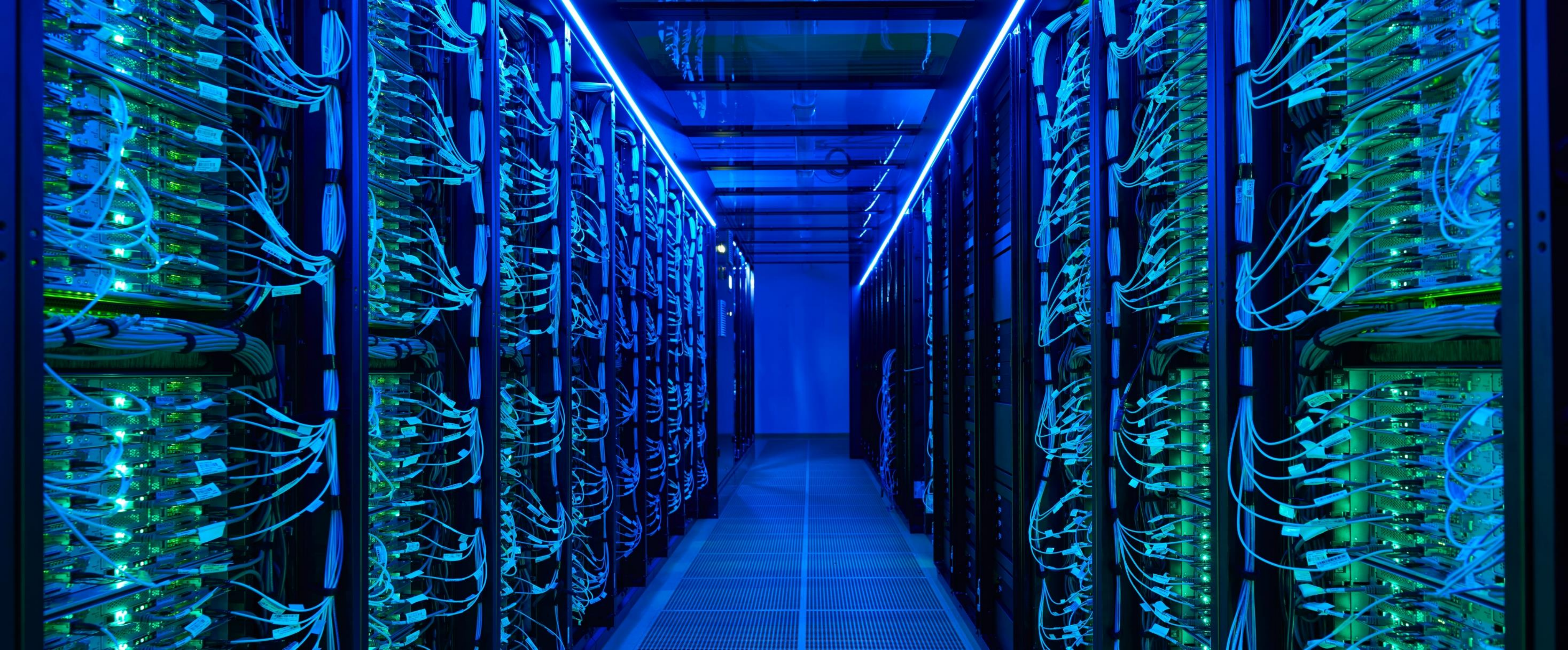
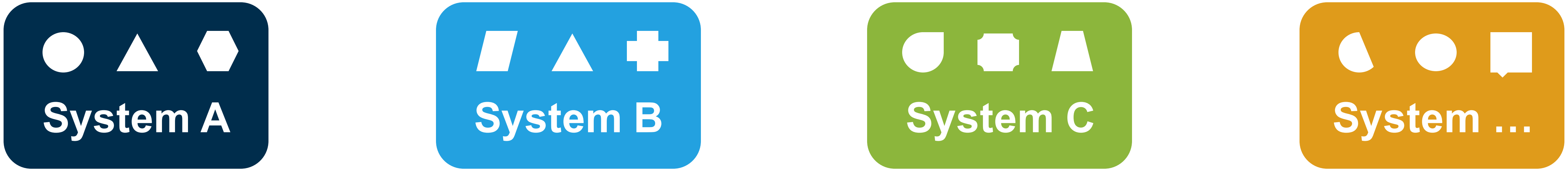


Towards Machine-actionable FAIR Digital Objects with a Typing Model that Enables Operations

Maximilian Inckmann, Nicolas Blumenröhr, Rossella Aversa
Scientific Computing Center – Karlsruhe Institute of Technology

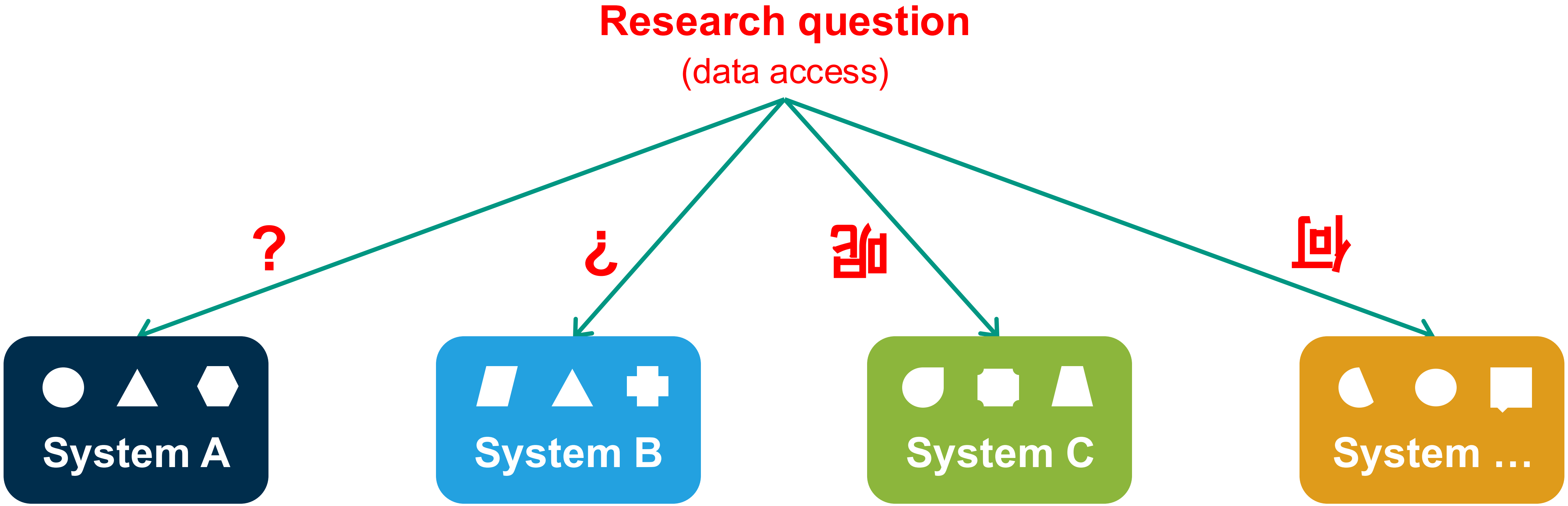
How research data looks today



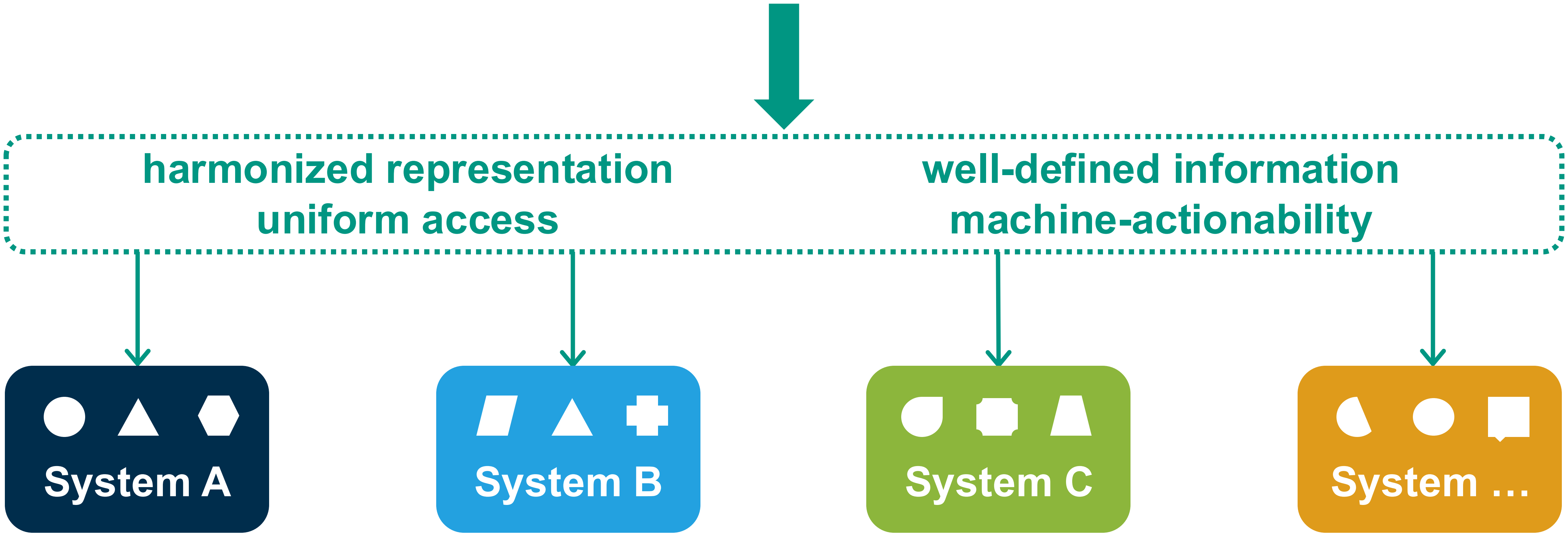
Many systems

A lot of (meta)data formats

How research data looks today



What is a FAIR Digital Object?

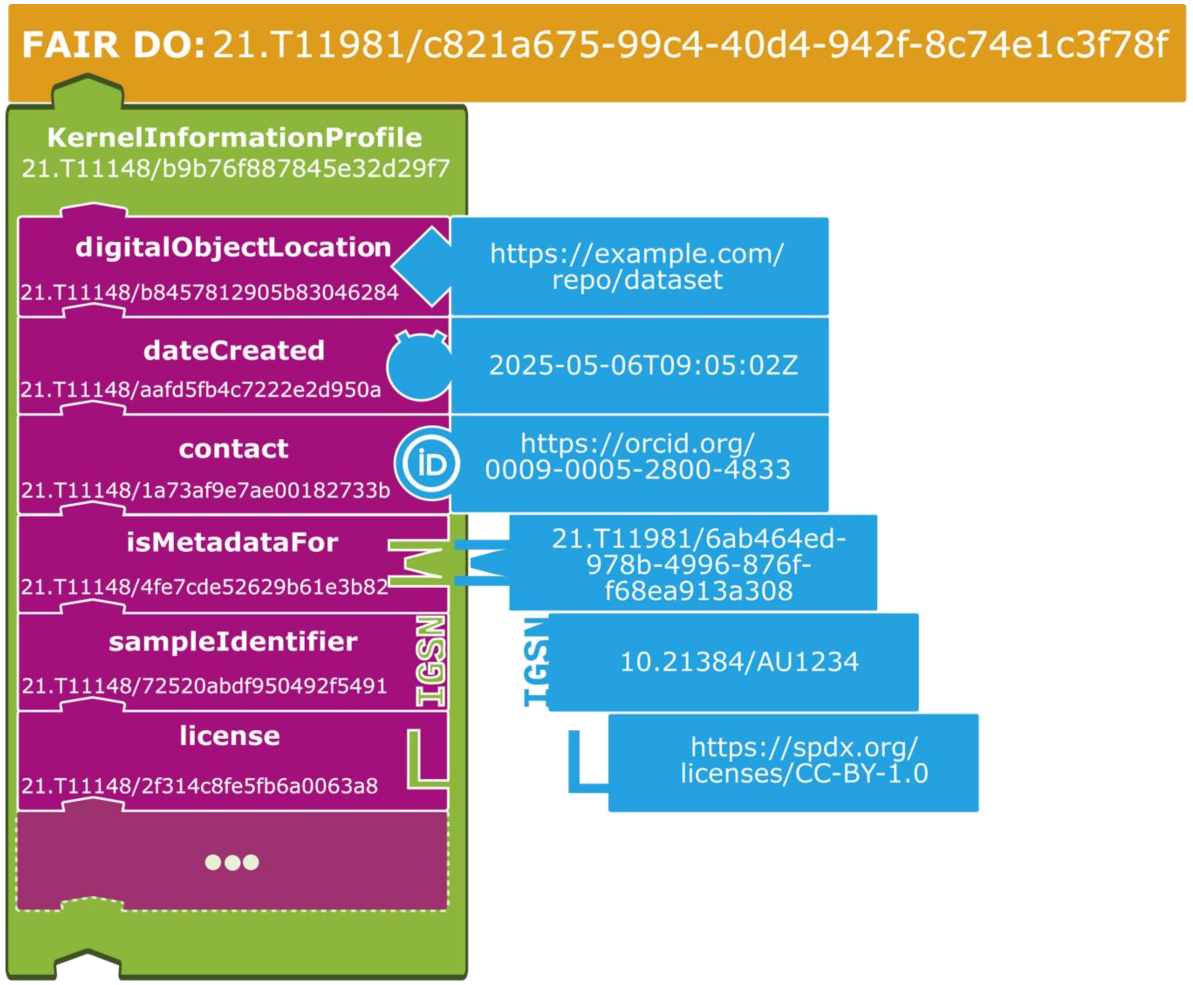


What is a FAIR Digital Object?

- Persistent identification & storage by using Handle PIDs and Handle records
- Values are typed and validated
- No fixed content schema (aka. profiles)
- Harmonization achieved by reusing existing data types and profiles
- Refers to (meta)data in existing systems

FAIR-DOs are machine-interpretable!

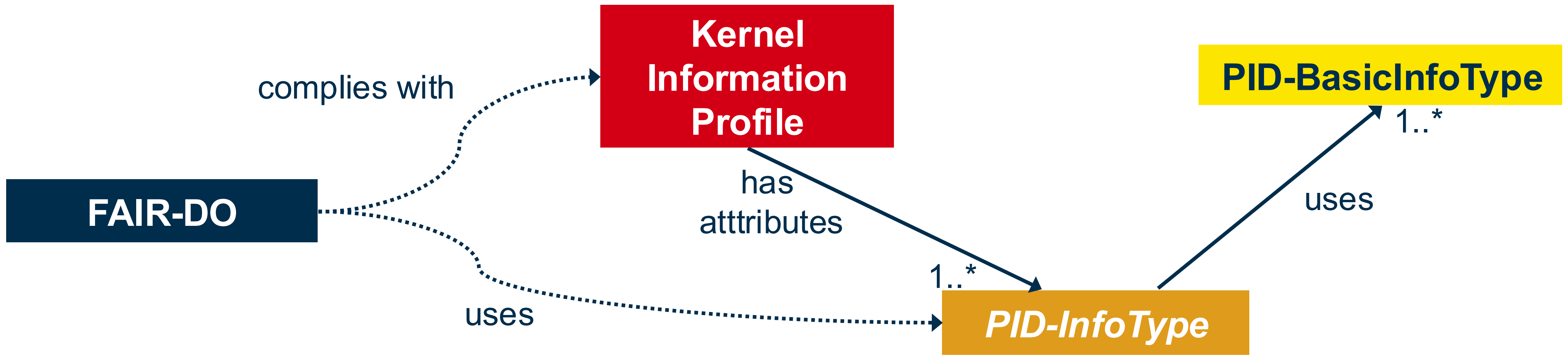
Next step: Use this for highly automated data processing based on a type system


FAIR DO: 21.T11981/c821a675-99c4-40d4-942f-8c74e1c3f78f
KernelInformationProfile
21.T11148/b9b76f887845e32d29f7
digitalObjectLocation
21.T11148/b8457812905b83046284
https://example.com/repo/dataset
dateCreated
21.T11148/aafd5fb4c7222e2d950a
2025-05-06T09:05:02Z
contact
21.T11148/1a73af9e7ae00182733b
https://orcid.org/0009-0005-2800-4833
isMetadataFor
21.T11148/4fe7cde52629b61e3b82
21.T11981/6ab464ed-978b-4996-876f-f68ea913a308
sampleIdentifier
21.T11148/72520abdf950492f5491
IGSN
10.21384/AU1234
license
21.T11148/2f314c8fe5fb6a0063a8
https://spdx.org/licenses/CC-BY-1.0
•••

FAIR-DO Type System

Currently: Cordra Data Type Registries

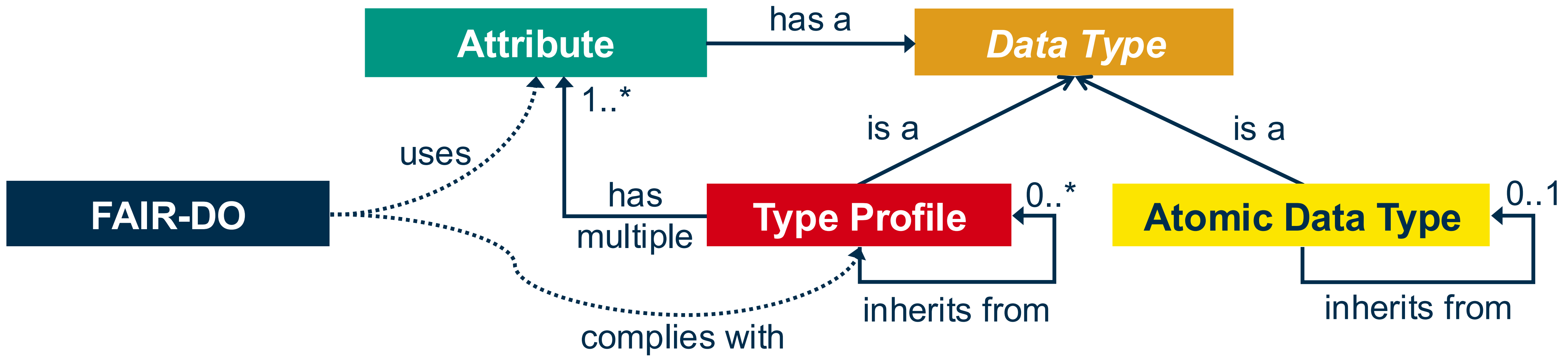
- PID-BasicInfoTypes, PID-InfoTypes, Kernel Information Profiles
- JSON schema capabilities → only syntactic validation
- No inheritance/reuse possible
- 3 instances – 3 slightly different schemas



FAIR-DO Type System

Our approach

- Data Type: abstract superclass
- Atomic Data Type: syntax of a value
- Type Profile: combination of other data types
- Attributes: semantics and cardinality
- Includes inheritance relationships



Goal: How can we achieve machine-actionability?

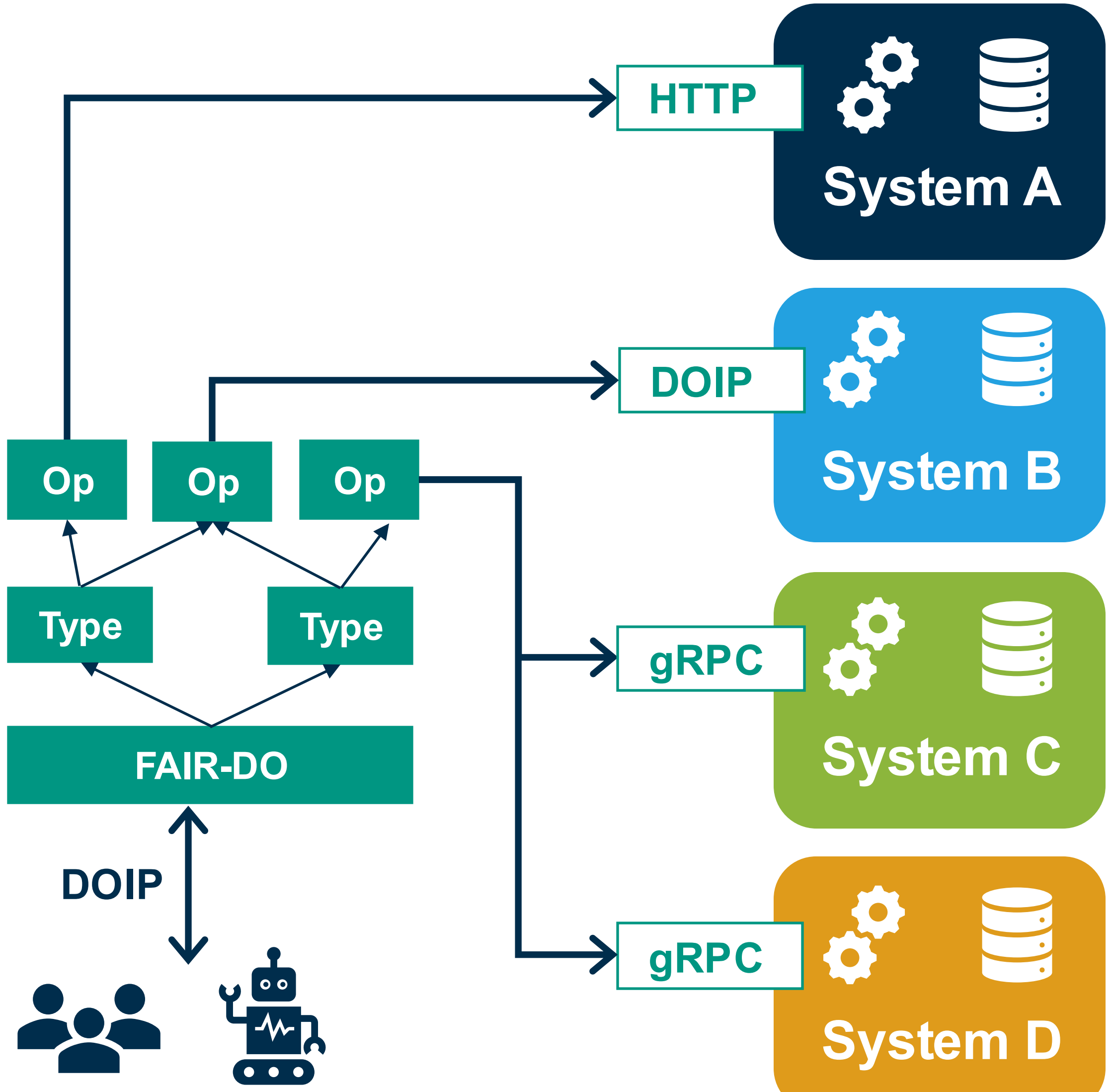
Our approach: type-associated Operations

- Operations may use arbitrary existing technologies (e.g., languages, protocols, …)
- Operations may be executed in various environments (e.g., Edge, Cloud, HPC)
- Automatic discovery of available Operations for a given FAIR-DO

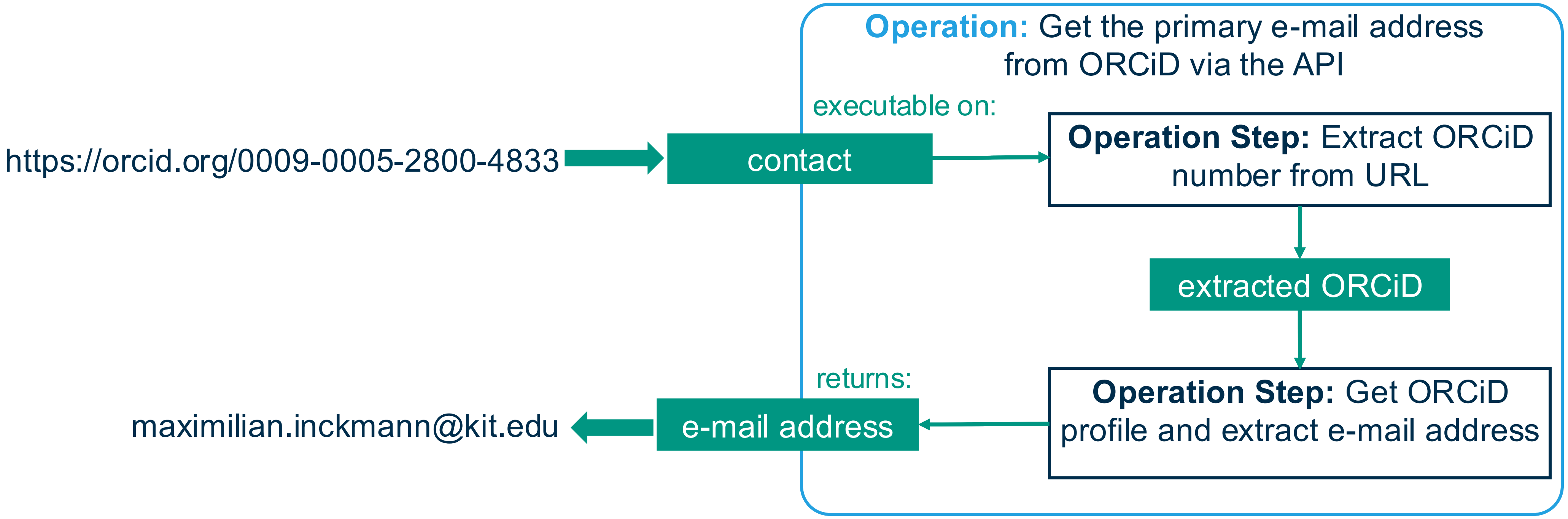
→ Associate data types with Operations

→ Describe technologies and how they are executed

→ Reuse technologies in multiple Operations

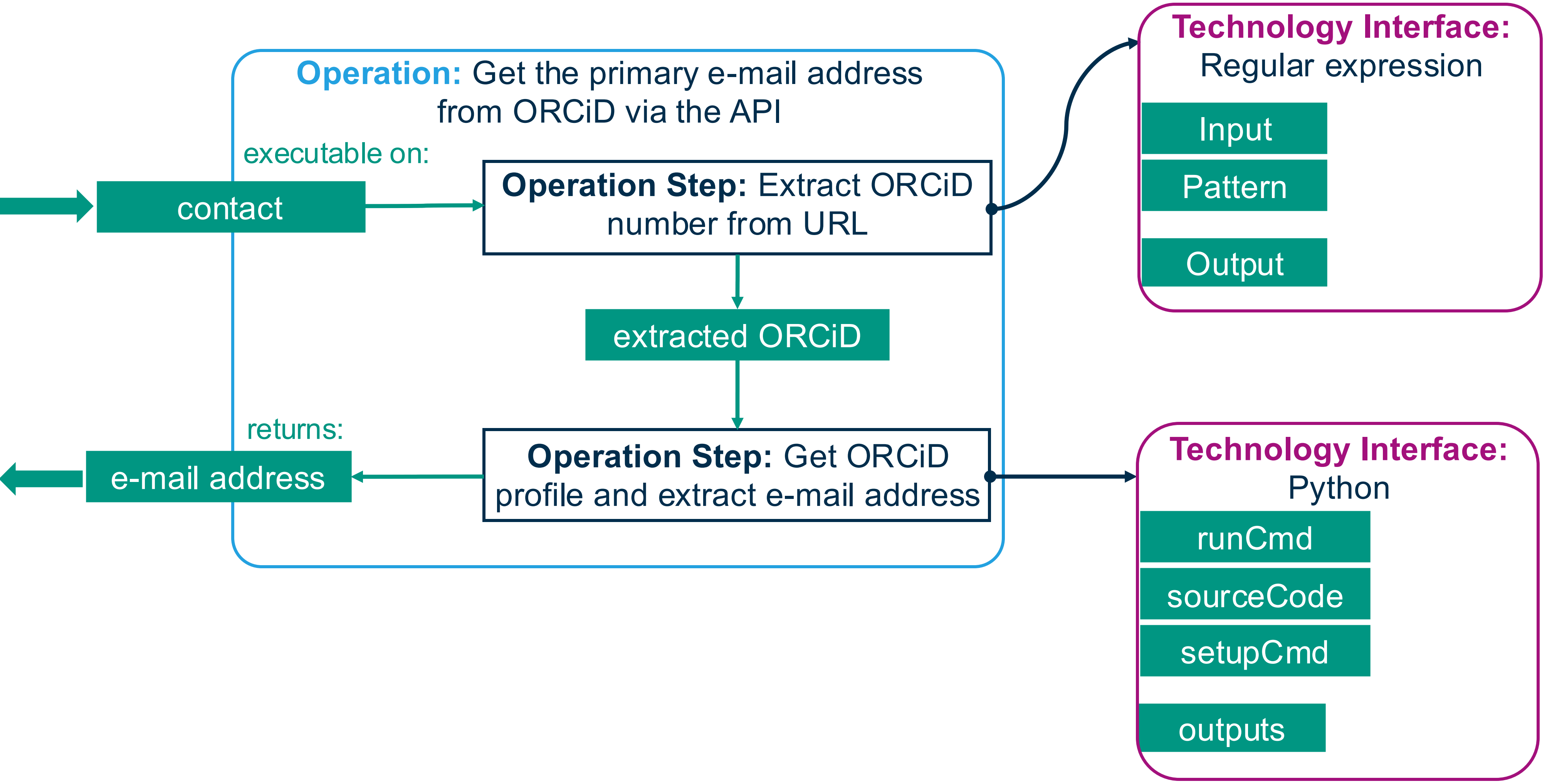


Example for a type-associated FAIR-DO Operation


Operation: Get the primary e-mail address from ORCiD via the API
executable on:
https://orcid.org/0009-0005-2800-4833
contact
Operation Step: Extract ORCiD number from URL
extracted ORCiD
Operation Step: Get ORCiD profile and extract e-mail address
returns:
e-mail address
maximilian.inckmann@kit.edu

Attribute Operation
Attribute Mapping Technology Interface

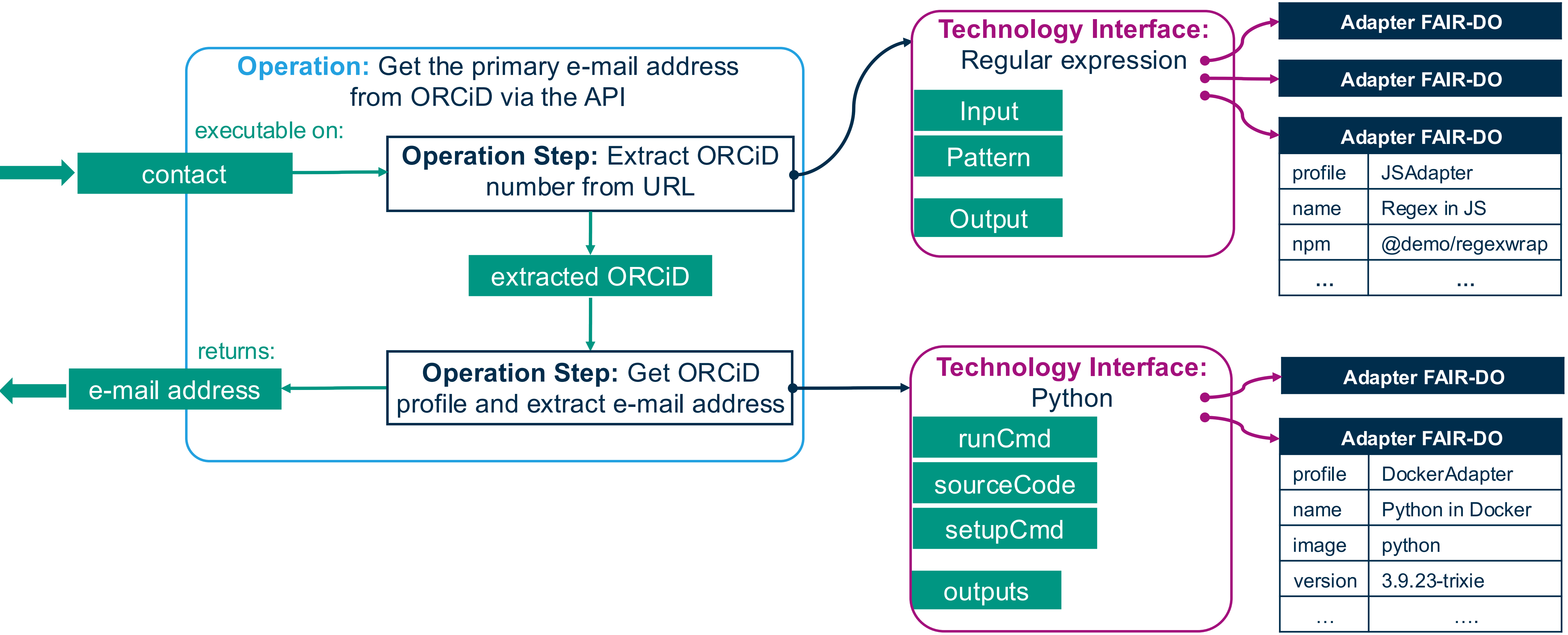
Example for a type-associated FAIR-DO Operation



Operation: Get the primary e-mail address from ORCiD via the API
executable on:
contact
Operation Step: Extract ORCiD number from URL
extracted ORCiD
returns:
e-mail address
Operation Step: Get ORCiD profile and extract e-mail address
Technology Interface: Regular expression
Input
Pattern
Output
Technology Interface: Python
runCmd
sourceCode
setupCmd
outputs

Attribute
Operation
Attribute Mapping
Technology Interface

Example for a type-associated FAIR-DO Operation

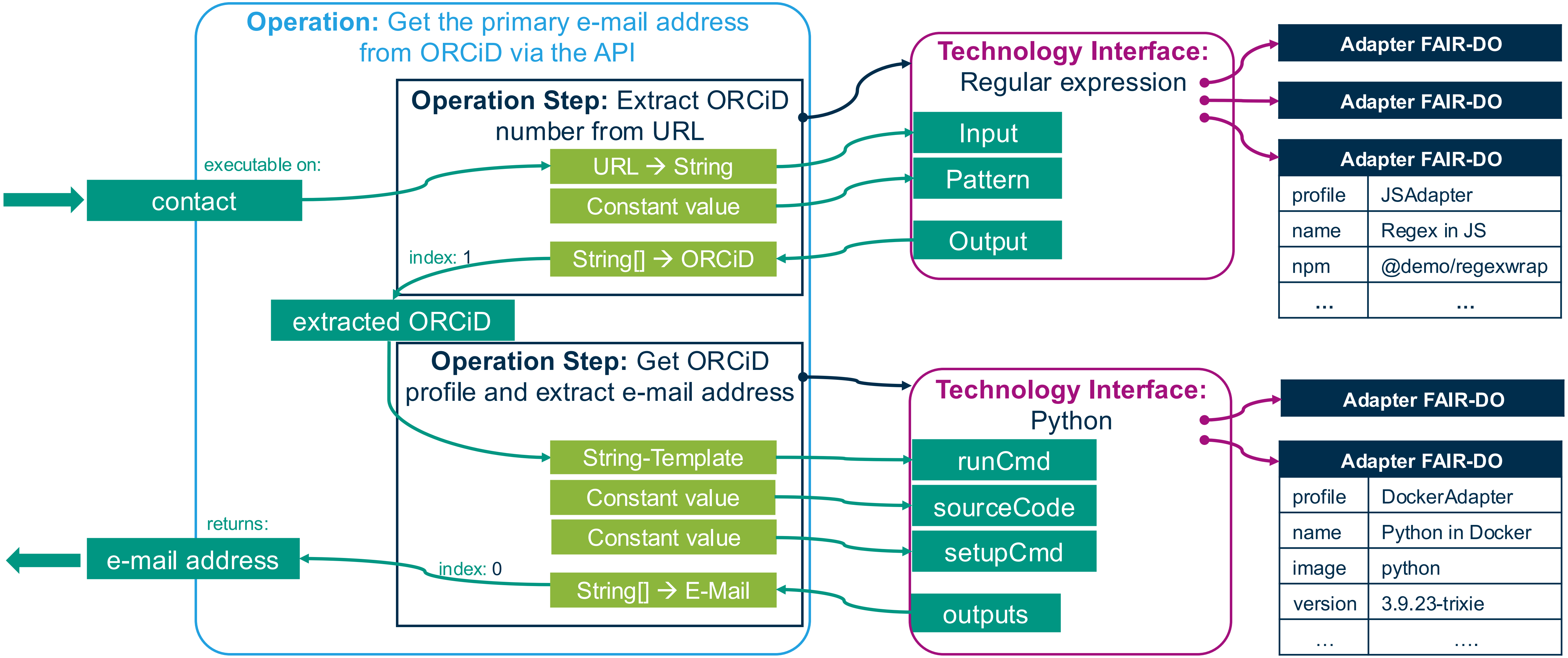

Operation: Get the primary e-mail address from ORCiD via the API
executable on:
contact
Operation Step: Extract ORCiD number from URL
extracted ORCiD
returns:
e-mail address
Operation Step: Get ORCiD profile and extract e-mail address
Technology Interface: Regular expression
Input
Pattern
Output
Technology Interface: Python
runCmd
sourceCode
setupCmd
outputs
Adapter FAIR-DO
Adapter FAIR-DO
Adapter FAIR-DO
Adapter FAIR-DO
Adapter FAIR-DO

profile	JSAdapter
name	Regex in JS
npm	@demo/regexwrap
…	…

profile	DockerAdapter
name	Python in Docker
image	python
version	3.9.23-trixie
…	….

Attribute
Attribute Mapping
Operation
Technology Interface

Example for a type-associated FAIR-DO Operation

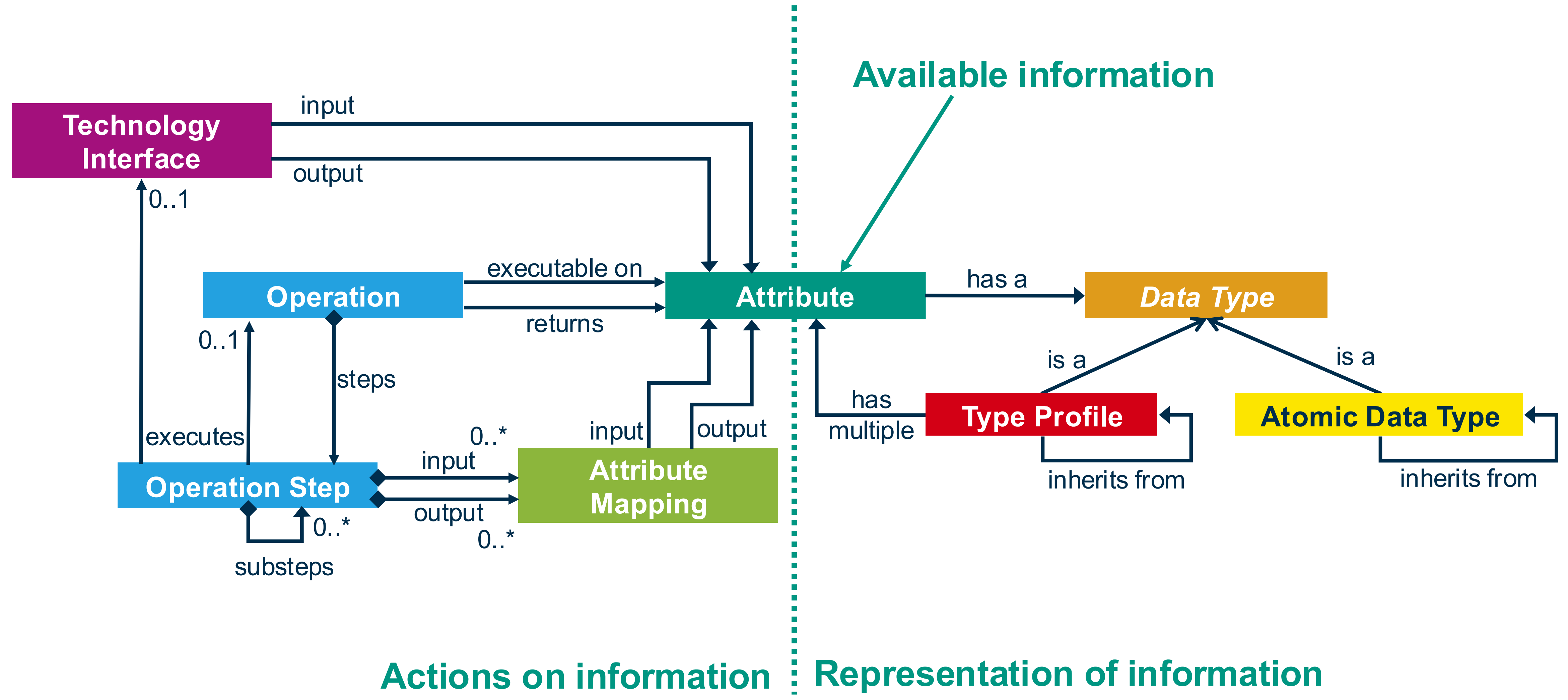


Adapter FAIR-DO	
profile	JSAdapter
name	Regex in JS
npm	@demo/regexwrap
…	…

Adapter FAIR-DO	
profile	DockerAdapter
name	Python in Docker
image	python
version	3.9.23-trixie
…	….

Attribute **Operation**
Attribute Mapping **Technology Interface**

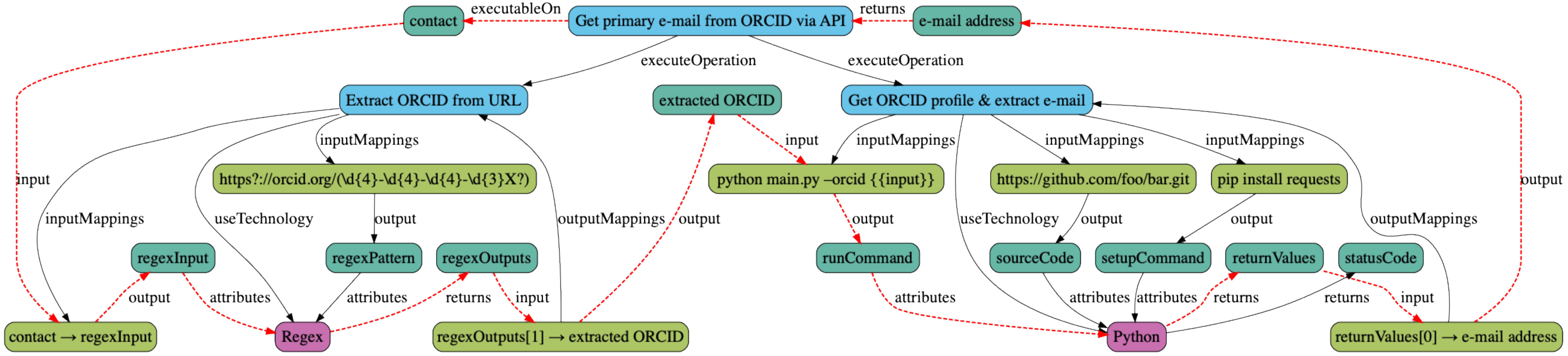
Our FAIR-DO Type System that Enables Operations



Prototypical Implementation

Integrated Data Type and Operation Registry with Inheritance System (IDORIS)

- Technologies used: Java, Spring, Neo4j
- Realizes inheritance mechanisms
- Automatically determines association between operations and attributes/data types
- Rule-based logic for validating entities not only syntactically, but also in their context
  → acyclicity; conflict-free inheritance hierarchy



Conclusions and Future Work

- Type-associated FAIR-DO Operations abstract the complexity of finding and executing Operations from users
  - Agnostic to the concrete environment they are executed in (e.g., Docker, bare-metal)
  - Able to describe and use various technologies (languages, libraries, APIs, etc.)
  - Enable reuse of technologies across multiple operations
  - Association mechanism between Operations and Data Types
  - Inheritance mechanisms that enable reuse of Data Types
- IDORIS realizes robust validation that ensures correct syntax, acyclicity, and other rules

→ Fundamental typing infrastructure for machine-actionable FAIR-DOs and reproducible Operations

Next step: Execute technology-agnostic FAIR-DO Operations

Maximilian Inckmann – maximilian.inckmann@kit.edu