Knowledge Structures over Simulation Units

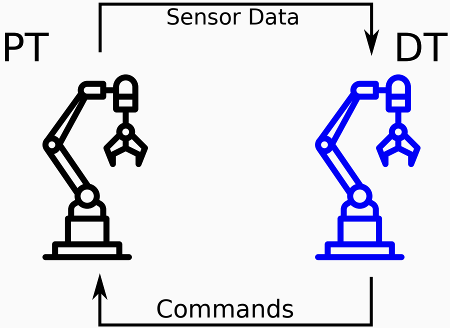
Eduard Kamburjan
Einar Broch Johnsen

University of Oslo
ANNSIM 2022

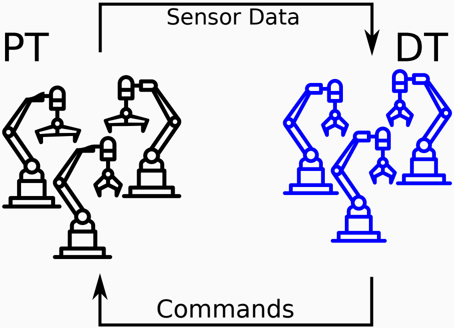
Do you know what your digital twin is twinning?

Introduction

Do you know what your digital twin is twinning?

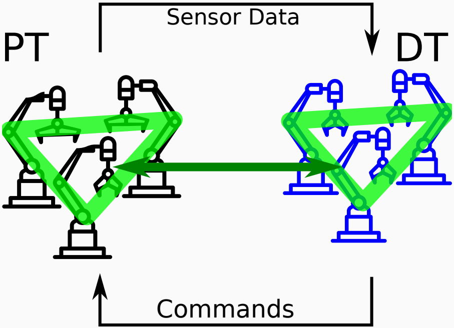


Do you know what your digital twin is twinning?



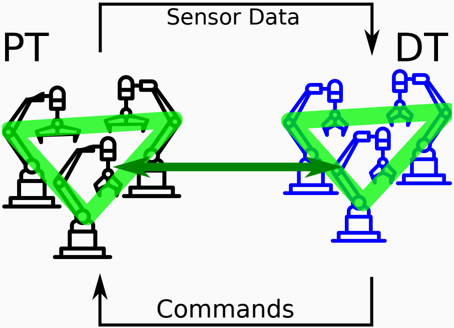
Introduction

Do you know what your digital twin is twinning?



Introduction

Do you know what your digital twin is twinning?



- Common data representation
- Data view on both twins
- Twinning as data property

Knowledge Structures

Knowledge Graphs

UNIVERSITY OF OSLO

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

Knowledge Graphs

UNIVERSITY OF OSLO

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

W3C Standards

RDF for data, OWL for knowledge, SPARQL for queries.

Knowledge Graphs

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

W3C Standards

RDF for data, OWL for knowledge, SPARQL for queries.

RDF:
```
Peter a Person. Paul a Person. Maria a Person.
Peter hasChild Paul. Paul hasChild Maria.
```

UNIVERSITY OF OSLO

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

W3C Standards

RDF for data, OWL for knowledge, SPARQL for queries.

RDF:

```
Peter a Person. Paul a Person. Maria a Person.
Peter hasChild Paul. Paul hasChild Maria.
```

OWL:

`GrandParent` **subClassOf**

`hasChild` **some** (`hasChild` **some** `Person`)

Knowledge Graphs

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

W3C Standards

RDF for data, OWL for knowledge, SPARQL for queries.

RDF: `Peter a Person. Paul a Person. Maria a Person.`
`Peter hasChild Paul. Paul hasChild Maria.`

OWL: `GrandParent` **subClassOf**
`hasChild` **some** (`hasChild` **some** `Person`)

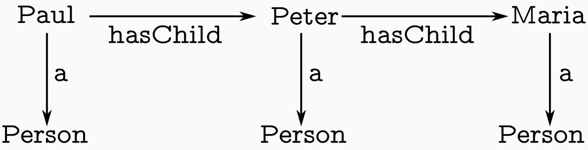
SPARQL: `SELECT ?x WHERE { ?x a GrandParent }`

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

W3C Standards

RDF for data, OWL for knowledge, SPARQL for queries.



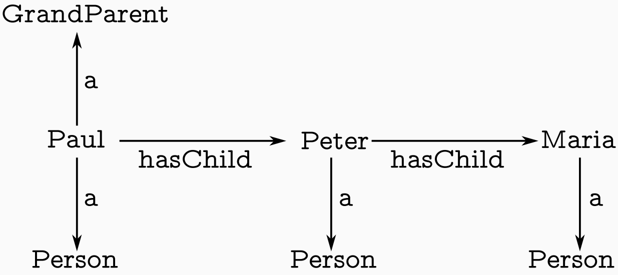
Knowledge Graphs

Triple-Based Knowledge Representation

Knowledge Graphs are a framework to (a) represent, (b) reason over, and (c) query domain knowledge and data.

W3C Standards

RDF for data, OWL for knowledge, SPARQL for queries.



Contributions

UNIVERSITY OF OSLO

Combining the Knowledge

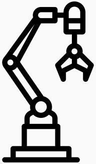
- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge

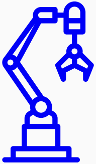
Contributions

UNIVERSITY OF OSLO

Combining the Knowledge

- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge



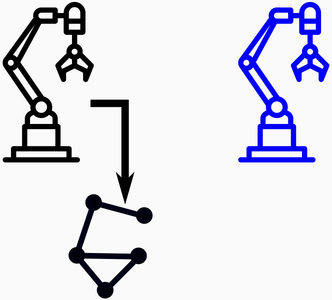


Contributions

UNIVERSITY OF OSLO

Combining the Knowledge

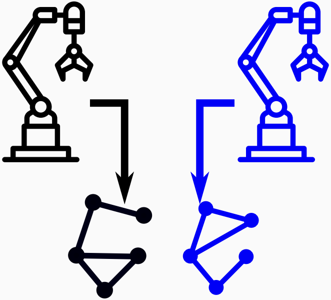
- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge



Contributions

Combining the Knowledge

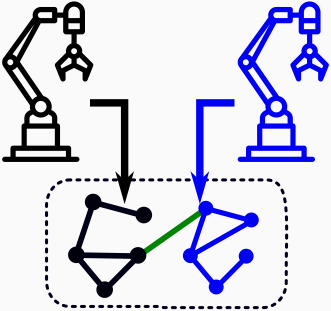
- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge



Contributions

Combining the Knowledge

- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge



Combining the Knowledge

- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge

Possible Constraints

- Constraint on asset model
  "Is the asset model consistent?"
- Constraint on program
  "Is this a sensible simulation structure?"
- Constraints on twinning
  "Does the program have the same structure as the asset?"

Knowledge Graphs and Asset Models

Asset Model

An asset model is an organized, digital description of the composition and properties of a physical asset.

Our Asset Model

A knowledge graph describing the structure of the physical twin.

Asset Model

An asset model is an organized, digital description of the composition and properties of a physical asset.

Our Asset Model

A knowledge graph describing the structure of the physical twin.

```
ast:heater1 a ast:Heater. ast:heater1 ast:in ast:room1.
ast:heater2 a ast:Heater. ast:heater2 ast:in ast:room2.
ast:heater1 ast:id 13. ast:heater2 ast:id 12.
ast:room1 ast:leftOf ast:room2.
```

Asset Model

An asset model is an organized, digital description of the composition and properties of a physical asset.

Our Asset Model

A knowledge graph describing the structure of the physical twin.

```
ast:heater1 a ast:Heater. ast:heater1 ast:in ast:room1.
ast:heater2 a ast:Heater. ast:heater2 ast:in ast:room2.
ast:heater1 ast:id 13. ast:heater2 ast:id 12.
ast:room1 ast:leftOf ast:room2.
```

`htLeftOf` **subPropertyOf** `ast:in` **o** `ast:leftOf` **o** **inverse**(`ast:in`)

Programs as Knowledge Graphs

Additionally to the data of the asset/physical twin, we can interpret the program state as data of the digital twin.

Knowledge Graphs and Programs

UNIVERSITY OF OSLO

Programs as Knowledge Graphs

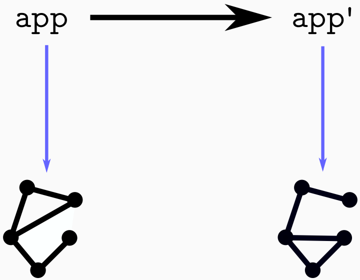
Additionally to the data of the asset/physical twin, we can interpret the program state as data of the digital twin.

$$\texttt{app} \longrightarrow \texttt{app'}$$

UNIVERSITY OF OSLO

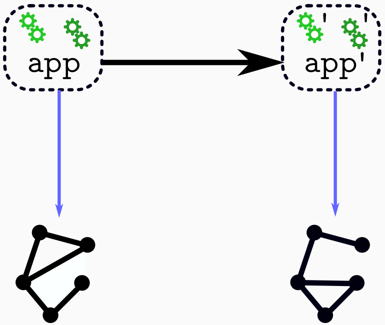
Programs as Knowledge Graphs

Additionally to the data of the asset/physical twin, we can interpret the program state as data of the digital twin.



Programs as Knowledge Graphs

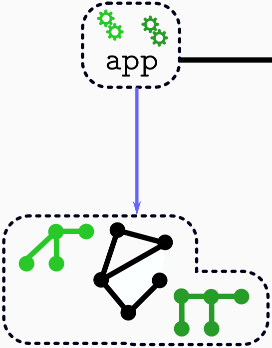
Additionally to the data of the asset/physical twin, we can interpret the program state as data of the digital twin.



UNIVERSITY OF OSLO

Programs as Knowledge Graphs

Additionally to the data of the asset/physical twin, we can interpret the program state as data of the digital twin.

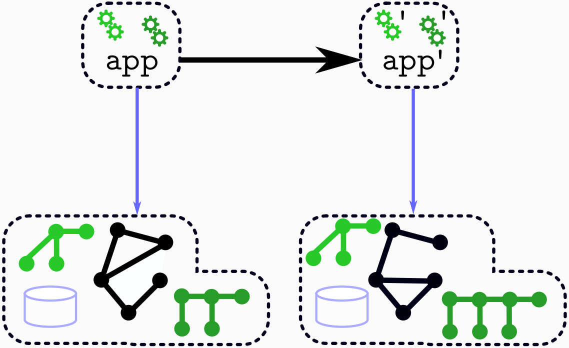


Knowledge Graphs and Programs

UNIVERSITY OF OSLO

Programs as Knowledge Graphs

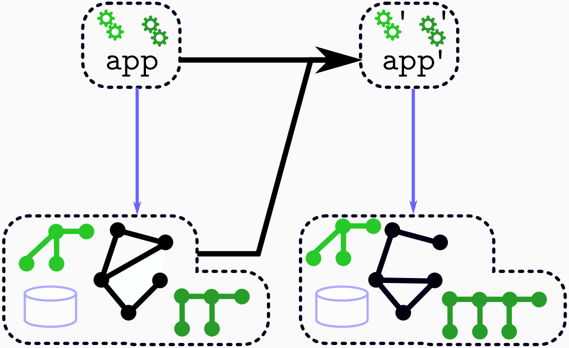
Additionally to the data of the asset/physical twin, we can interpret the program state as data of the digital twin.



Knowledge Graphs and Programs

Programs as Knowledge Graphs

Additionally to the data of the asset/physical twin, we can interpret the program state as data of the digital twin.



UNIVERSITY OF OSLO

SMOL: Integration of Programs and Knowledge

Map each program state to a knowledge graph and allow program to operate on the KG. Implemented in SMOL (`smolang.org`).

```
class C (Int i) Unit inc() this.i = this.i + 1; end end
main C c = new C(5);  Int i = c.inc(); end
```

UNIVERSITY OF OSLO

SMOL: Integration of Programs and Knowledge

Map each program state to a knowledge graph and allow program to operate on the KG. Implemented in SMOL (`smolang.org`).

```
class C (Int i) Unit inc() this.i = this.i + 1; end end
main C c = new C(5);  Int i = c.inc(); end
```

```
prog:C a prog:class.  prog:C prog:hasField prog:i.
run:obj1 a prog:C.    run:obj1 prog:i 5.
....
```

Simulation Units

FMI

Functional Mock-Up Interface (FMI)

Standard for exchange of black-box (co-)simulation units, called function mock-up units (FMUs).

- Directly exportable from simulation frameworks
- Wrapper around existing simulators
- Can also serve as interface to sensors and actuators.

UNIVERSITY OF OSLO

Functional Mock-Up Interface (FMI)

Standard for exchange of black-box (co-)simulation units, called function mock-up units (FMUs).

- Directly exportable from simulation frameworks
- Wrapper around existing simulators
- Can also serve as interface to sensors and actuators.

Model Description

The FMI defines a set of functions on an FMU (e.g., advance time via `doStep`) and a format for the interface of the FMU.

Model Description

```
<fmiModelDescription fmiVersion="2.0" modelName="Example" ...>
  <CoSimulation needsExecutionTool="true" .../>
  <ModelVariables>
    <ScalarVariable name="p" variability="continuous"
                    causality ="parameter">
      <Real start="0.0"/>
    </ScalarVariable>
    <ScalarVariable name="input" variability ="continuous"
                    causality ="input">
      <Real start="0.0"/>
    </ScalarVariable>
    <ScalarVariable name="val" variability ="continuous"
                    causality ="output" initial ="calculated">
      <Real/>
  </ModelVariables>
  <ModelStructure> ... </ModelStructure>
</fmiModelDescription>
```

SMOL and FMI

UNIVERSITY OF OSLO

Functional Mock-Up Objects (FMOs)

Tight integration of simulation units using FMI into programs.

```
//setup
Cont[out Double val] shadow =
    simulate("Sim.fmu", input=sys.val, p=1.0);
Cont[out Double val] sys = simulate("Realsys.fmu");
Monitor m = new Monitor(sys,shadow); m.run(1.0);
```

SMOL and FMI

Functional Mock-Up Objects (FMOs)

Tight integration of simulation units using FMI into programs.

```
//setup
Cont[out Double val] shadow =
    simulate("Sim.fmu", input=sys.val, p=1.0);
Cont[out Double val] sys = simulate("Realsys.fmu");
Monitor m = new Monitor(sys,shadow); m.run(1.0);
```

Integration

- Type of FMO directly checked against model description
- Variables become fields, functions become methods
- Causality reflected in type

SMOL and FMI

UNIVERSITY OF OSLO

Functional Mock-Up Interface (FMI)

Standard for (co-)simulation units, called function mock-up units (FMUs). Can also serve as interface to sensors and actuators.

SMOL and FMI

UNIVERSITY OF OSLO

Functional Mock-Up Interface (FMI)

Standard for (co-)simulation units, called function mock-up units (FMUs). Can also serve as interface to sensors and actuators.

```
//simplified shadow
class Monitor(Cont[out Double val] sys,
              Cont[out Double val] shadow)
  Unit run(Double threshold)
    while shadow != null do
      sys.doStep(1.0);  shadow.doStep(1.0);
      if(sys.val - shadow.val >= threshold) then ... end
    end ...
```

Is this twinning something? Is this setup correctly?

Constraints on Digital Twins

SMOL and FMI

UNIVERSITY OF OSLO

SMOL with FMOs

FMOs are objects, so they are part of the knowledge graph.

```
class Monitor(Cont[out Double val] sys,
                Cont[out Double val] shadow)
```

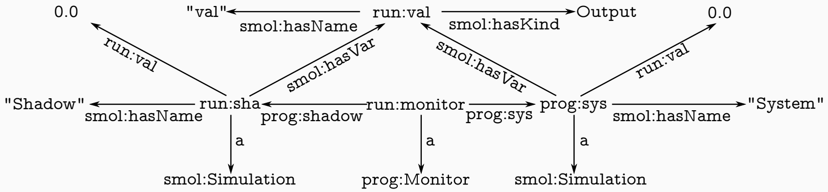
SMOL and FMI

UNIVERSITY OF OSLO

SMOL with FMOs

FMOs are objects, so they are part of the knowledge graph.

```
class Monitor(Cont[out Double val] sys,
              Cont[out Double val] shadow)
```



UNIVERSITY OF OSLO

SHACL

Define structural requirements as graph constraints in SHACL.

Example

Every monitor has a `shadow` FMU in its `shadow` field.

```
x:ShadowShape a sh:NodeShape;
              sh:targetClass prog:Monitor ;
sh:property [
   sh:path ( prog:shadow smol:hasName );
   sh:hasValue "Shadow" ; ].
```

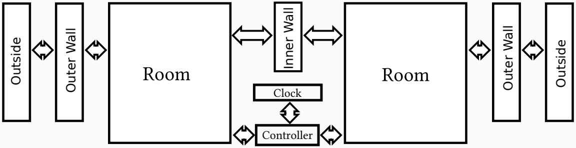
SHACL ignores reasoning, pure data constraints.

SPARQL

Define structural requirements as queries in SPARQL on *combined* knowledge graph, to use domain constraints on digital twins.

SPARQL

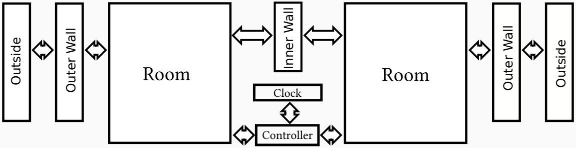
Define structural requirements as queries in SPARQL on *combined* knowledge graph, to use domain constraints on digital twin.



UNIVERSITY OF OSLO

SPARQL

Define structural requirements as queries in SPARQL on *combined* knowledge graph, to use domain constraints on digital twin.




```
class Room(Cont[...] f,
           Wall inner, Wall outer, Controller ctrl,
           Int id) end
class Controller(Cont[...] f,
                 Room left, Room right, Int id) end
class InnerWall(Cont[...] f, Room left, Room right) end
```

UNIVERSITY OF OSLO

SPARQL

Define structural requirements as queries in SPARQL on *combined* knowledge graph, to use domain constraints on digital twin.

Query to detect non-sensical setups:

```
SELECT ?room WHERE { ?ctrl a prog:Controller.
                    ?ctrl prog:left ?room.
                    ?ctrl prog:right ?room }
```

SPARQL

Define structural requirements as queries in SPARQL on *combined* knowledge graph, to use domain constraints on digital twin.

Query to check structural consistency for heaters:

```
SELECT * WHERE { ?o1 prog:id ?id1. ?h1 ast:id ?id1.
                 ?o2 prog:id ?id2. ?h2 ast:id ?id2.
                 ?h1 htLeftOf ?h2.
                 ?c a prog:Controller.
                 ?c prog:left ?o1. ?c prog:right ?o2.}
```

Outlook: Repairing your Twin

Semantic Reflection

One can use the knowledge graph *within* the program to detect structural drift: Formulate query to retrieve all mismatching parts

```
....
List<Repairs> repairs =
 construct("SELECT ?room ?wallLeft ?wallRight WHERE
   {?x ast:id ?room.
    ?x ast:right [ast:id ?wallRight].
    ?x ast:left [ast:id ?wallLeft].
    FILTER NOT EXISTS {?y a prog:Room; prog:id ?room.}}");
```

Repair function must restore structure.

Conclusion

Contributions

UNIVERSITY OF OSLO

Combining the Knowledge

- Export asset model of physical system as knowledge graph
- Export program state with simulators as knowledge graph
- Formulate constraints over combined knowledge

Possible Constraints

- Constraint on asset model
  "Is the asset model consistent?"
- Constraint on program
  "Is this a sensible simulation structure?"
- Constraints on twinning
  "Does the program have the same structure as the asset?"

Conclusion

Making sure your digital twin is twinning something.

Conclusion

Making sure your digital twin is twinning something.

Presented

- Semantic lifting of FMOs
- Using ontological information to formulate twinning

Conclusion

UNIVERSITY OF OSLO

Making sure your digital twin is twinning something.

Presented

- Semantic lifting of FMOs
- Using ontological information to formulate twinning

On-Going and Future Work

- Reconfiguring DT based on changes in asset model
- Adding ontological information to FMI model description

Conclusion

UNIVERSITY OF OSLO

Making sure your digital twin is twinning something.

Presented

- Semantic lifting of FMOs
- Using ontological information to formulate twinning

On-Going and Future Work

- Reconfiguring DT based on changes in asset model
- Adding ontological information to FMI model description

Thank you for your attention