this is only one sample, discovered by the PDDL solver, from the larger valid solution space.

Visualize Classical AI Planning Solutions

Domain file

```
(define (domain BLOCKS)
(:requirements :strips)
(:predicates (on ?x ?y)
(ontable ?x)
(clear ?x)
(handempty)
(holding ?x)
)

(:action pick-up
:parameters (?x)
:precondition (and (clear ?x)
(ontable ?x) (handempty))
:effect
(and (not (ontable ?x))
(not (clear ?x))
(not (handempty))
(holding ?x)))
...
```

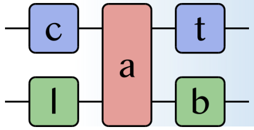
Problem file

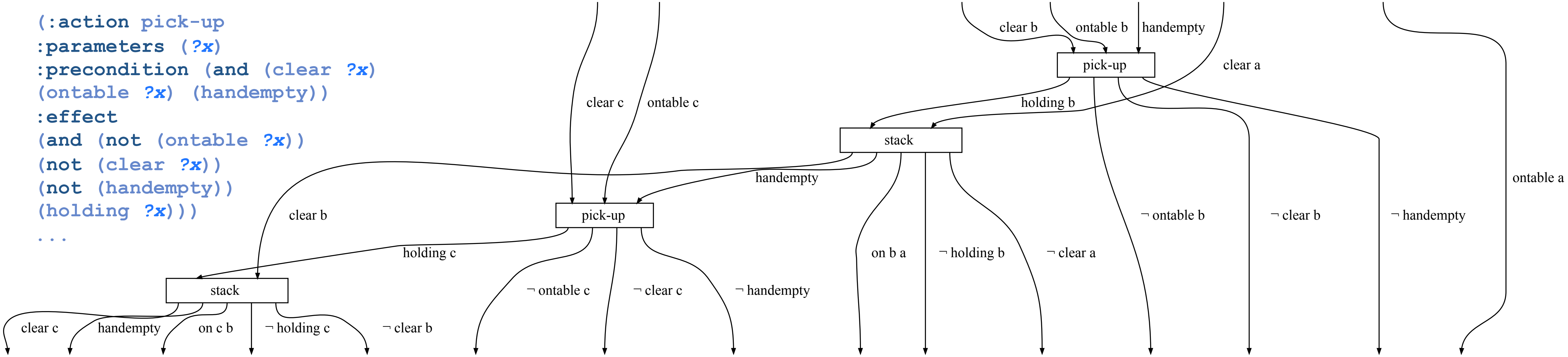
```
(define (problem BLOCKS-3-0)
(:domain BLOCKS)
(:objects a b c)
(:init (clear c) (clear a) (clear b)
(ontable c) (ontable a) (ontable b)
(handempty))
(:goal (AND (on c b) (on b a)))
)
```

Solution

```
pick-up b
stack b a
pick-up c
stack c b
```

Angeline Aguinaldo and William Regli. *Encoding Compositionality in Classical Planning Solutions.* International Joint Conference for Artificial Intelligence (IJCAI) Generalization in Planning Workshop. 2021.





Contents

What is a functor?

C. Translating Using Functors

The ability to translate the context from one abstraction level to another is a necessary step when compiling goal-oriented description to motion primitives. Category theory permits this concept via *functors*. Functors map objects and arrows between pairs of categories. If **X** and **Y** are categories, a functor, $F : \mathbf{X} \to \mathbf{Y}$ maps an object in **X** to some object in **Y** and maps each arrow between two objects in **X** to an arrow in **Y**, such that (9) and (10) are satisfied:

$$F(\mathrm{id}_{\mathbf{X}}) = \mathrm{id}_{F\mathbf{X}} \tag{9}$$

$$F(g \circ f) = Fg \circ Ff \tag{10}$$

where f and g are composable arrows in **X**.

Structure-preserving map between categories

Example

X and **Y** are categories where,

Objects($\boldsymbol{X}$) =
$\{A, B, C, D\}$

Arrows($\boldsymbol{X}$) =
$\{f: A \to \mathrm{B}, g: B \to \mathrm{C}, h: B \to D\}$

Objects($\boldsymbol{Y}$) =
{dog, cat, mouse, rabbit}

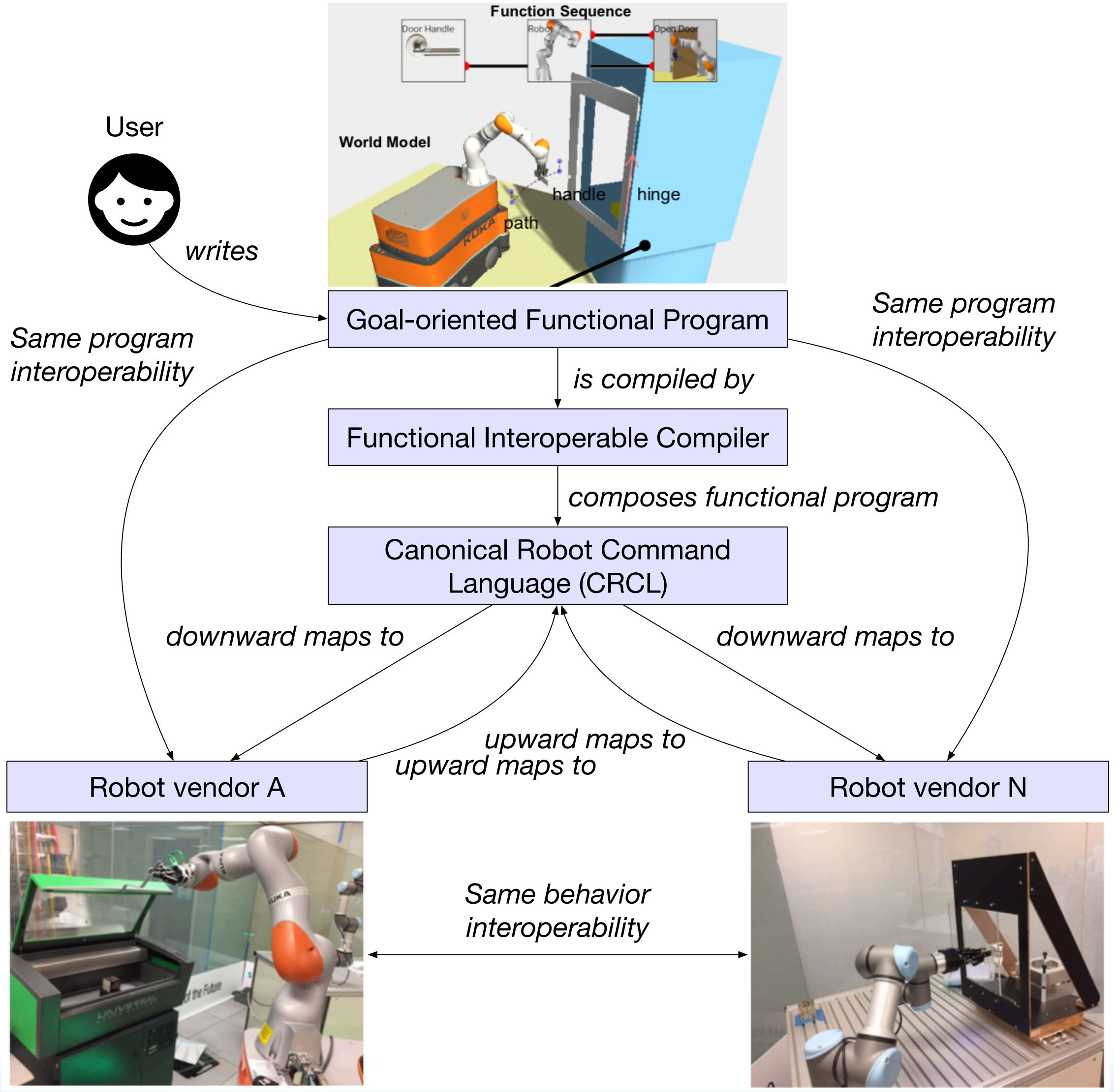
Arrows($\boldsymbol{Y}$) =
$\{f': \text{dog} \to \text{cat}, g': \text{cat} \to \text{mouse}, h': \text{cat} \to \text{rabbit}\}$

A possible functor, F, could be

objects	**arrows**	**identities**
$A \longmapsto$ dog	$f \longmapsto f'$	$id_A \longmapsto id_{\text{dog}}$
$B \longmapsto$ cat	$g \longmapsto g'$	$id_B \longmapsto id_{\text{cat}}$
$C \longmapsto$ mouse	$h \longmapsto h'$	$id_C \longmapsto id_{\text{mouse}}$
$D \longmapsto$ rabbit		$id_D \longmapsto id_{\text{rabbit}}$

Check
$F(g \circ f) = Fg \circ Ff = g' \circ f' =$ mouse

Goal-Oriented Robot Programming

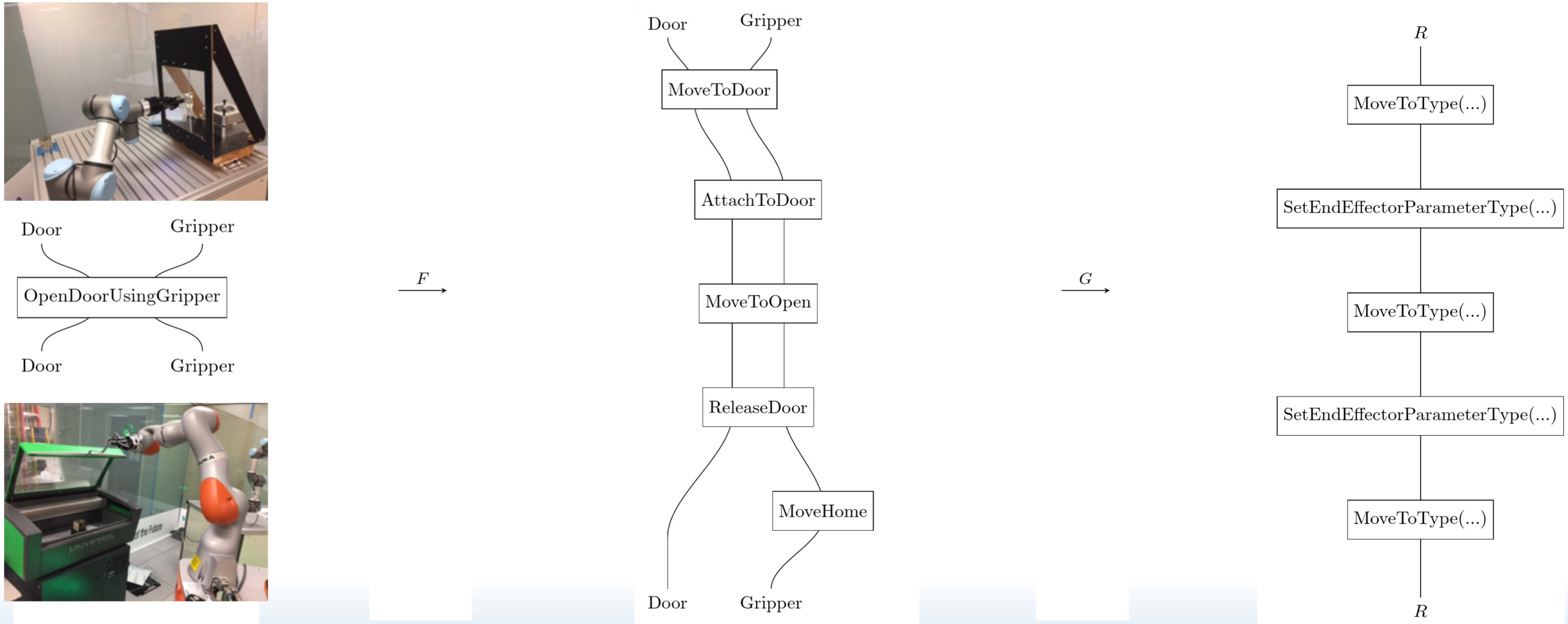


Goal-Oriented Robot Programming

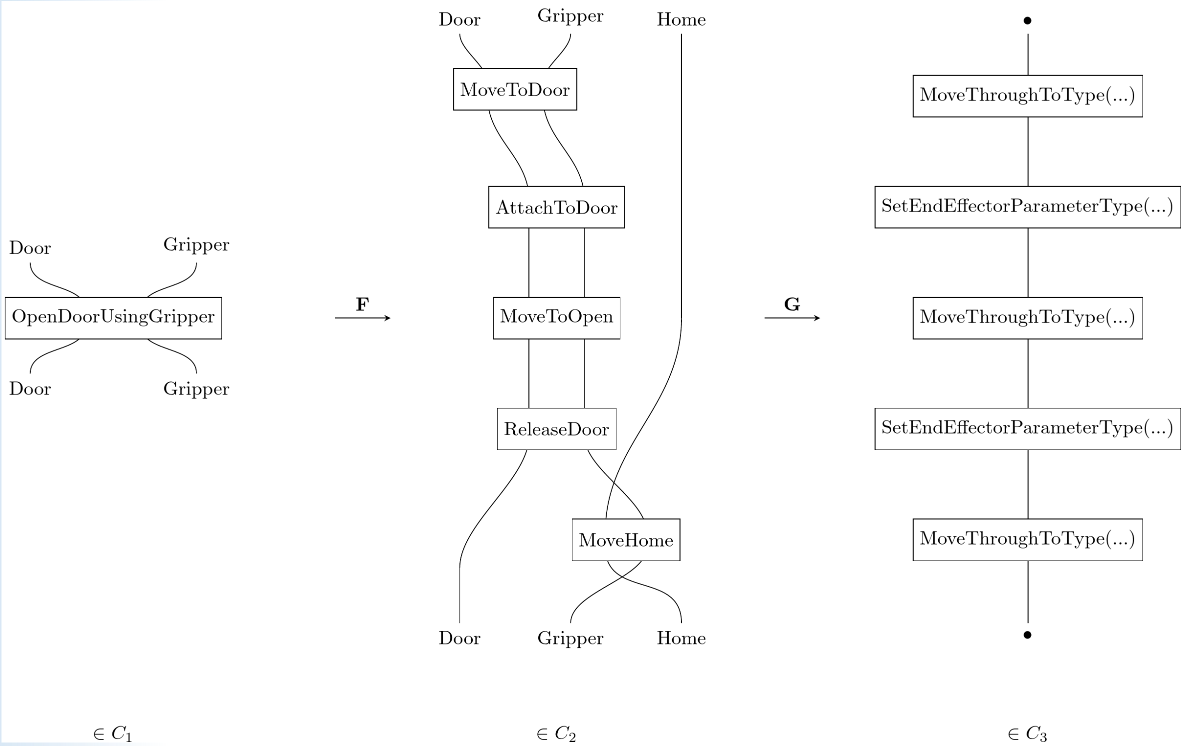
Physical
Identify types of physical resources needed to execute program. Name the program.

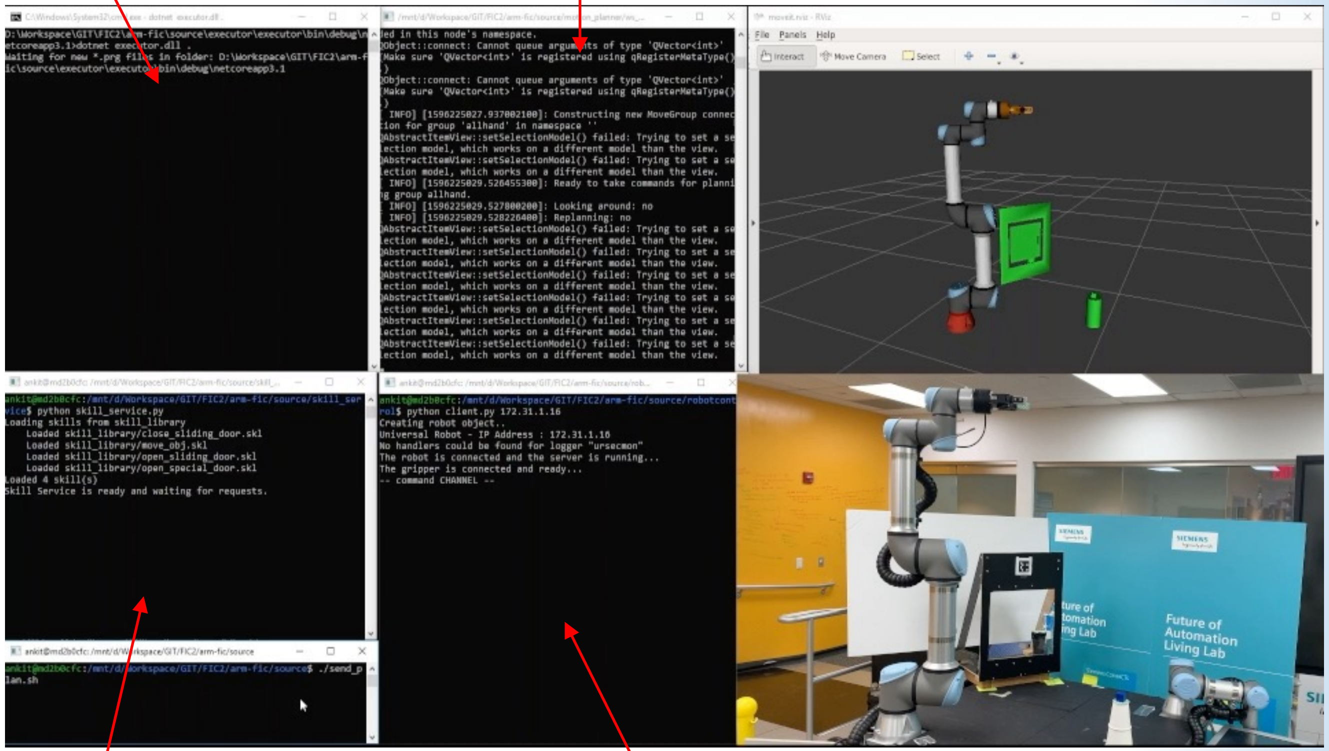
Software
Identify skills necessary to complete the desired action. Identify informational inputs and outputs for each skill.

Specification
Identify available command types and their possible parameters according to target robot command specification.



Goal-Oriented Robot Programming





A. Aguinaldo, J. Bunker, B. Pollard, A. Canedo, G. Quiros, W. Regli. *RoboCat: A category theoretic framework for robotic interoperability using goal-oriented programming.* IEEE Transactions for Automated Science and Engineering. 2022.

SIEMENS ARM ADVANCED ROBOTICS FOR MANUFACTURING

Contents

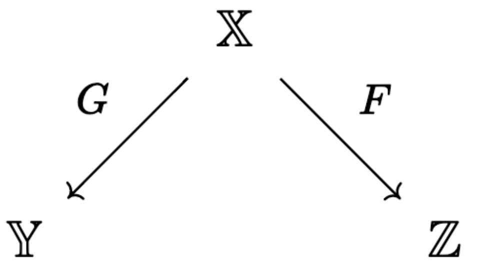
Synchronization within robot architectures

All robotic architectures involve some synchronization of knowledge, plan, and control

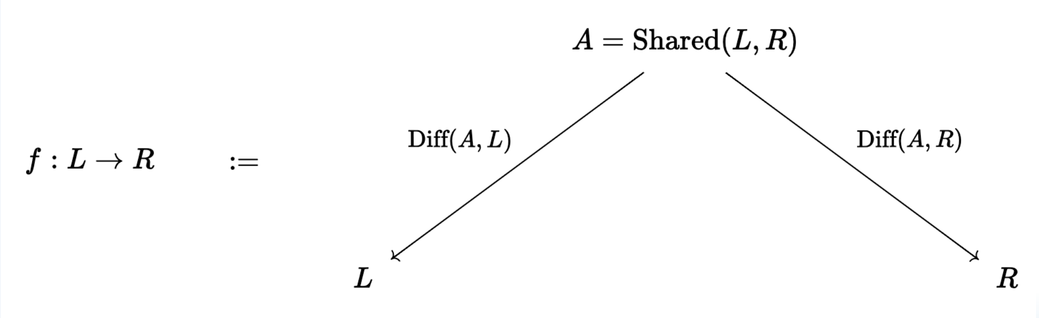
	Environment (world model, skills)	**Task/Plan (symbolic)**	**Control (programs, waypoints)**
Description	Environment refers to the world models such as what objects are present and where they are located, the symbolic actions the robot can accomplish, and its kinematic design.	Task and motion plans describe how the robot will achieve a goal by identifying a sequence of operations that symbolically update the state of the world.	Control refers to the low-level instructions given to the agent that tell it how to actuate.
Example syntax	ontologies, description logics, first-order predicate logic	hierarchical task nets (HTN), bi-partite directed acyclic graphs (DAGs), Markov decision processes (MDP)	finite state machines, directed graphs (control flow graphs, abstract syntax trees), petri nets
Example semantics	URDF, SDF, KNOWROB	STRIPS, PDDL plans	General purpose languages (C, C++, Python), robot controller languages (Kuka KRL, ABB RAPIDS, etc.)

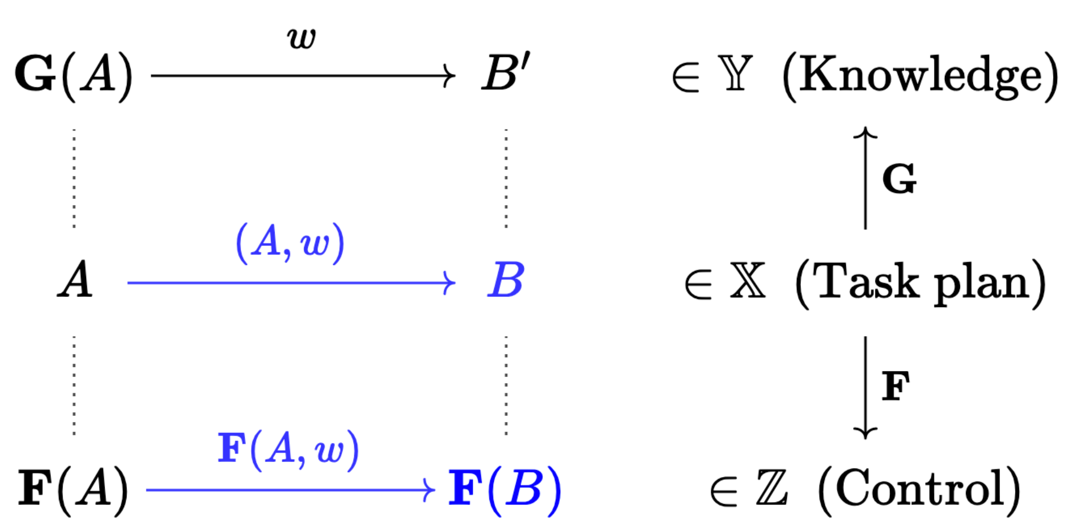
Contextual Attention Categorical Model

Symmetric delta lenses (Johnson 2017)
Spans in the category of small categories, **Cat**
- Left leg: Discrete opfibration functor, G
- Right leg: Arbitrary functor, F



Within each category, $(\mathbb{X}, \mathbb{Y}, \mathbb{Z})$
- Objects are models
- Arrows, f, are model updates (deltas)





Modeling capabilities
- **Traceability** is defined via functors, G and F
- **Change information** is captured via the span, or delta, construction for arrows
- **Synthesis** of new implementations, namely task plans and control programs, is computed automatically using the forward and backward propagation operations

Aguinaldo A., Regli W. Modeling traceability, change information, and synthesis in autonomous system design using symmetric delta lenses. ICRA Compositional Robotics Workshop 2022.

Category of Knowledge Configurations

$\mathbb{D}$ is an Olog category (Spivak 2012), the syntactic category for databases, where objects are types and arrows are relations and properties.

Objects

$I\colon \mathbb{D} \to \mathbf{Set}$

$J\colon \mathbb{D} \to \mathbf{Set}$

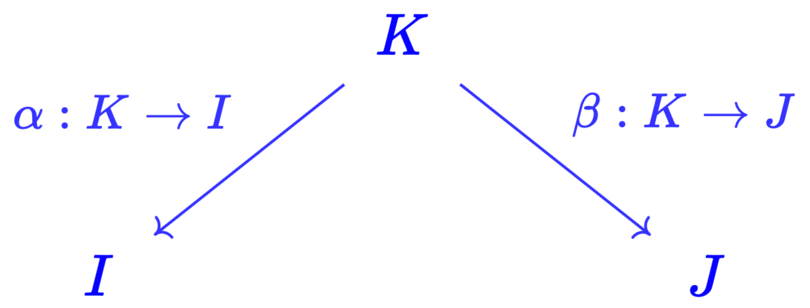
…

$K\colon \mathbb{D} \to \mathbf{Set}$

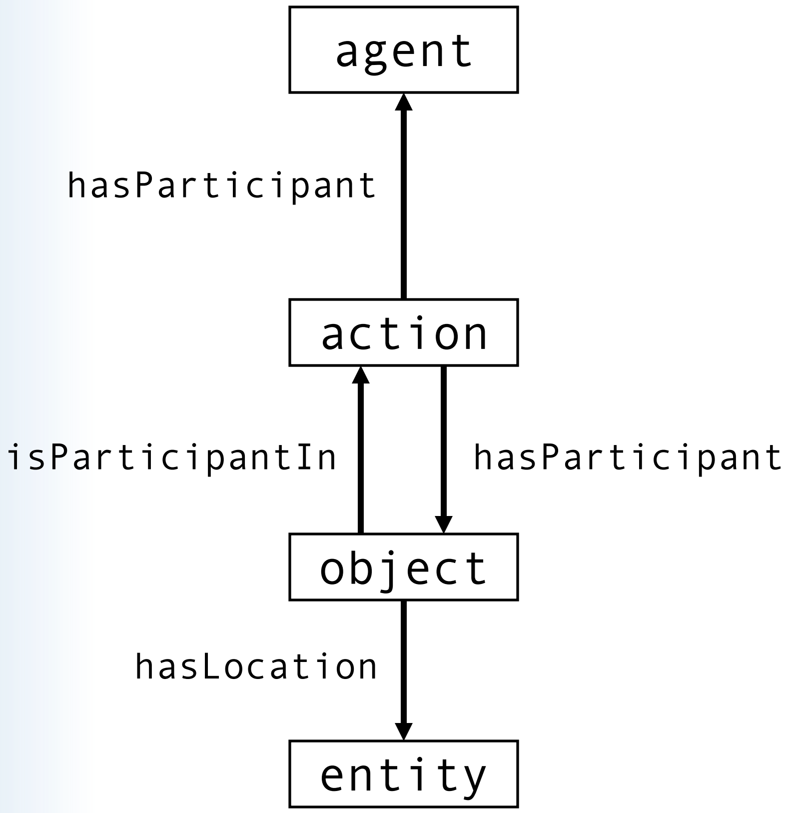
where $I, J, \ldots K \in \mathbf{D} - \mathbf{Inst}$ map to sets with the empty element

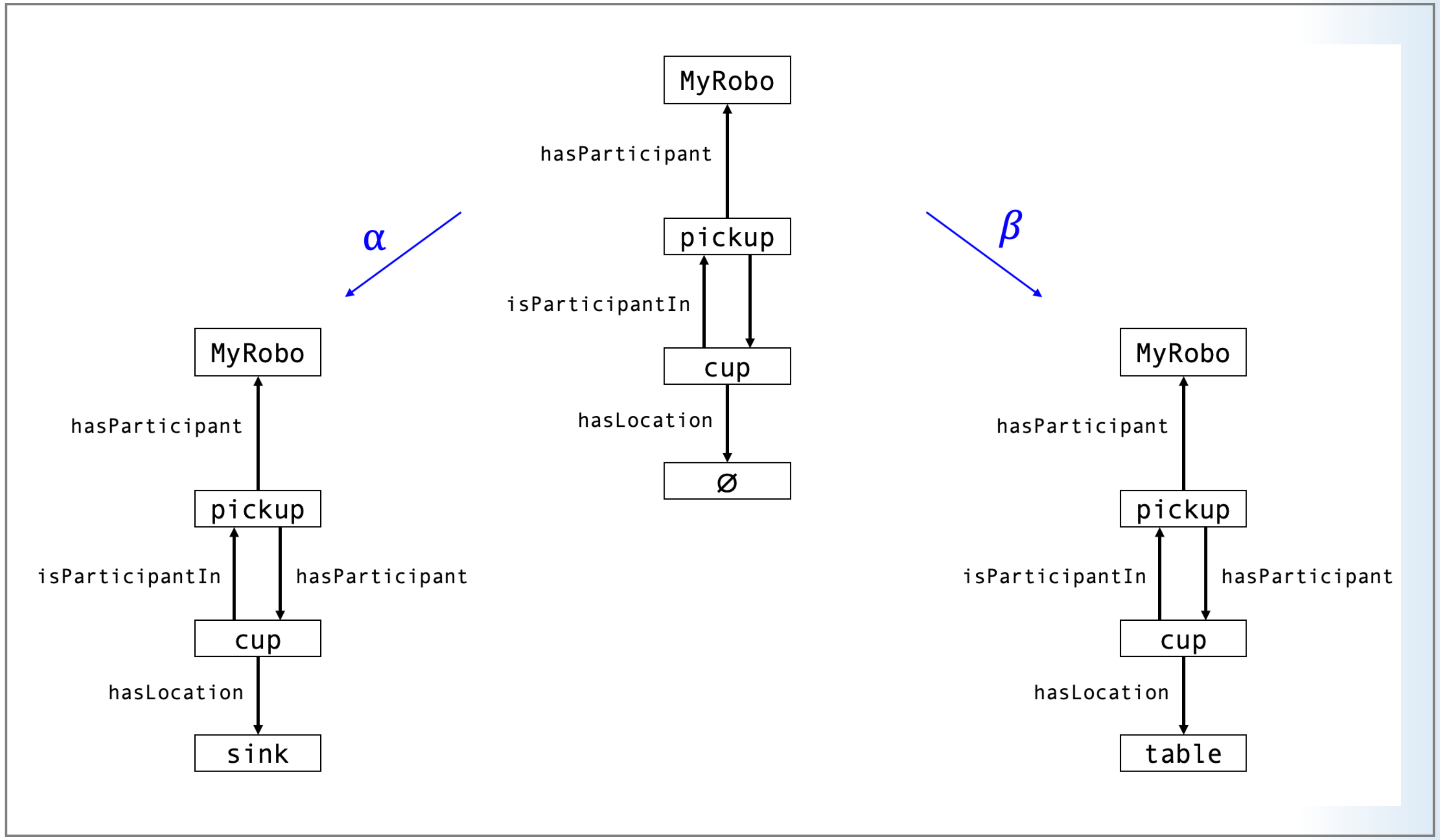
Arrows

$f : I \to J$



Category of Knowledge Configuration (Example)





Based on KNOWROB ontology (Tenorth 2015)

Category of Plans

$\mathbb{T}$ is the category of monoidal categories. Functors between monoidal categories preserve the monoidal structure.

Objects

$\text{Monoidal}(X_1, A_1, \otimes)$

$\text{Monoidal}(X_2, A_2, \otimes)$

...

$\text{Monoidal}(X_n, A_n, \otimes)$

where,

X_i ∈ Set of possible predicates in the world

and

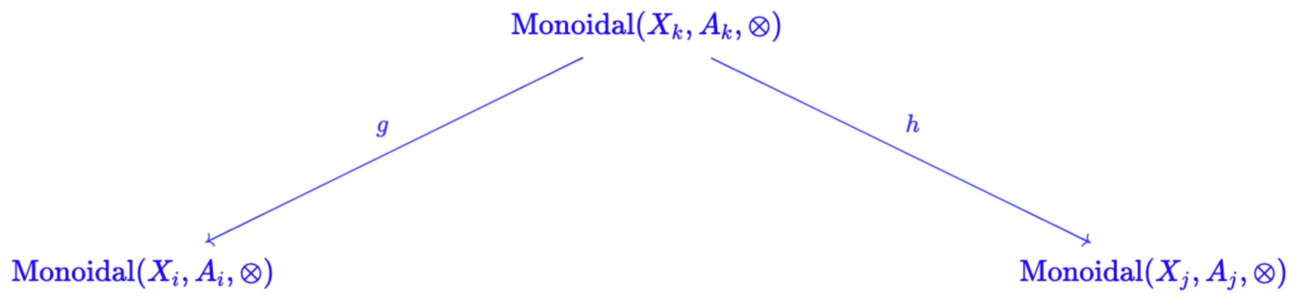
A_i ∈ Set of possible actions in the world that transition states in the world

and

$\otimes$ is the conjunction of predicates and actions

Arrows

$f : \text{Monoidal}(X_i, A_i, \otimes) \rightarrow \text{Monoidal}(X_j, A_j, \otimes)$



Future Work

- ❑ What **functors, G and F,** can be defined between the proposed categories?
    - Do G and F meet the requirements of symmetric delta lenses?
- ❑ Are all items in the **formal semantic framework requirements (a)-(e)** met in this framework?
- ❑ What **other properties** does this framework afford us?
    - Can we make a statement about whether a reasoning engine is more capable than another given the same information using this framework?
- ❑ How might we **implement this framework on a computer**? What is the computational complexity of these queries?

Contents

Flexible and adaptable formal semantics

Category theory as conceptual stem-cell

Category theory (CT) can differentiate into many forms:

- All forms of pure math... (we'll briefly discuss this)
- Databases and knowledge representation (categories and functors)
- Functional programming languages (cartesian closed categories)
- Universal algebra (finite-product categories)
- Dynamical systems and fractals (operad-algebras, co-algebras)
- Hierarchical planning (lenses and monads)
- Shannon Entropy (operad of simplices)
- Partially-ordered sets and metric spaces (enriched categories)
- Higher order logic (toposes = categories of sheaves)
- Measurements of diversity in populations (magnitude of categories)
- Collaborative design (enriched categories and profunctors)
- Petri nets and chemical reaction networks (monoidal categories)
- Quantum processes and NLP (compact closed categories)

2 / 32

David Spivak: https://applied-compositional-thinking.engineering/wp-content/uploads/2021/01/ETH2021-ACT4E.pdf

Tooling in development

Autogenerate PDDL String Diagrams

String diagrams are a graphical language used to describe symmetric monoidal categories (SMCs) from category theory. They can be seen as mathematical rigorous expressions to describe processes and their dependencies. In this notebook, we use string diagrams to express the solutions to Planning Domain Definition Language (PDDL) problems. More specifically, we seek to observe if the string diagram representation can elucidate interesting properties of robot manipulator program plans in a manufacturing work cell. This code uses the WiringDiagram Catlab Julia library to construct the string diagrams. In these examples, the objects are considered to be Boolean expressions and the arrows, or morphisms, are the PDDL actions.

In [1305]:

```
# SOFTWARE PRE-REQ
#
# Julia 1.3.1
# Catlab 0.5.3
# Latex

using Catlab.WiringDiagrams
using Catlab.Doctrines
using Catlab

using Catlab.Graphics
import Catlab.Graphics: Graphviz

import TikzPictures
using Catlab.Graphics
```

In [1306]:

```
EXAMPLE = "blocksworld";
```

Process PDDL Files and PDDL solution

To run this notebook, you must provide the name of directory (in `examples/`) containing domain.pddl, problem.pddl, and solution.txt in the `EXAMPLE` variable (above), then run *all* cells. The composed string diagram is shown as the output of the last cell. It can also be seen as an SVG in `smc.dot.svg`.

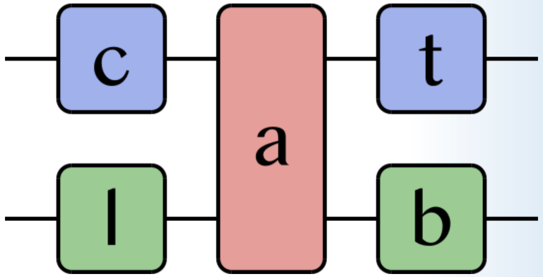
About files

The `domain.pddl` and `problem.pddl` files must adhere to PDDL specifications following the `:strips` requirement.

The `solution.txt` file should be a newline for each action with parameters provided by a PDDL planner of choice. An example is shown below:

```
move lochome locbox2
pick boxa locbox2 grippera
drop boxa locbox2 grippera
```

One possible way to obtain a PDDL solution is to run PDDL4j solver, using



https://github.com/AlgebraicJulia/Catlab.jl

Thank you for listening!

Angeline Aguinaldo

aaguinal@cs.umd.edu

Please feel free to ask questions and provide feedback during 1-1s or via email.