The Role of Higher-order Models in Robotics and its Reasoning Challenges

Herman Bruyninckx, RobMoSys project
KU Leuven – TU Eindhoven

MODELS 2019, 17 September, 2019, München

RobMoSvs

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732410.

RobMoSys' five levels of modelling

1. **Abstraction**: guidance for humans, by following harmonized interpretation of abstractions.
2. **Reuse & Flexibility**: reuse and customization of robotics software assets, via **data sheets**.
3. **Predictability**:
   composition is *correct by construction*.
4. **Automation**: automate labor-intensive stuff:
   Validation & Verification, code generation,...
5. **Autonomy**: models at run-time.
   self-configuration & -adaptation, explanation,...

What is "higher-order modelling"?

model:

- set of **entities** connected by **relations**.
- **data structure** for each entity & relation.

→ **property graph** (or "entity-relation" graph)

higher-order model:

- set of **relations** on top of other **relations**,
- with (partially ordered) **hierarchy** in the **semantics** of the relations.

→ one sees the hierarchy in the **directed** graph structure, and in the **properties** of the relations.

Relevant for RobMoSys' modelling levels 3–5.

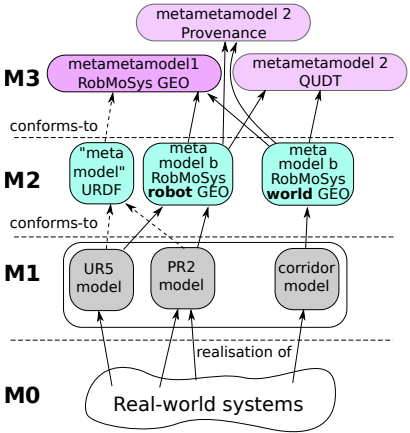
Added value higher-order modelling?

- **knowledge representation** of the domain:
  - data $\rightarrow$ **information** $\rightarrow$ knowledge
  - **composability** and **compositionality**:
    to combine pieces of knowledge in "the right way"
- **reasoning**: to explain, to generate, to monitor

- **sofware** brings *knowledge representation* too:
  - **configuration**: *gazillions* of "magic numbers" to be combined when composing components.
  - **coordination**: generate *task state machines* at runtime, because *gazillions* of contextual requirements require different **interaction behaviour** of components.

Major example

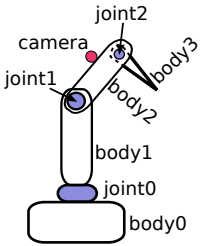
M0–M3 meta model of model-driven engineering

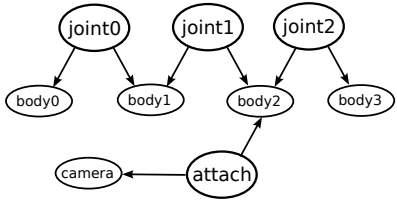


The **higher order** represented here is:

- M1–M3 relations are **relative**; "hierarchy" can be extended "upwards" indefinitely.
- level n models the constraints that must be satisfied in a model at level $n-1$.
- allows **translation** between (meta) models, for their **conforming** parts.
- typically, that knowledge is used by **humans** using a **tool chain**.
- dream of robotics: develop **robots** that can use that knowledge, themselves.

Robotics example: motion stack
Most abstract model: mereo-topology

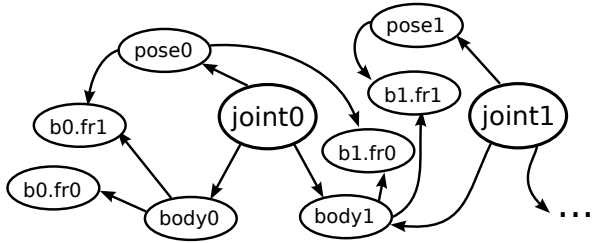




The **model** represents:

- **parts** in the model, and
- **connections** between those parts.

One more concrete level: motion constraint relation

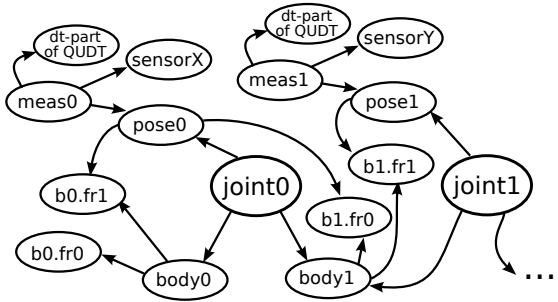


The extra higher-order **model** represents:

- joint is a **motion constraint** between robot's links
- at every moment in time, two links have a relative **pose** whose properties depend on the type of the joint constraint

→ mathematical constraints between positions on connected body points.

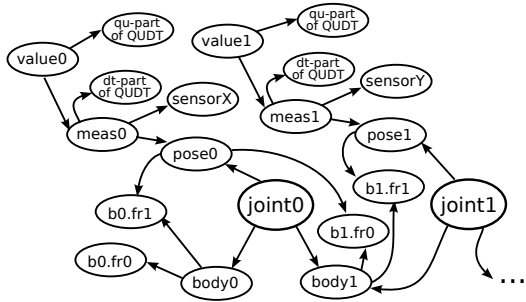
Yet one more concrete level: pose measurement type



The extra higher-order **model** represents:

- the pose is **measured** by sensors
- it has a **d**imension and **t**ype
- **QUDT** is a standard meta model for this purpose

Yet one more concrete level: pose measurement values



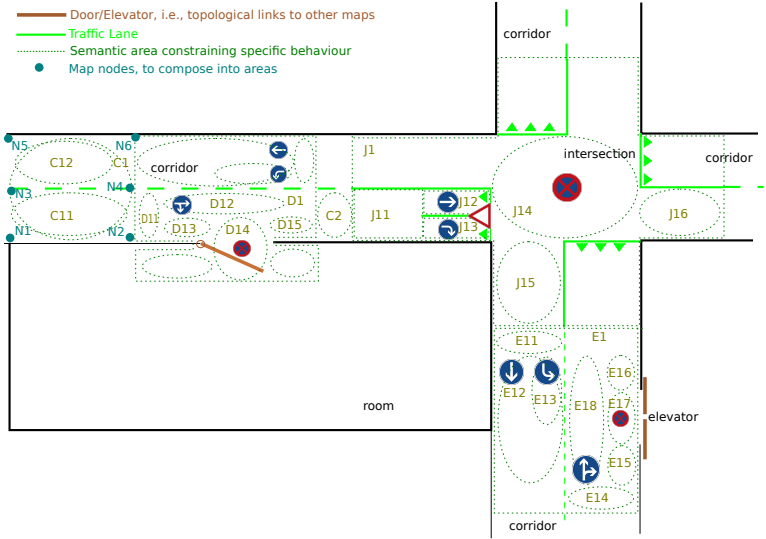
The extra higher-order **model** represents:

- measurement of pose gives **numerical values**.
- those **q**uantities have physical **u**nits
- **QUDT** is a standard meta model for this purpose

Queries on such higher-order models

- *raise an event when camera speed is below "motion blur" limit*
- *can my software components exchange velocity data with yours?*
- *if not, which software component can provide the missing translation between both coordinate representation?*
- *generate the composite kinematics solver when* `Arm_xyz` *is put on top of* `MobileBase_123`
- *generate a dynamics solver that adapts to a 1kHz torque control loop around it, and to the accuracy of the sensors*

Robotics example: semantic map



Queries on such higher-order models

- *raise event if robot is near high-risk area*
- *can my software component interpret all semantic tags relevant for a given task?*
- *if not, which software component can provide the missing translation between the map's semantics and my software components data sheet?*
- *generate the task graph to move from* `Area_xyz` *to* `Area_123`*, while maintaining safe behaviour against all expected other users of the building*

We're not there yet. . . !

Nevertheless:

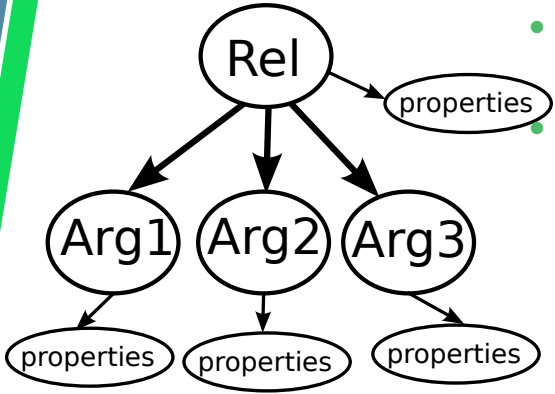
- problem investigated for **50 years** already. . .
- the **market pull** is tremendous. . .
- all **researchers** claim they provide solutions. . .

Why?

- mainstream focus: **sofware only**.
- higher-order modelling is **tough**;
  developing **query solvers** even more.
- software components: too limited **compositionality** via models.
- . . .

Mechanism of higher-order modelling

`Entity-Relation` models



- *entities* are the **arguments** in *relations*.
- supporting data structure: **property graph**: every node in the graph has:
  - a **property** data structure.
  - a **list of outgoing edges**.
  - a **list of incoming edges**.
  - a `Semantic_ID`: its own ID + IDs of its **meta model** + **list** of **meta meta model** IDs

Mechanism (continued)
Reasoning

- **query** is a property graph in itself:
  - sub-graph of **entities** one wants information about,
  - when constrained by specified **relations** in the graph,
  - and extra **constraint** relations introduced by the query.

  For example: *"Give all places on the map close to the robot, and observable by its sensors"*.
- **solver** base on **graph traversal**:
  - represents the **knowledge** required to travel through the graph in the "right way" to find the answer to the query.
  - is **also** property graph, of **higher order**!
- **state of the art**: close to nowhere, still...
  RobMoSys: develops **platform** infrastructure.

Major modelling errors

- using `is-a` instead of `conforms-to`.
  E.g.: a mobile robot *is not* a robot, but it shares a lot of the behaviour

- using *property* (`has-a`) instead of *attribute* (property of argument in relation)
  E.g.: a robot *does not have* a position value, because that is the property of a relation between the robot and its environment

- **reification** is not a *first-class citizen*:
  - every relation becomes an entity in itself

  $\rightarrow$ can become part of higher-order relation itself!

  Examples of "modelling" languages that fail here:
  UML, StateCharts, SysML, OWL, Prolog, Lisp,...

Developments in RobMoSys

Cgraph library (from `graphviz` ecosystem):

- efficient `C`-library for property graphs

→ **in-memory** reasoning possible.

- query formulation and graph traversal solving are **not** supported...

Application focus, for now:

- geometric world model, with semantic labels.
- kinematic and dynamic solvers, with control and estimation around it.
- semantic localization at "platform" level.

JSON-LD: identified as (non-exclusive!) *best fit* for purpose of serialisation and file format:

- supports `Semantic_ID` out of the box.

Conclusions

- robotics has **a lot of context**, hence higher-order modelling is a must.

→ surprisingly few results in that direction...

- modelling remains an **art**.
  querying and **solving** queries even more so!
  - especially for the **higher-order** models,
  - connected to **system-wide** dependencies.

→ those are where *the money* is!

→ **business models** feasible,
even on top of fully *open source* models!

Thank you for your attention