1 St Petersburg University

2 Perm Sate University

3 UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Semantic Hashing to Remedy Uncertainties in Ontology-Driven Edge Computing

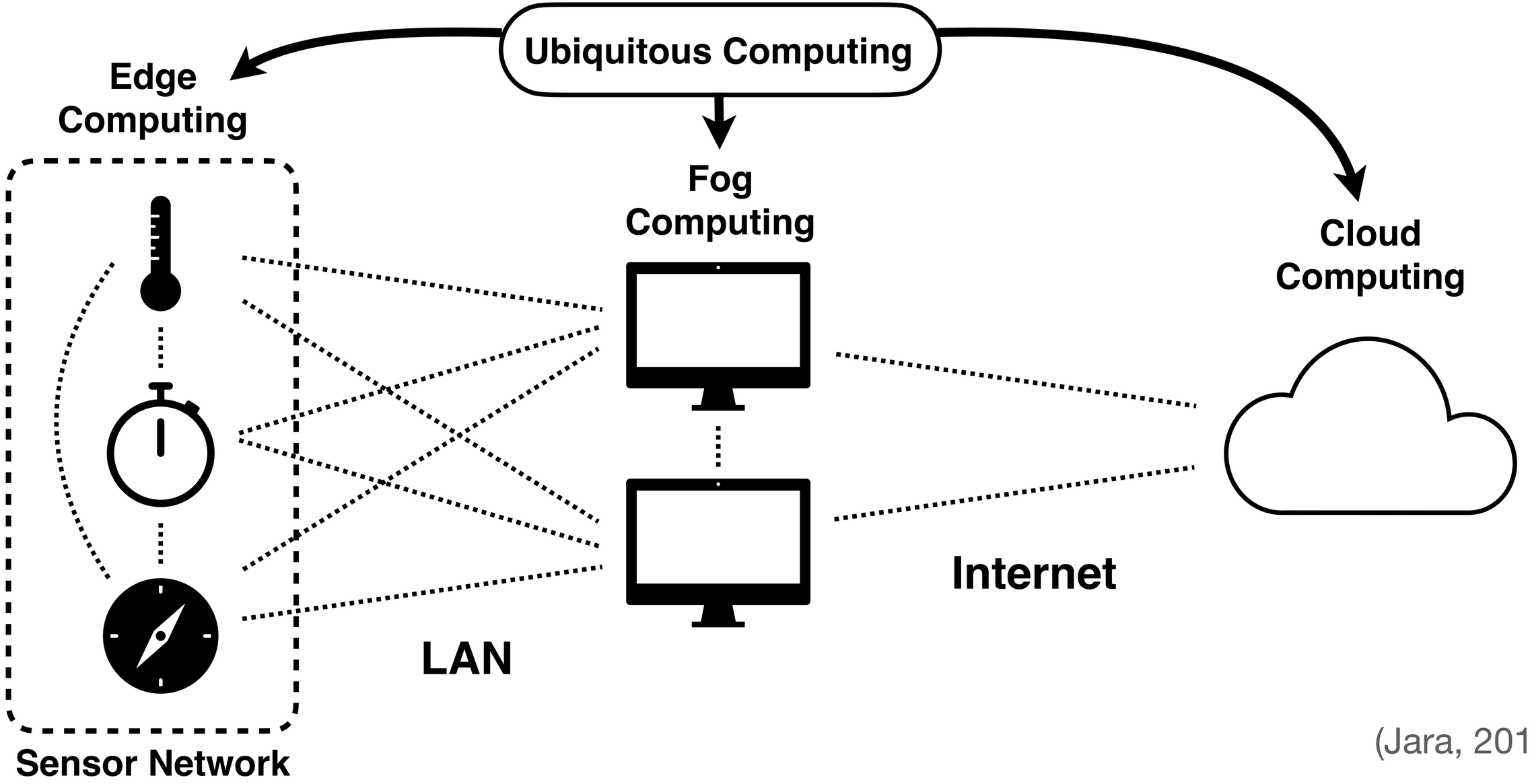
This study is supported by the research grant No. ID92566385 from Saint Petersburg University, "Text processing in L1 and L2: Experimental study with eye-tracking, visual analytics and virtual reality technologies"

Konstantin Ryabinin [1,2,3], kostya.ryabinin@gmail.com

Svetlana Chuprina [2], chuprinas@inbox.ru

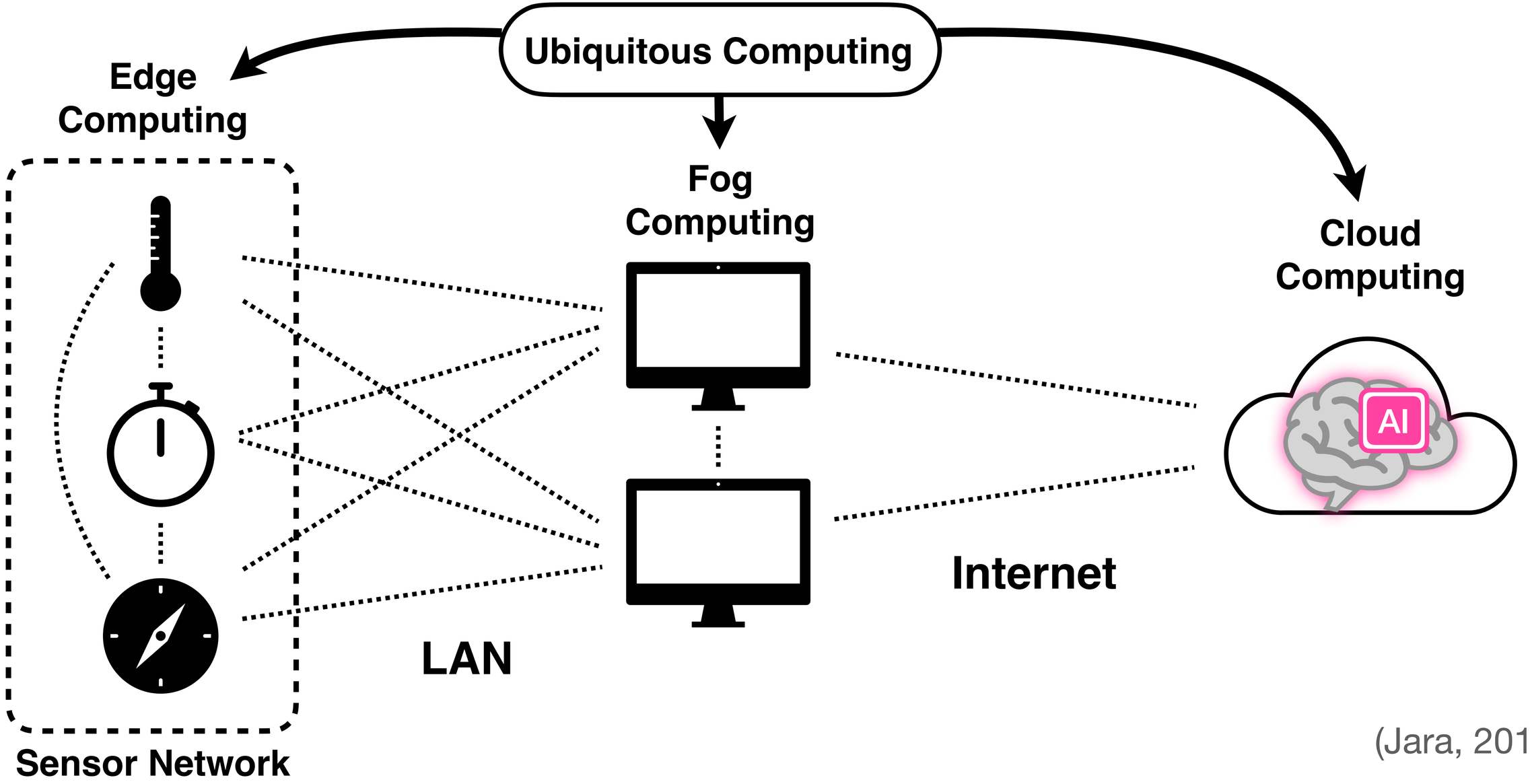
Prague – 2023

Edge Intelligence



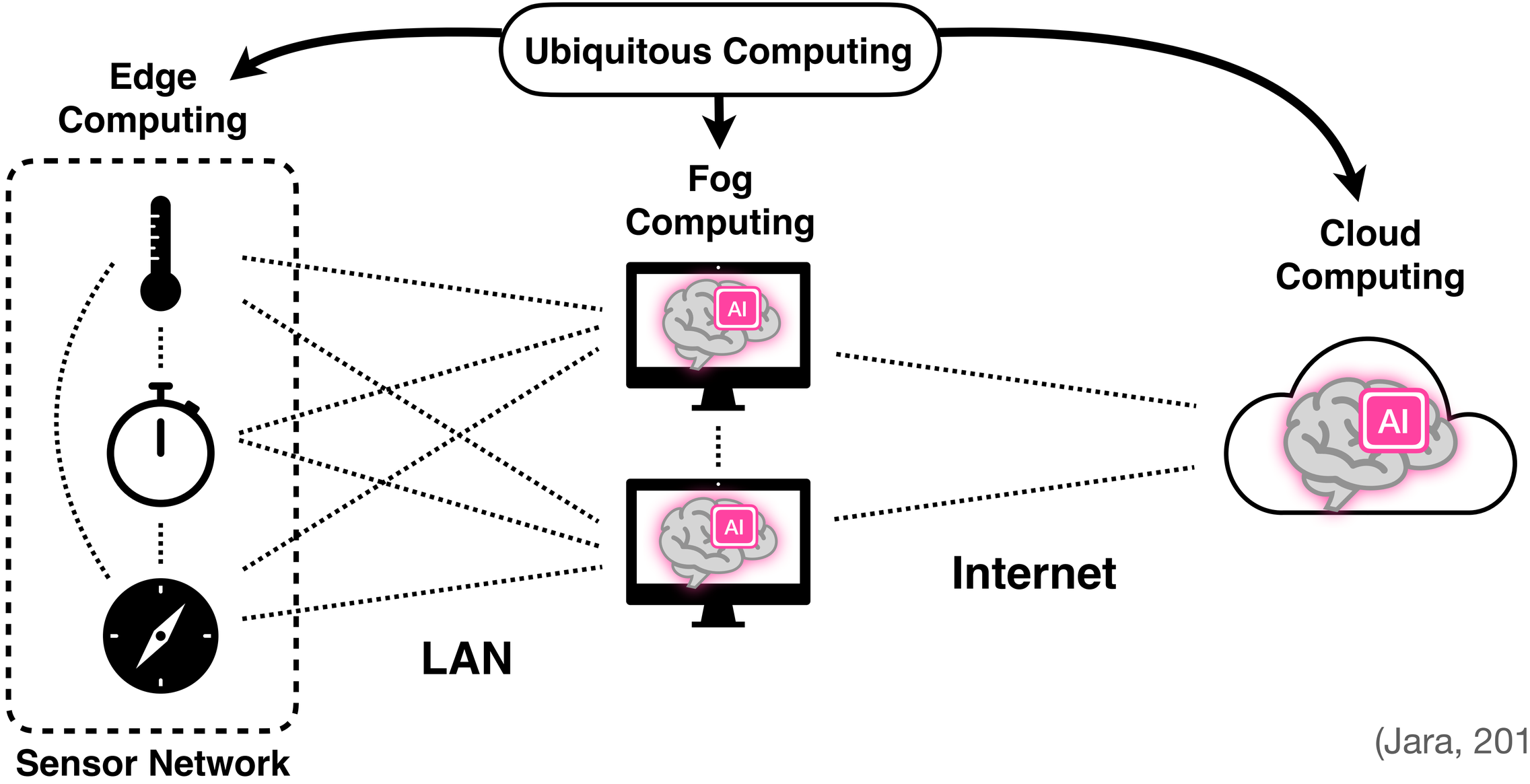
(Jara, 2014)

Edge Intelligence



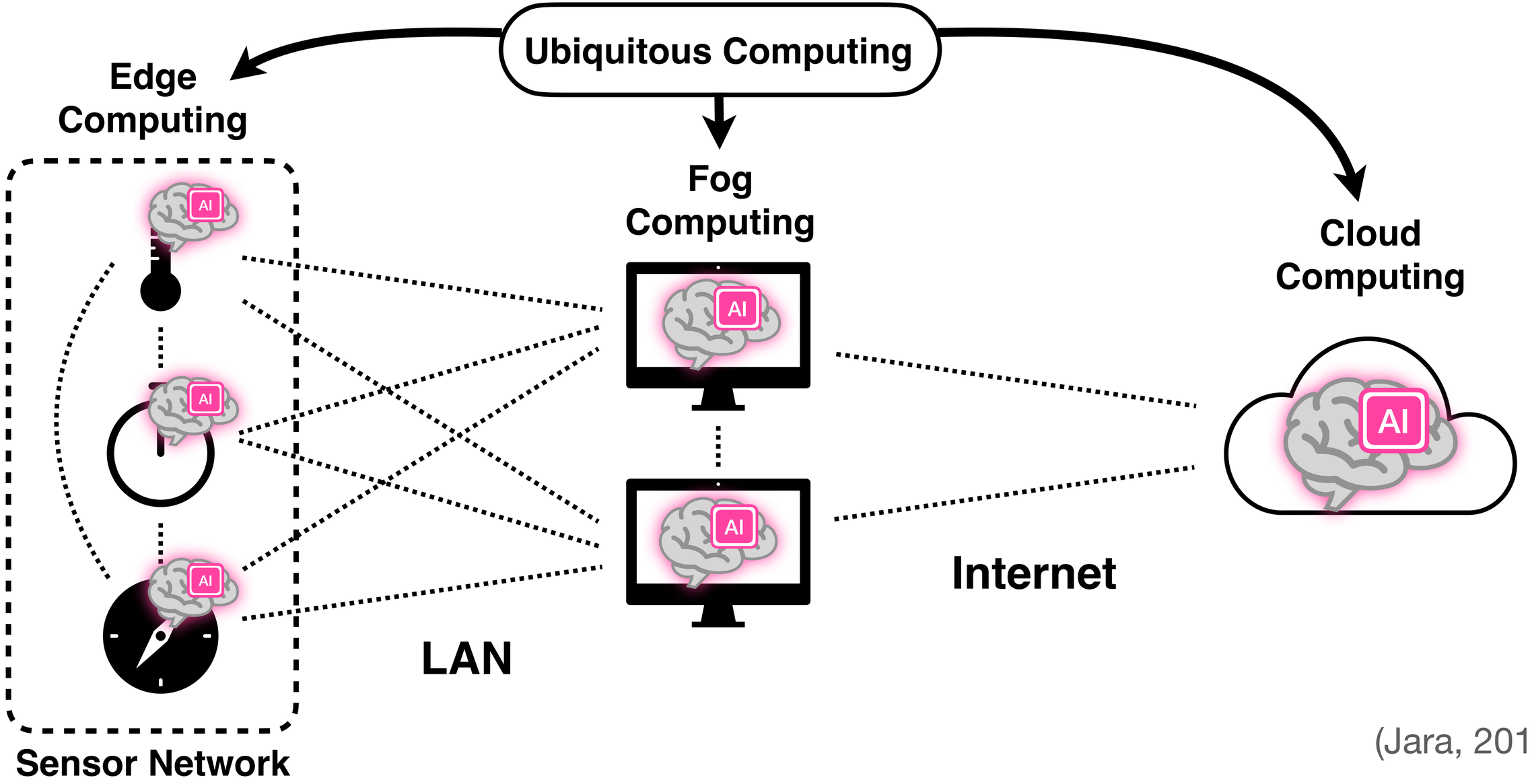
(Jara, 2014)

Edge Intelligence



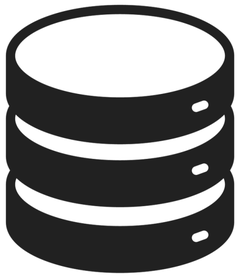
(Jara, 2014)

Edge Intelligence

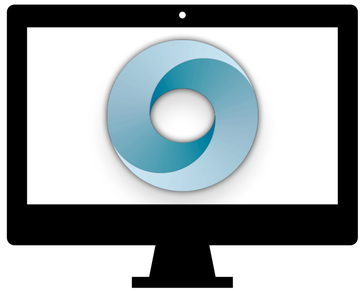


(Jara, 2014)

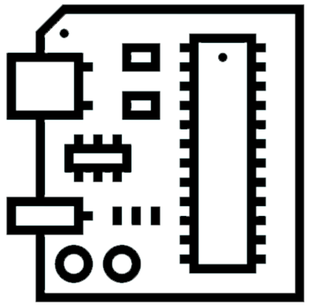
Ontology-Driven Edge Computing (ODEC)



Application Ontology of Operators



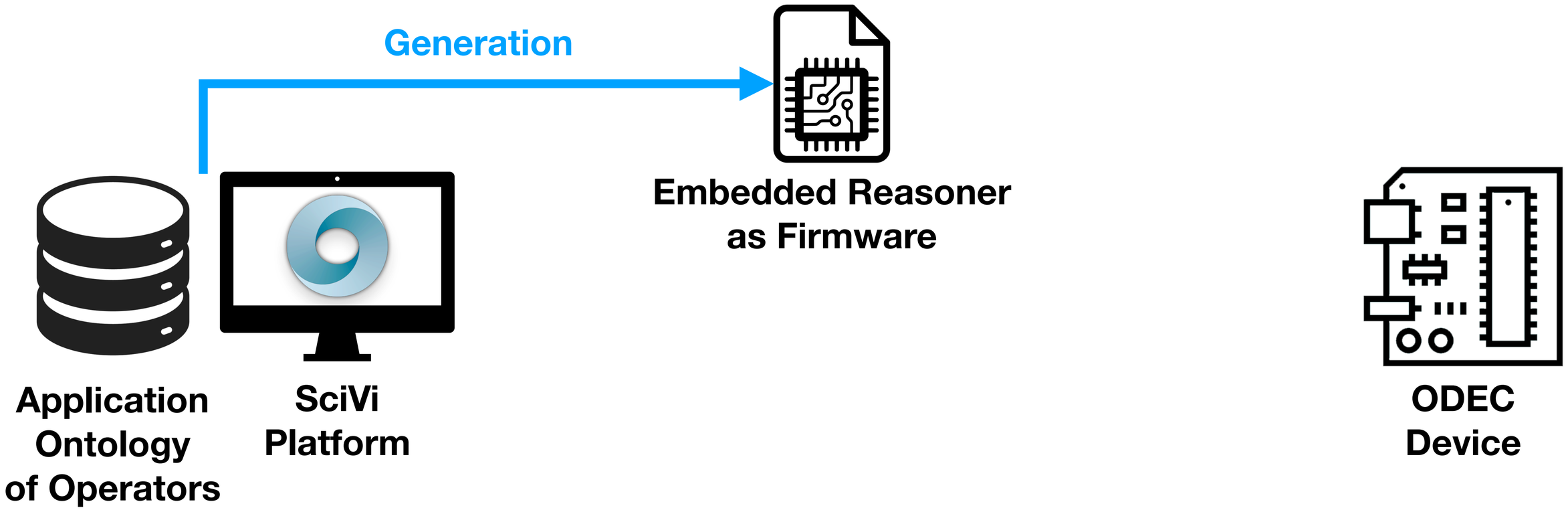
SciVi Platform



ODEC Device

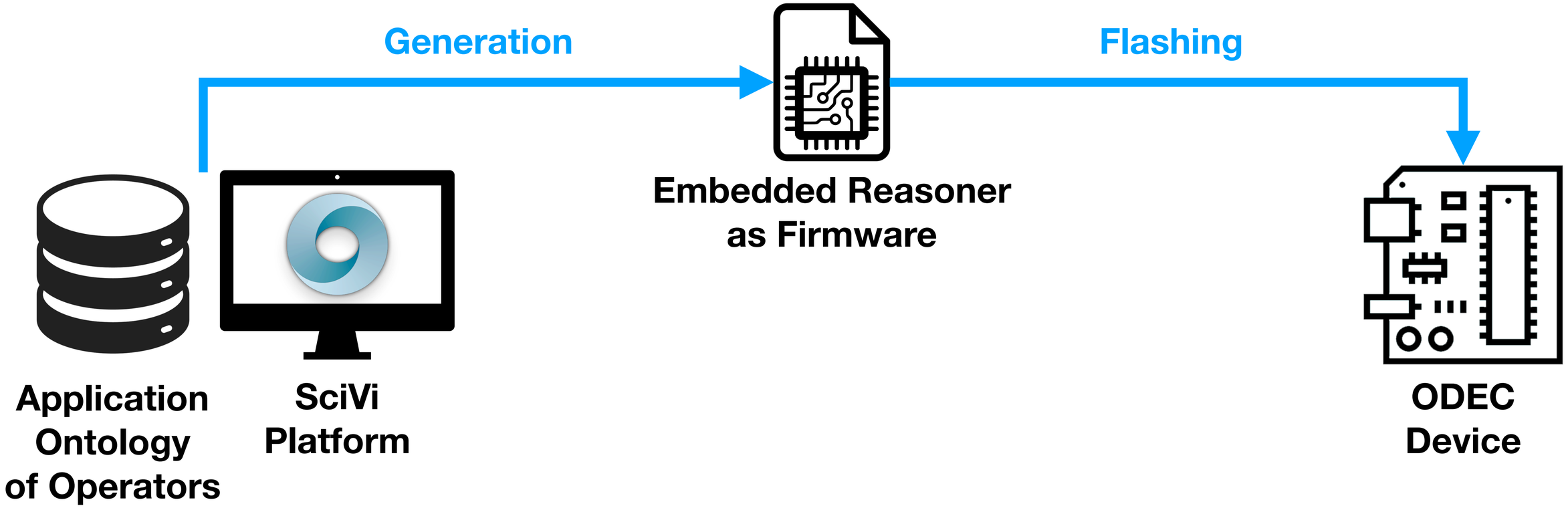
(Ryabinin, 2020)

Ontology-Driven Edge Computing (ODEC)



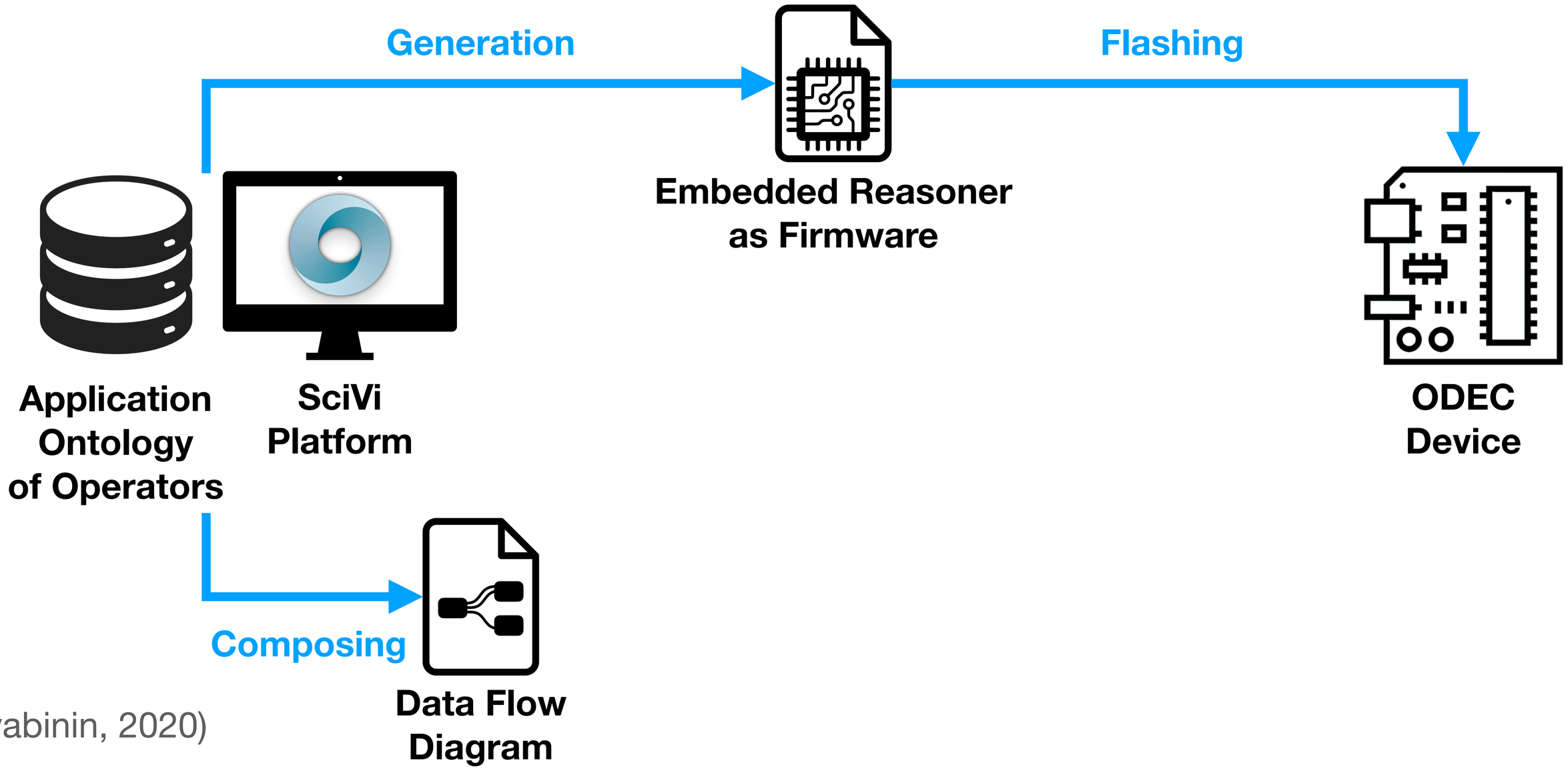
(Ryabinin, 2020)

Ontology-Driven Edge Computing (ODEC)



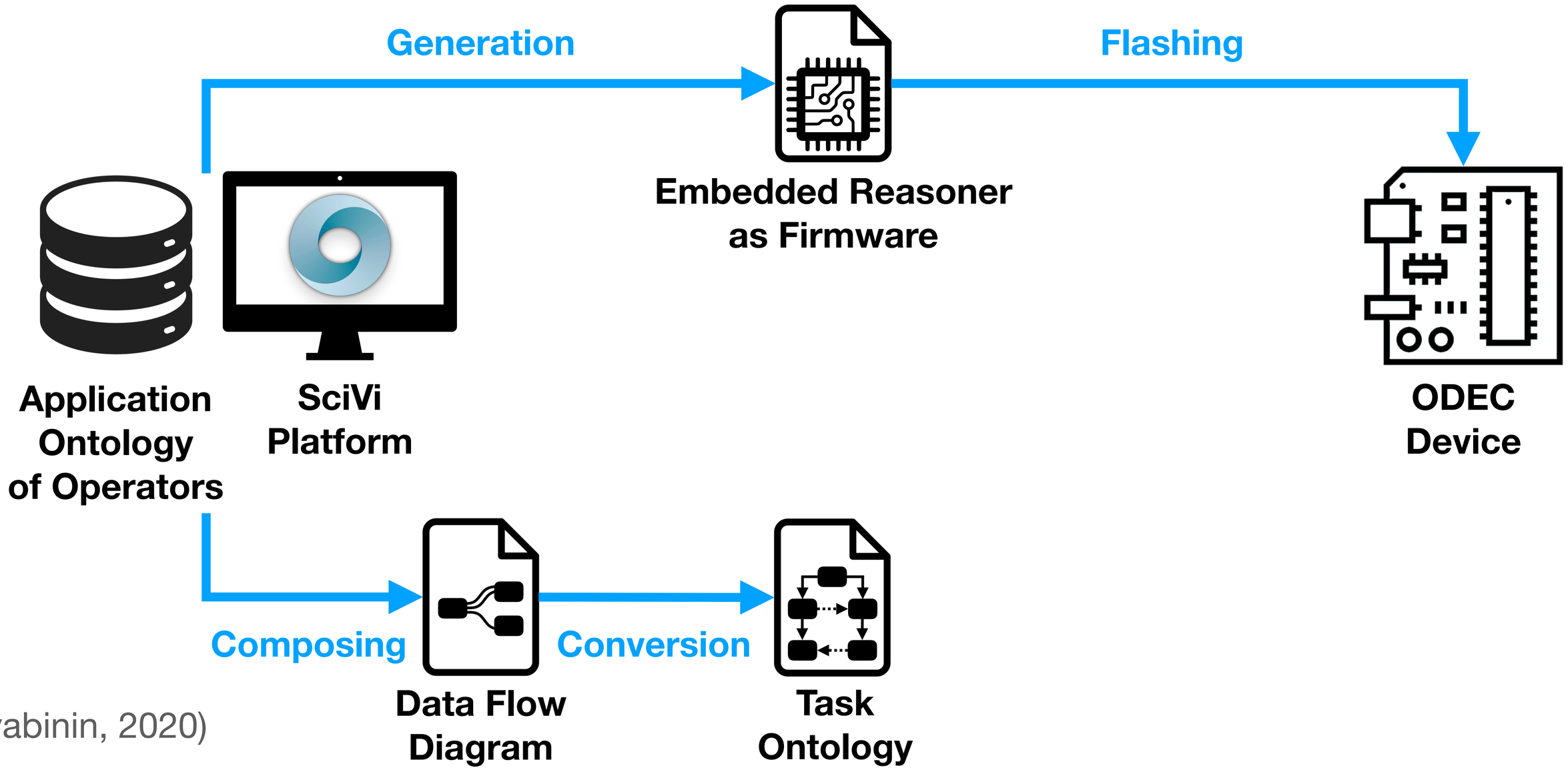
(Ryabinin, 2020)

Ontology-Driven Edge Computing (ODEC)



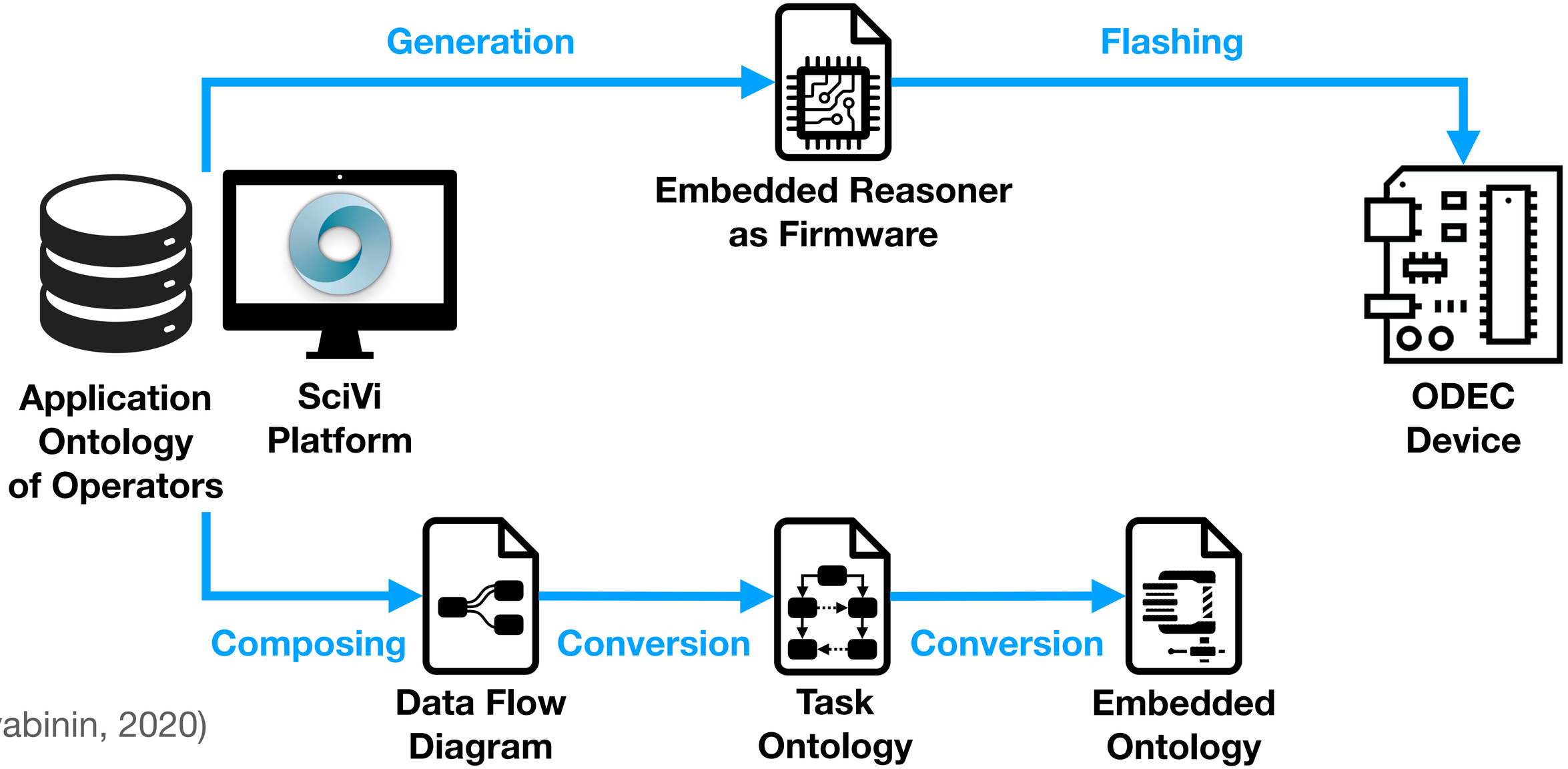
(Ryabinin, 2020)

Ontology-Driven Edge Computing (ODEC)



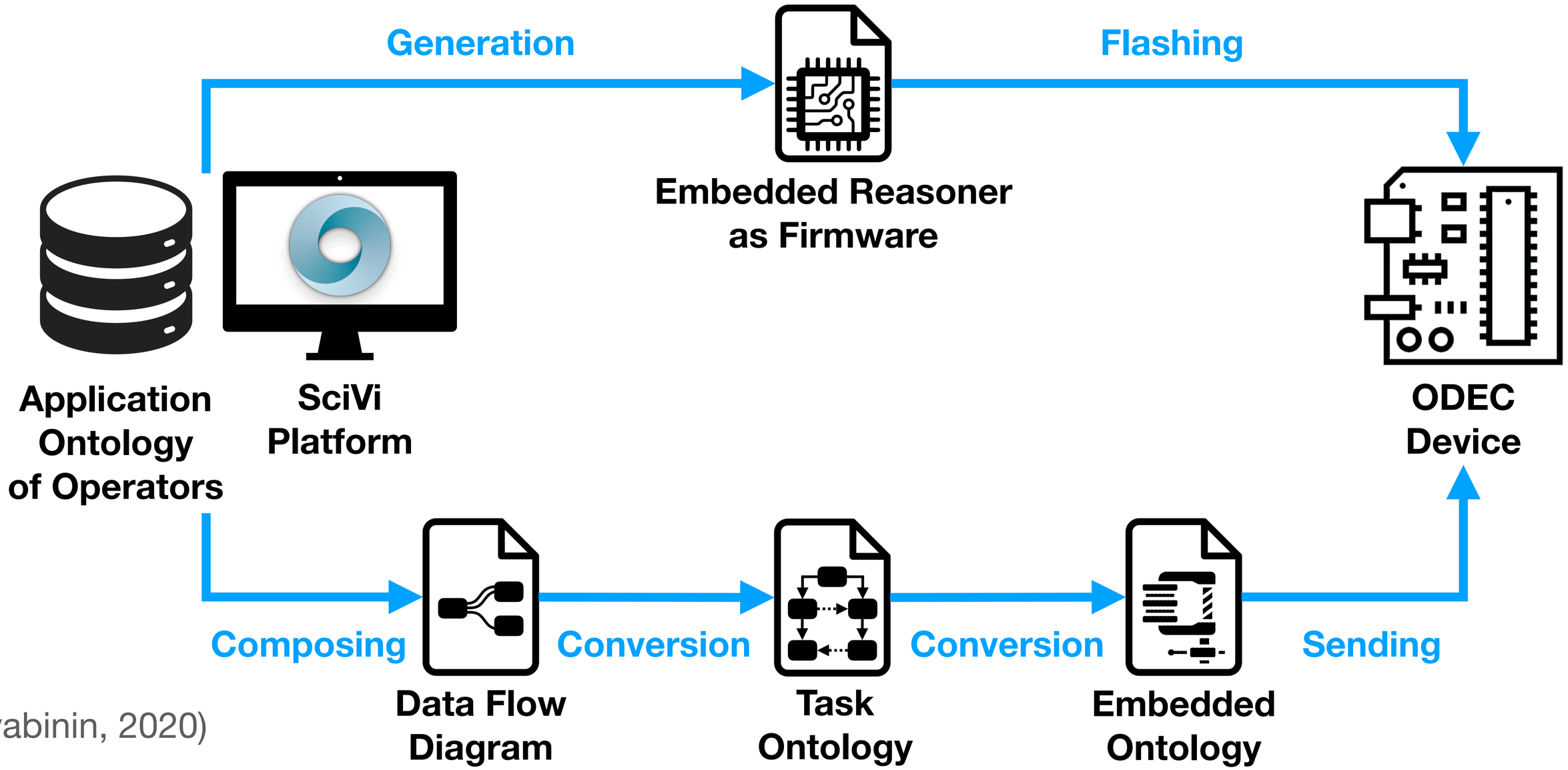
(Ryabinin, 2020)

Ontology-Driven Edge Computing (ODEC)



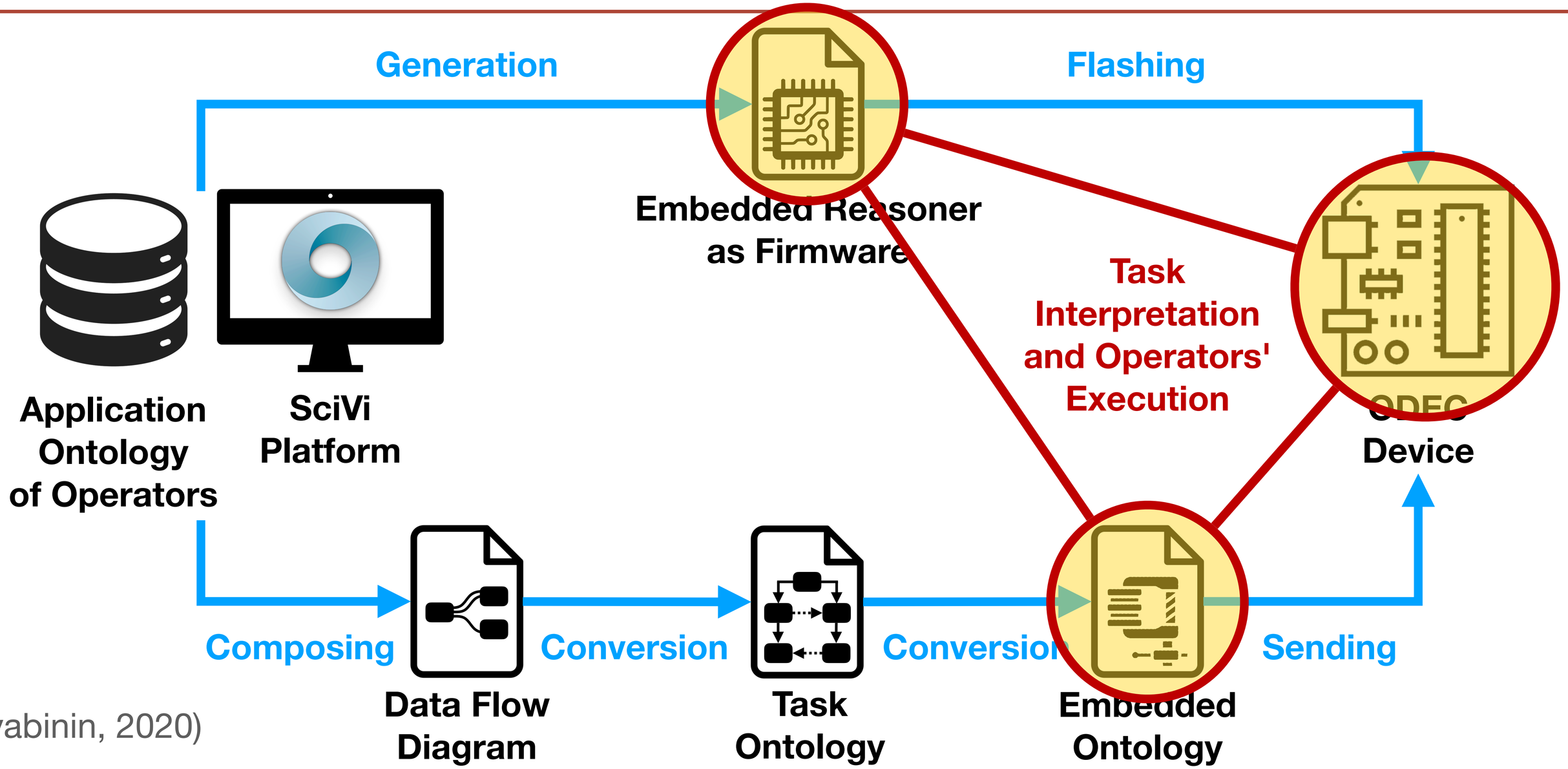
(Ryabinin, 2020)

Ontology-Driven Edge Computing (ODEC)



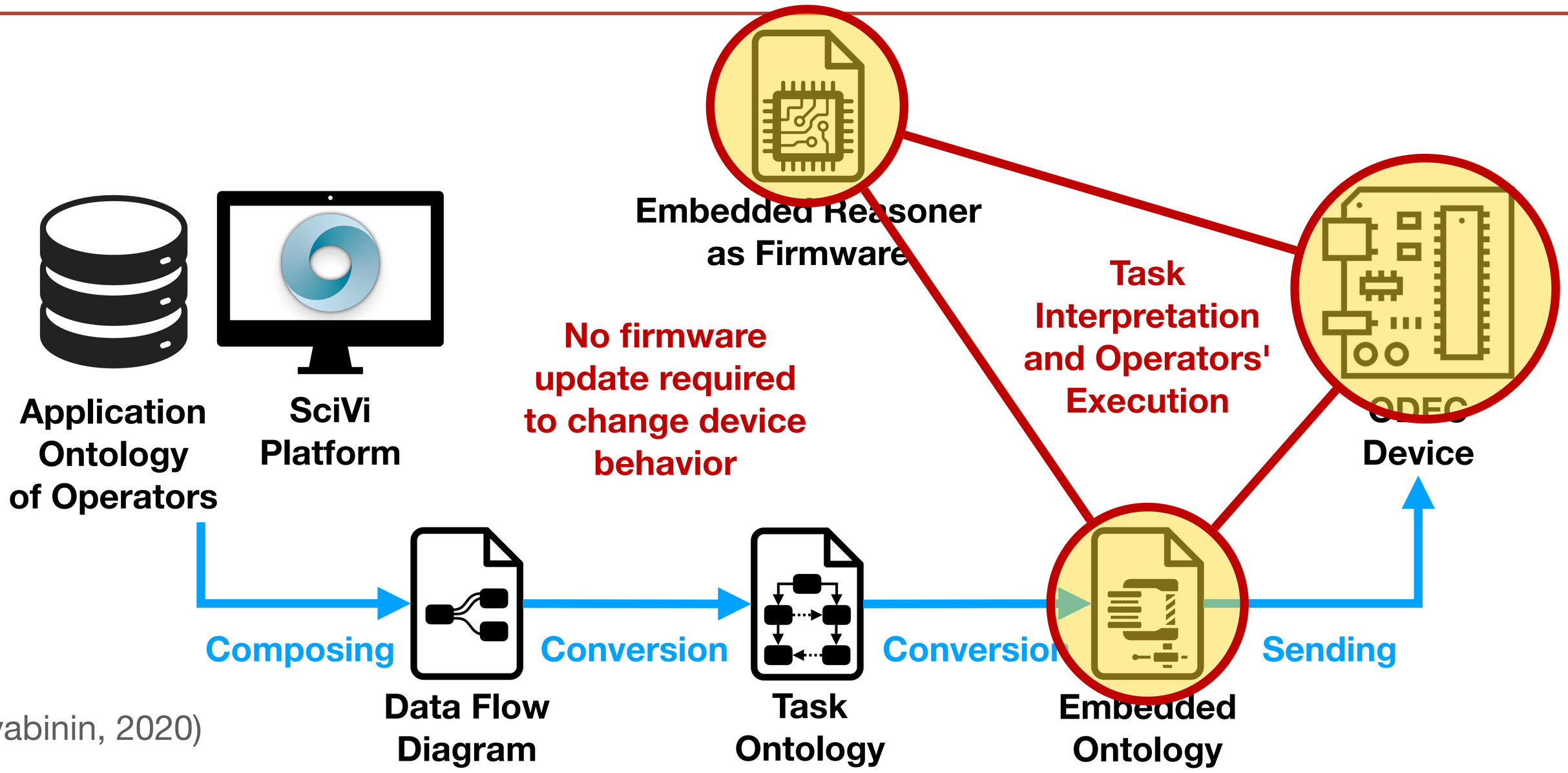
(Ryabinin, 2020)

Ontology-Driven Edge Computing (ODEC)



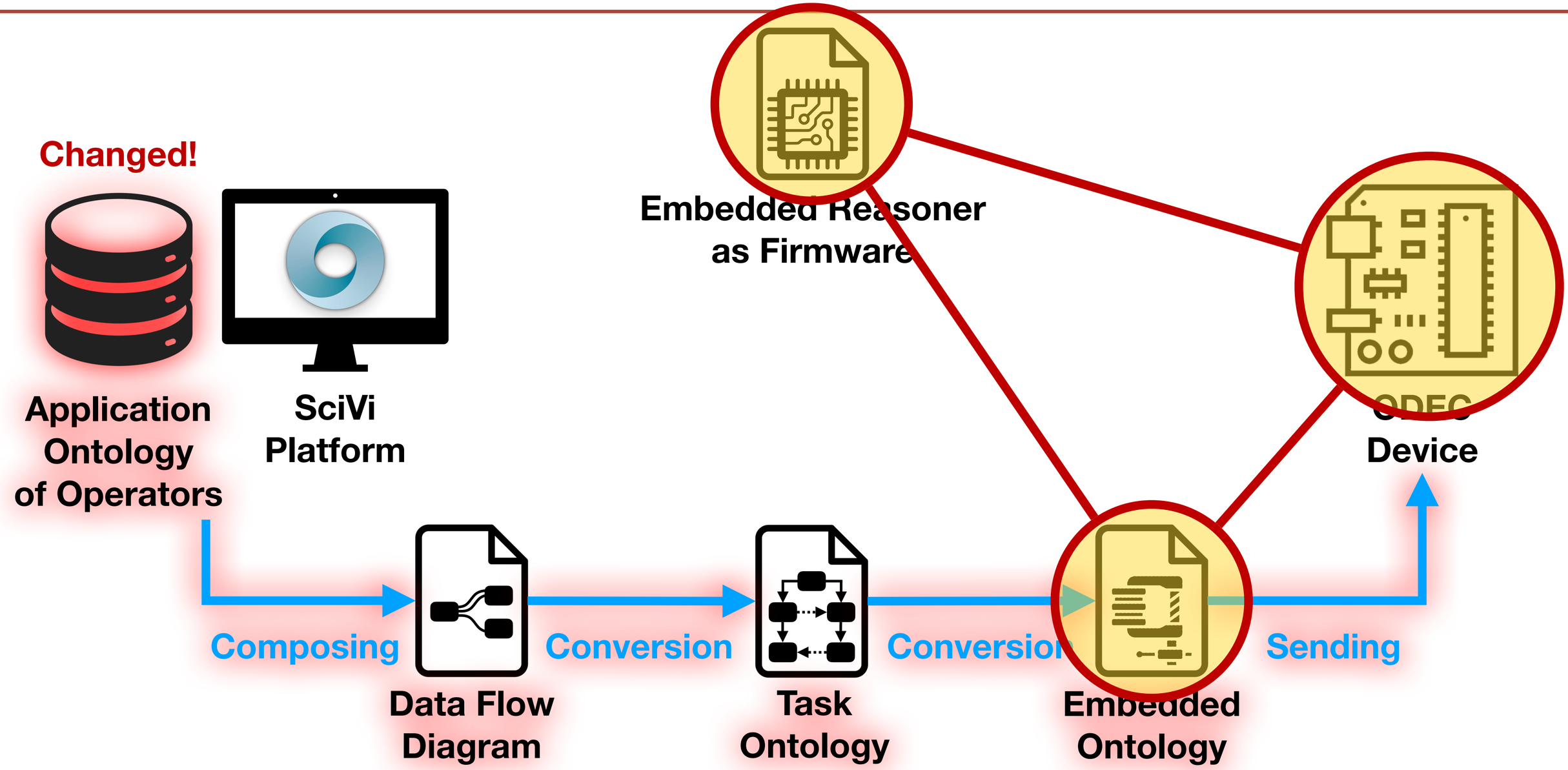
(Ryabinin, 2020)

Ontology-Driven Edge Computing (ODEC)

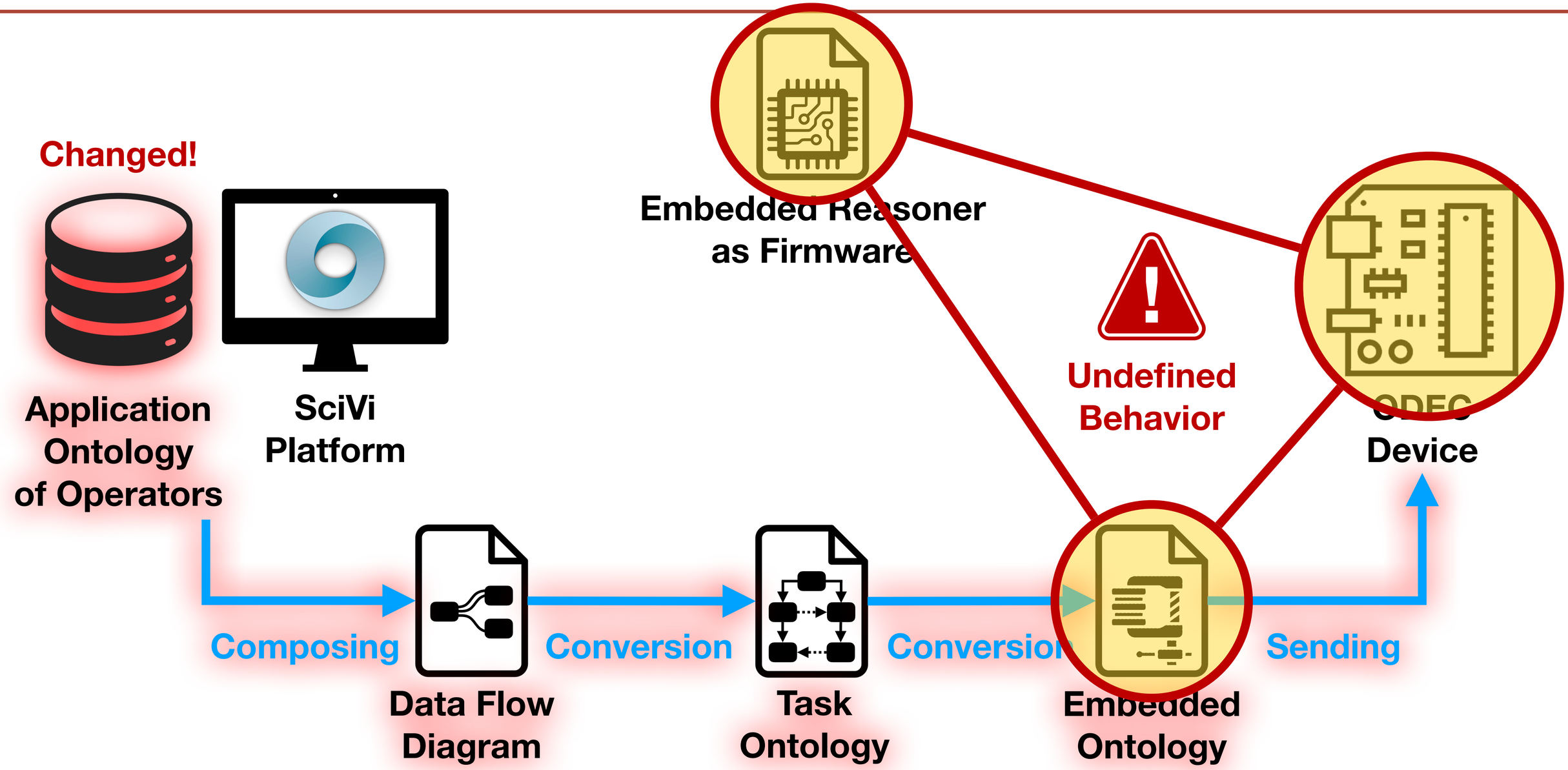


(Ryabinin, 2020)

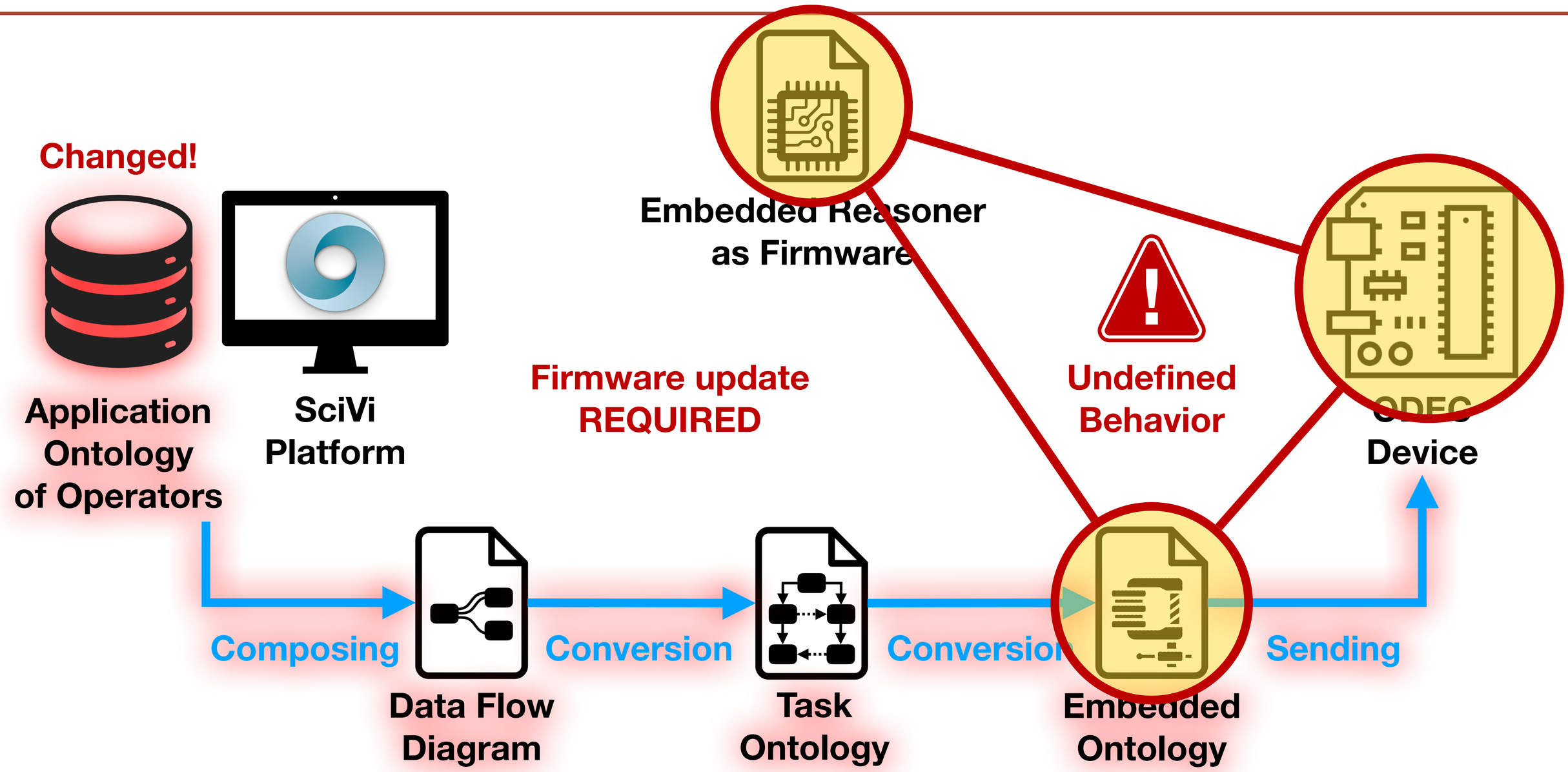
Compatibility Uncertainty



Compatibility Uncertainty

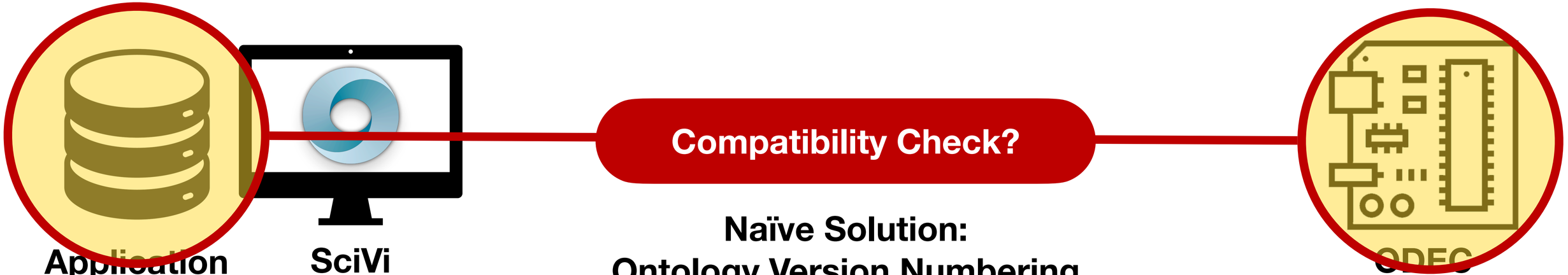


Compatibility Uncertainty



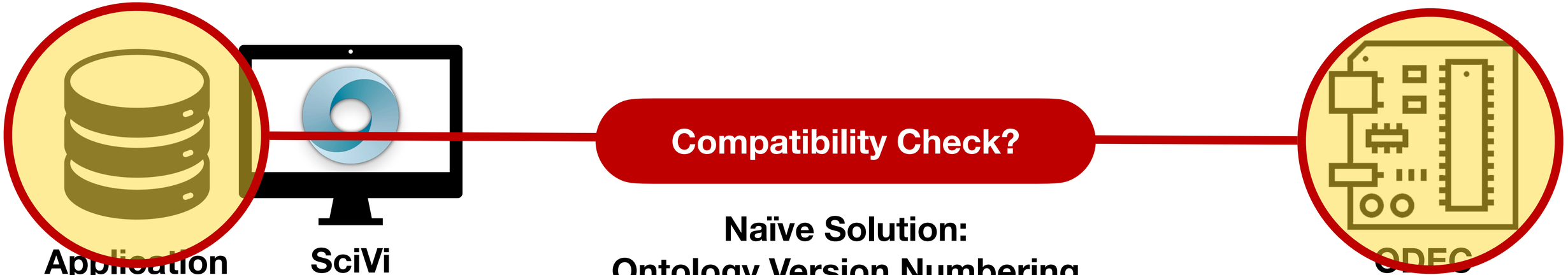
How to Remedy Compatibility Uncertainty?

How to Remedy Compatibility Uncertainty?



Naïve Solution:
Ontology Version Numbering

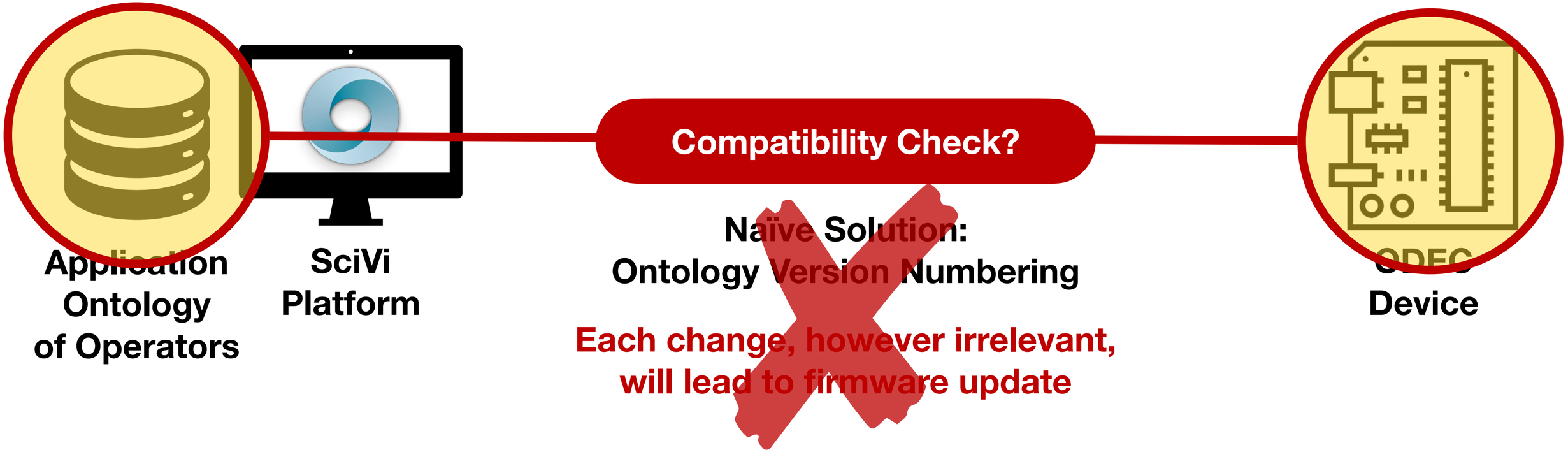
How to Remedy Compatibility Uncertainty?



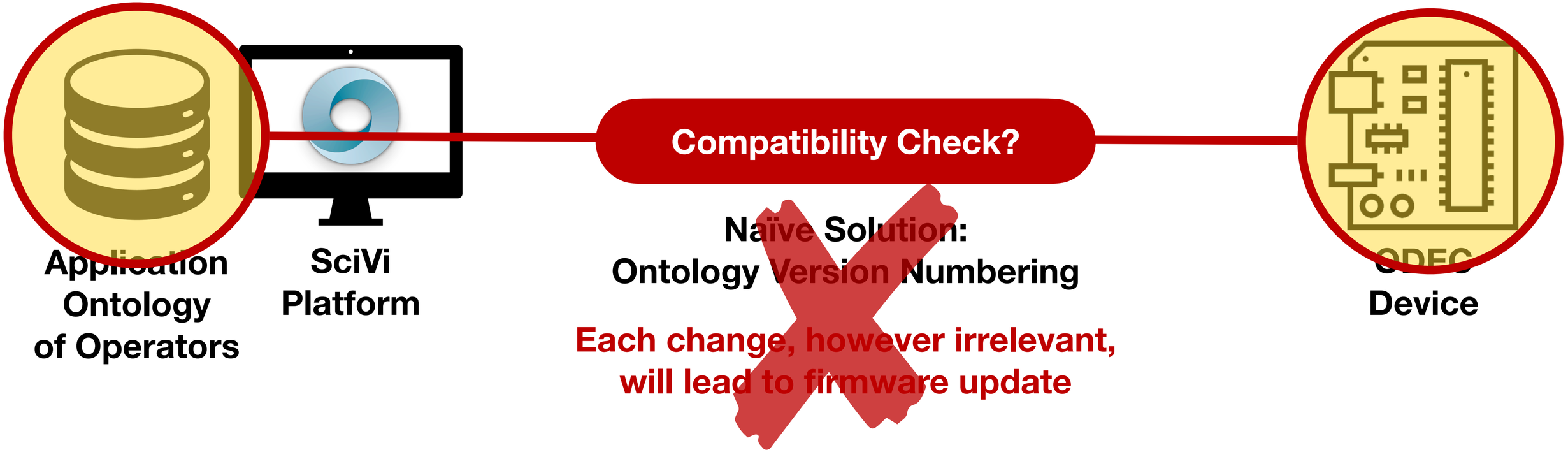
Naïve Solution:
Ontology Version Numbering

Each change, however irrelevant, will lead to firmware update

How to Remedy Compatibility Uncertainty?



How to Remedy Compatibility Uncertainty?



NEW

Semantic Hashing!

Semantic Hashing of an Operator: Signature

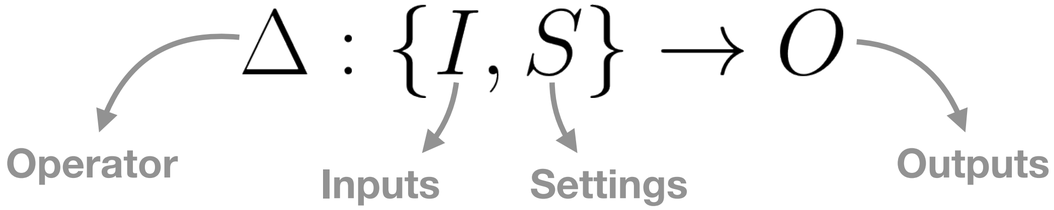
Semantic Hashing of an Operator: Signature

Math Model of an Operator

$$\Delta : \{I, S\} \rightarrow O$$

Semantic Hashing of an Operator: Signature

Math Model of an Operator

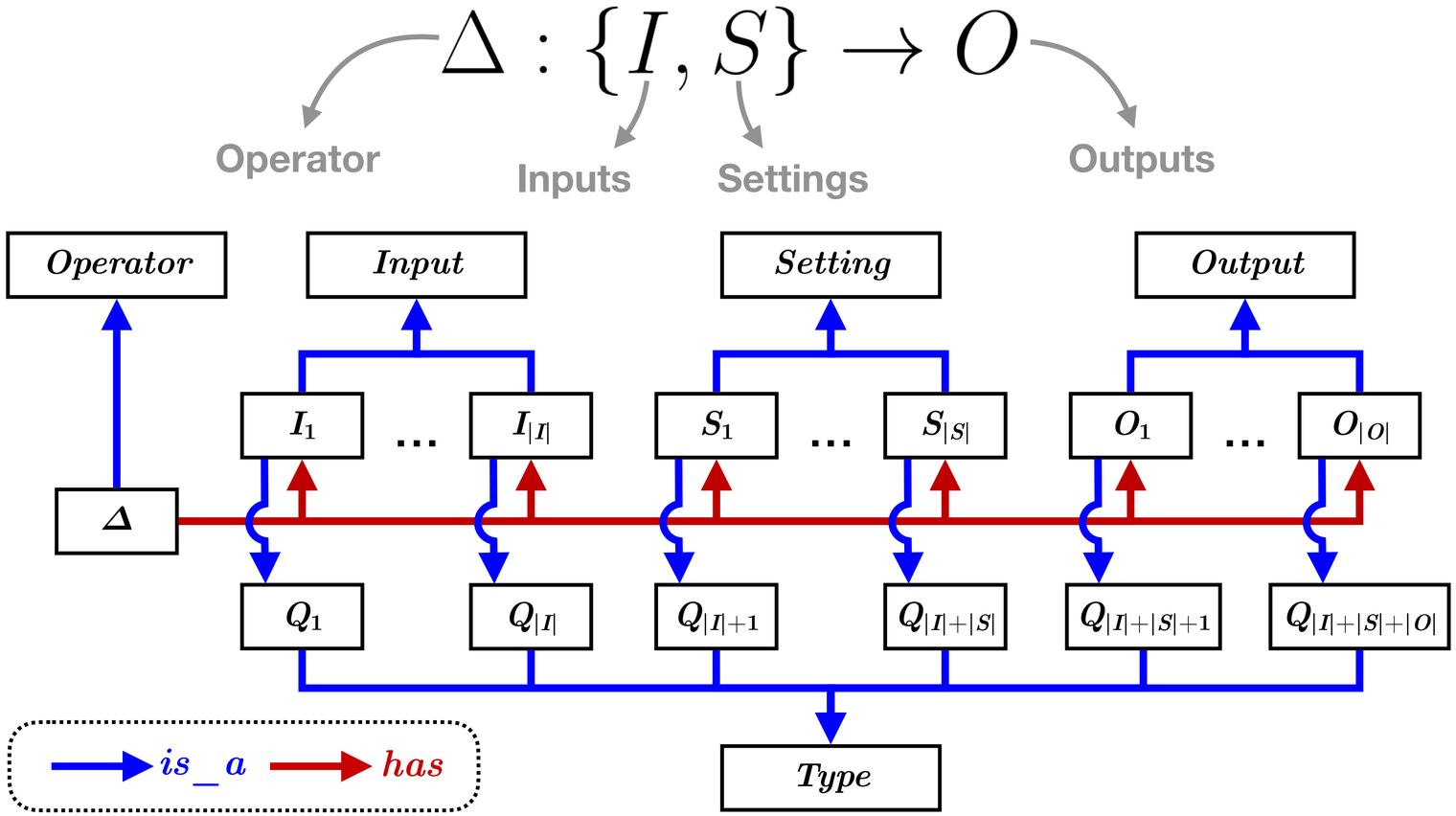


Semantic Hashing of an Operator: Signature

Math Model of an Operator

Ontological Description of an Operator

(equivalence theorem is proven in the paper)



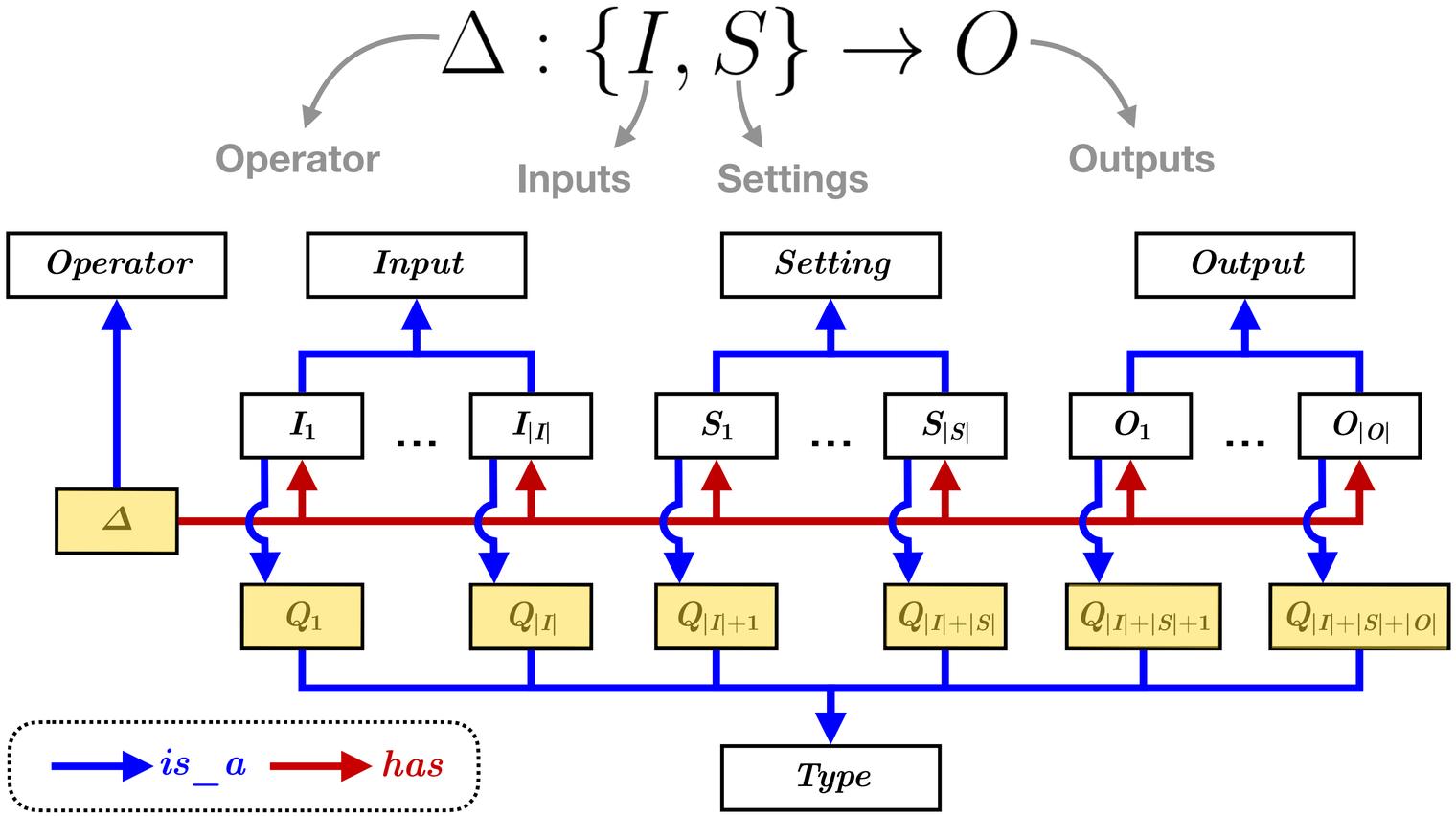
Semantic Hashing of an Operator: Signature

Math Model of an Operator

Ontological Description of an Operator
(equivalence theorem is proven in the paper)

"Signature" of an Operator

NEW



Semantic Hashing of an Operator: Signature

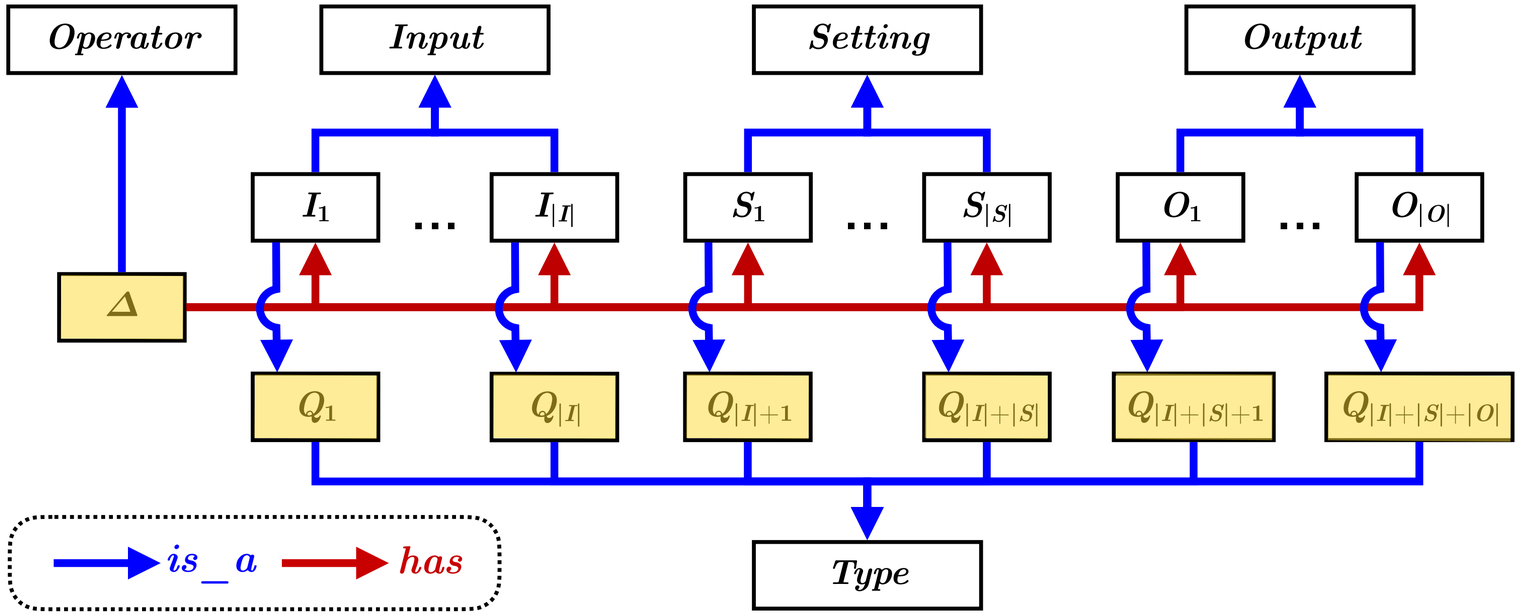
Math Model of an Operator

$$\Delta : \{I, S\} \rightarrow O$$

Operator — Inputs — Settings — Outputs

Ontological Description of an Operator

(equivalence theorem is proven in the paper)



"Signature" of an Operator

NEW

String Representation of the "Signature"

$$\sigma(\Delta) = \text{name}(\Delta) + \text{“@I”} + \sum_{i=1}^{|I|} \text{name}(Q_i) + \text{“@S”} + \sum_{i=|I|+1}^{|I|+|S|} \text{name}(Q_i) + \text{“@O”} + \sum_{i=|I|+|S|+1}^{|I|+|S|+|O|} \text{name}(Q_i)$$

Semantic Hashing of an Operator: Signature

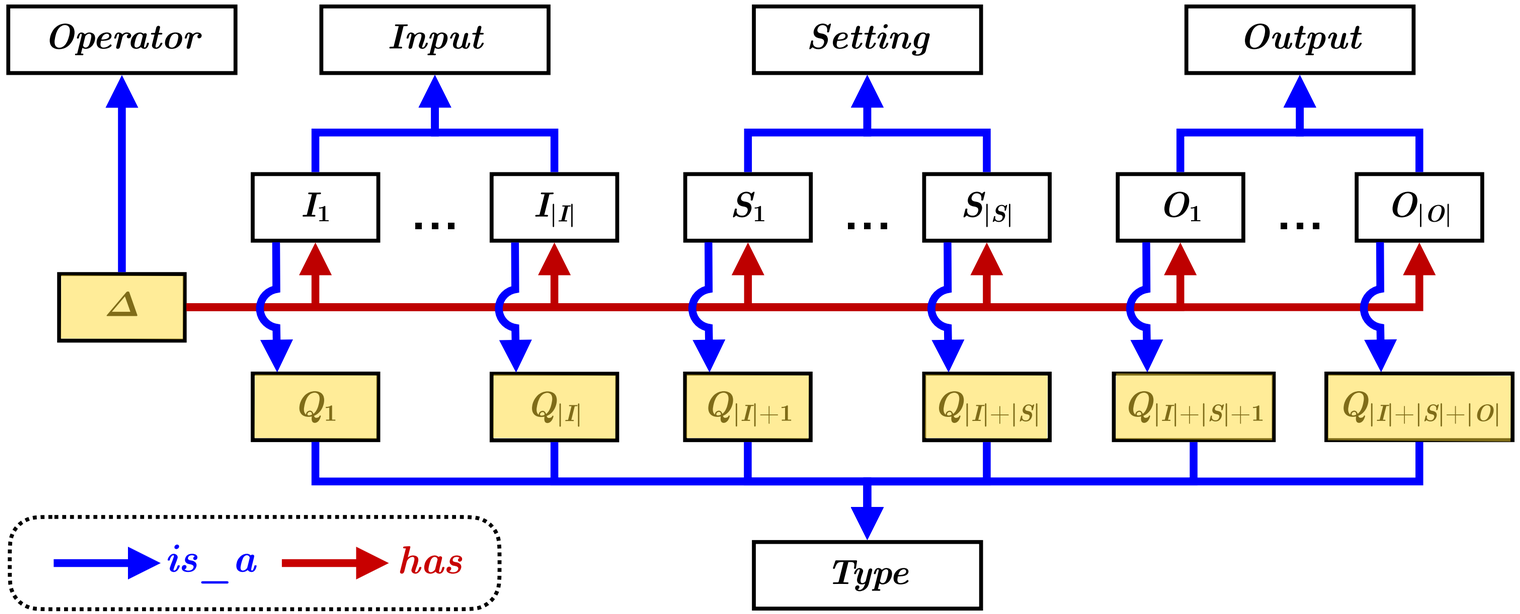
Math Model of an Operator

$$\Delta : \{I, S\} \rightarrow O$$

Operator — Inputs — Settings — Outputs

Ontological Description of an Operator

(equivalence theorem is proven in the paper)



"Signature" of an Operator

NEW

String Representation of the "Signature"

$$\sigma(\Delta) = \text{name}(\Delta) + \text{“@I”} + \sum_{i=1}^{|I|} \text{name}(Q_i) + \text{“@S”} + \sum_{i=|I|+1}^{|I|+|S|} \text{name}(Q_i) + \text{“@O”} + \sum_{i=|I|+|S|+1}^{|I|+|S|+|O|} \text{name}(Q_i)$$

">"-delimitered hierarchy concatenation

":"-delimitered string concatenation

Semantic Hashing of an Operator: Signature

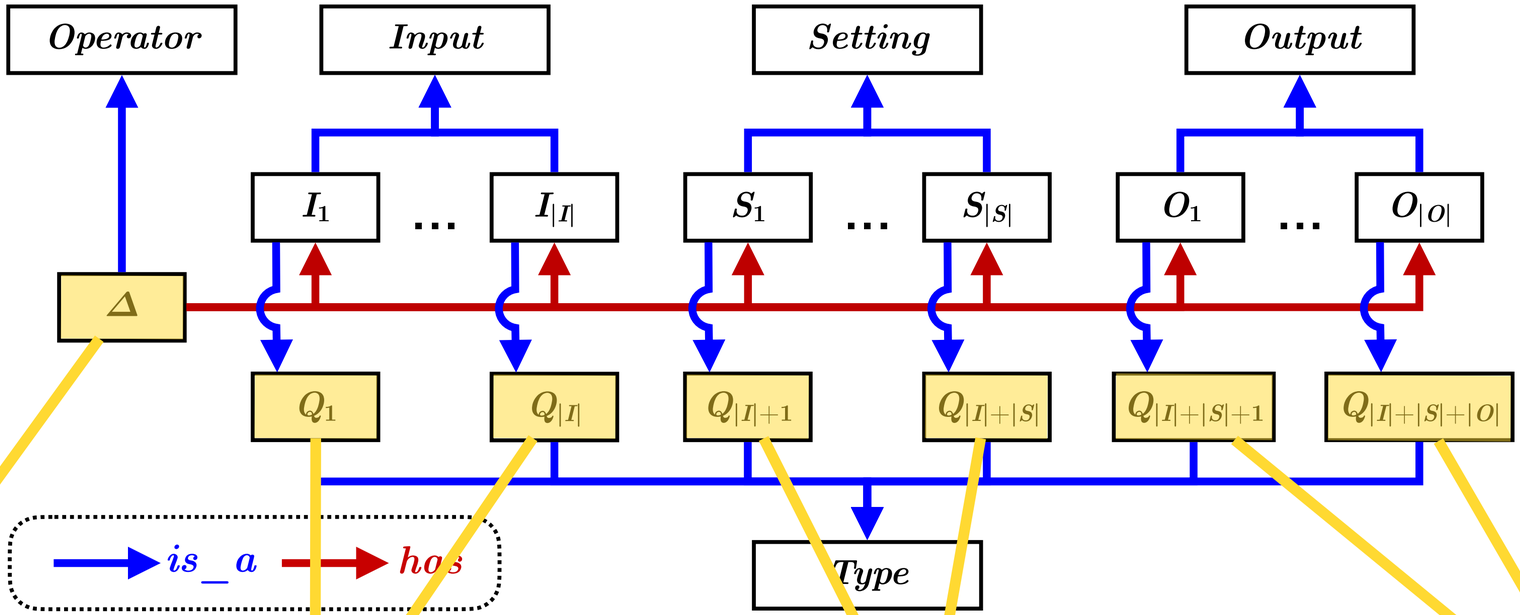
Math Model of an Operator

$$\Delta : \{I, S\} \rightarrow O$$

Operator — Inputs — Settings — Outputs

Ontological Description of an Operator

(equivalence theorem is proven in the paper)



"Signature" of an Operator

NEW

String Representation of the "Signature"

$$\sigma(\Delta) = \text{name}(\Delta) + \text{“@I”} + \sum_{i=1}^{|I|} \text{name}(Q_i) + \text{“@S”} + \sum_{i=|I|+1}^{|I|+|S|} \text{name}(Q_i) + \text{“@O”} + \sum_{i=|I|+|S|+1}^{|I|+|S|+|O|} \text{name}(Q_i)$$

">"-delimitered hierarchy concatenation

":"-delimitered string concatenation

Semantic Hashing of an Operator: Hash Sum

$$\sigma(\Delta) = \text{name}(\Delta) + \text{“@I”} + \sum_{i=1}^{|I|} \text{name}(Q_i) + \text{“@S”} + \sum_{i=|I|+1}^{|I|+|S|} \text{name}(Q_i) + \text{“@O”} + \sum_{i=|I|+|S|+1}^{|I|+|S|+|O|} \text{name}(Q_i)$$

NEW

$$\pi(\Delta) = \text{Pearson}(\sigma(\Delta))$$

Semantic Hashing of an Operator: Hash Sum

$$\sigma(\Delta) = \text{name}(\Delta) + \text{“@I”} + \sum_{i=1}^{|I|} \text{name}(Q_i) + \text{“@S”} + \sum_{i=|I|+1}^{|I|+|S|} \text{name}(Q_i) + \text{“@O”} + \sum_{i=|I|+|S|+1}^{|I|+|S|+|O|} \text{name}(Q_i)$$

NEW

$$\pi(\Delta) = \text{Pearson}(\sigma(\Delta))$$

2-bytes Pearson hash with custom lookup table

Semantic Hashing of an Operator: Hash Sum

$$\sigma(\Delta) = \text{name}(\Delta) + \text{“@I”} + \sum_{i=1}^{|I|} \text{name}(Q_i) + \text{“@S”} + \sum_{i=|I|+1}^{|I|+|S|} \text{name}(Q_i) + \text{“@O”} + \sum_{i=|I|+|S|+1}^{|I|+|S|+|O|} \text{name}(Q_i)$$

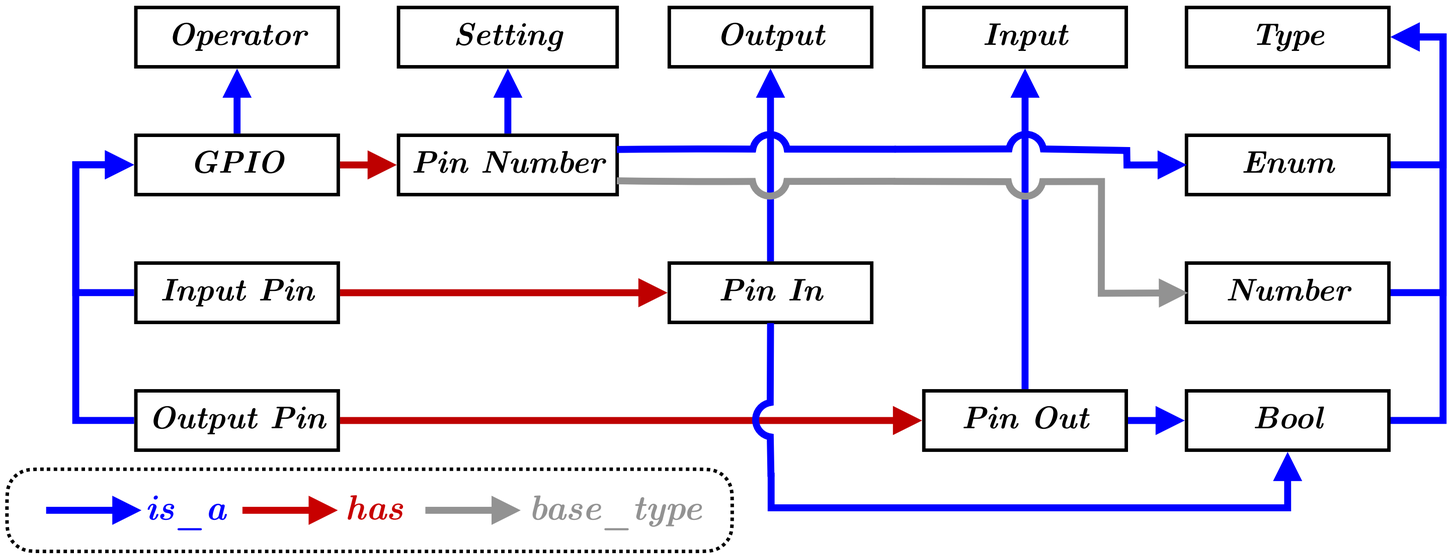
NEW

$$\pi(\Delta) = \text{Pearson}(\sigma(\Delta))$$

2-bytes Pearson hash with custom lookup table

```
def __init__(self):
    self.table = [ \
        29,  186, 180, 162, 184, 218, 3,   141, 55,  0,   72,  98,  226, 108, 220, \
        158, 231, 248, 247, 251, 130, 46,  174, 135, 170, 127, 163, 109, 229, 36,  \
        45,  145, 79,  137, 122, 12,  182, 117, 17,  198, 204, 212, 39,  189, 52,  \
        200, 102, 149, 15,  124, 233, 64,  88,  225, 105, 183, 131, 114, 187, 197, \
        165, 48,  56,  214, 227, 41,  95,  4,   93,  243, 239, 38,  61,  116, 51,  \
        90,  236, 89,  18,  196, 213, 42,  96,  104, 27,  11,  21,  203, 250, 194, \
        57,  85,  54,  211, 32,  25,  140, 121, 147, 171, 6,   115, 234, 206, 101, \
        8,   7,   33,  112, 159, 28,  240, 238, 92,  249, 22,  129, 208, 118, 125, \
        179, 24,  178, 143, 156, 63,  207, 164, 103, 172, 71,  157, 185, 199, 128, \
        181, 175, 193, 154, 152, 176, 26,  9,   132, 62,  151, 2,   97,  205, 120, \
        77,  190, 150, 146, 50,  23,  155, 47,  126, 119, 254, 40,  241, 192, 144, \
        83,  138, 49,  113, 160, 74,  70,  253, 217, 110, 58,  5,   228, 136, 87,  \
        215, 169, 14,  168, 73,  219, 167, 10,  148, 173, 100, 35,  222, 76,  221, \
        139, 235, 16,  69,  166, 133, 210, 67,  30,  84,  43,  202, 161, 195, 223, \
        53,  34,  232, 245, 237, 230, 59,  80,  191, 91,  66,  209, 75,  78,  44,  \
        65,  1,   188, 252, 107, 86,  177, 242, 134, 13,  246, 99,  20,  81,  111, \
        68,  153, 37,  123, 216, 224, 19,  31,  82,  106, 201, 244, 60,  142, 94,  \
        255                                                                        \
    ]
def hash_key(self, key) -> int:
    hashLen = 2
    result = 0
    for j in range(hashLen):
        h = self.table[(ord(key[0]) + j) % 256]
        for i in range(1, len(key)):
            h = self.table[(h ^ ord(key[i])) % 256]
        h = self.table[(h ^ len(key)) % 256]
        result = (result << 8) | h
    return result
```

Semantic Hashing of an Operator: Example



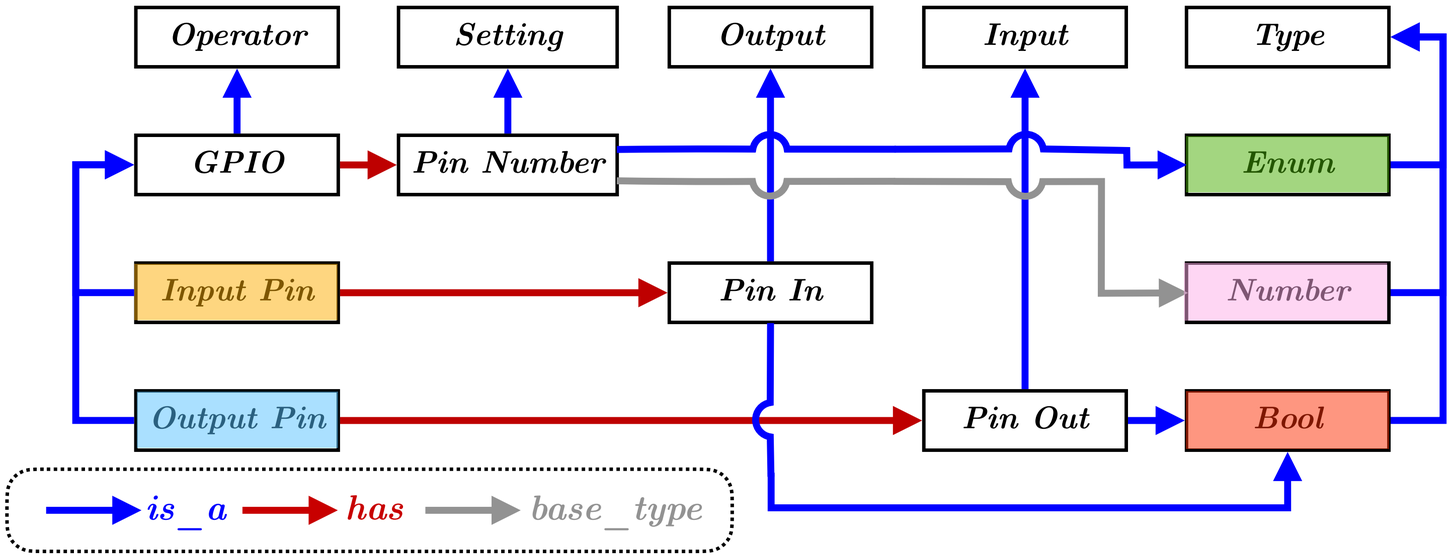
$$\sigma(Input\ Pin) = \text{“}\texttt{Input Pin@SEnum>Number@OBool}\text{”}$$

$$\sigma(Output\ Pin) = \text{“}\texttt{Output Pin@IBool@SEnum>Number}\text{”}$$

$$\pi(Input\ Pin) = 19218$$

$$\pi(Output\ Pin) = 57372$$

Semantic Hashing of an Operator: Example



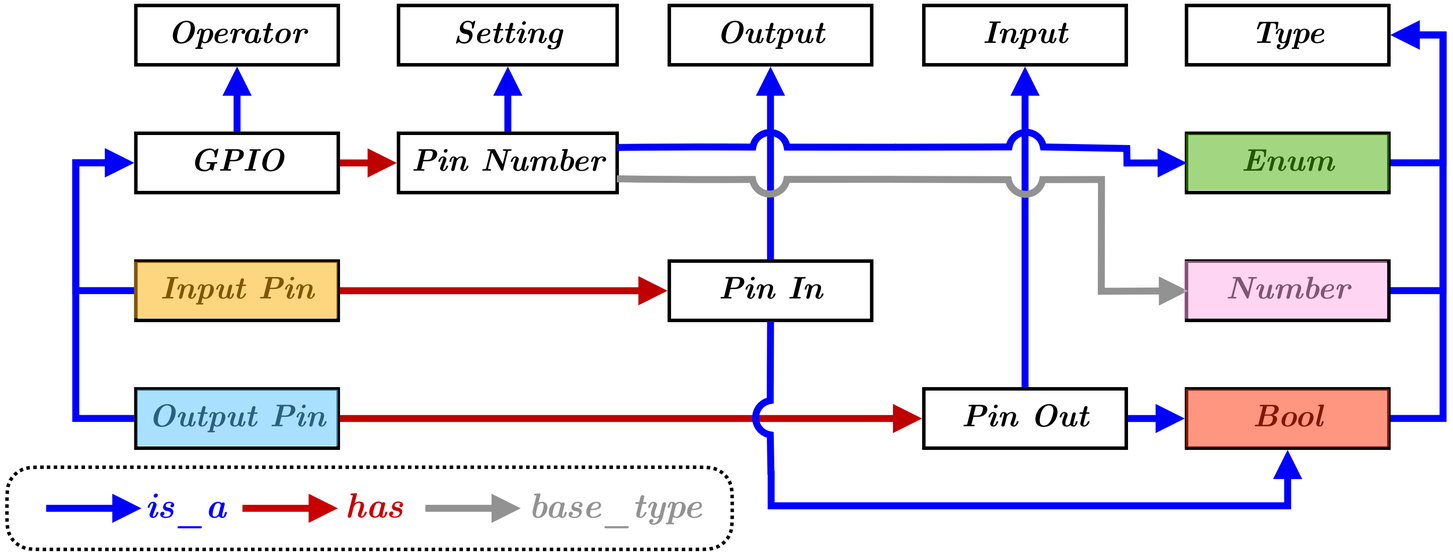
$$\sigma(Input\ Pin) = \text{“Input Pin@SEnum>Number@OBool”}$$

$$\sigma(Output\ Pin) = \text{“Output Pin@IBool@SEnum>Number”}$$

$$\pi(Input\ Pin) = 19218$$

$$\pi(Output\ Pin) = 57372$$

Semantic Hashing of an Operator: Example



$$\sigma(Input\ Pin) = \text{“Input Pin@SEnum>Number@OBool”}$$

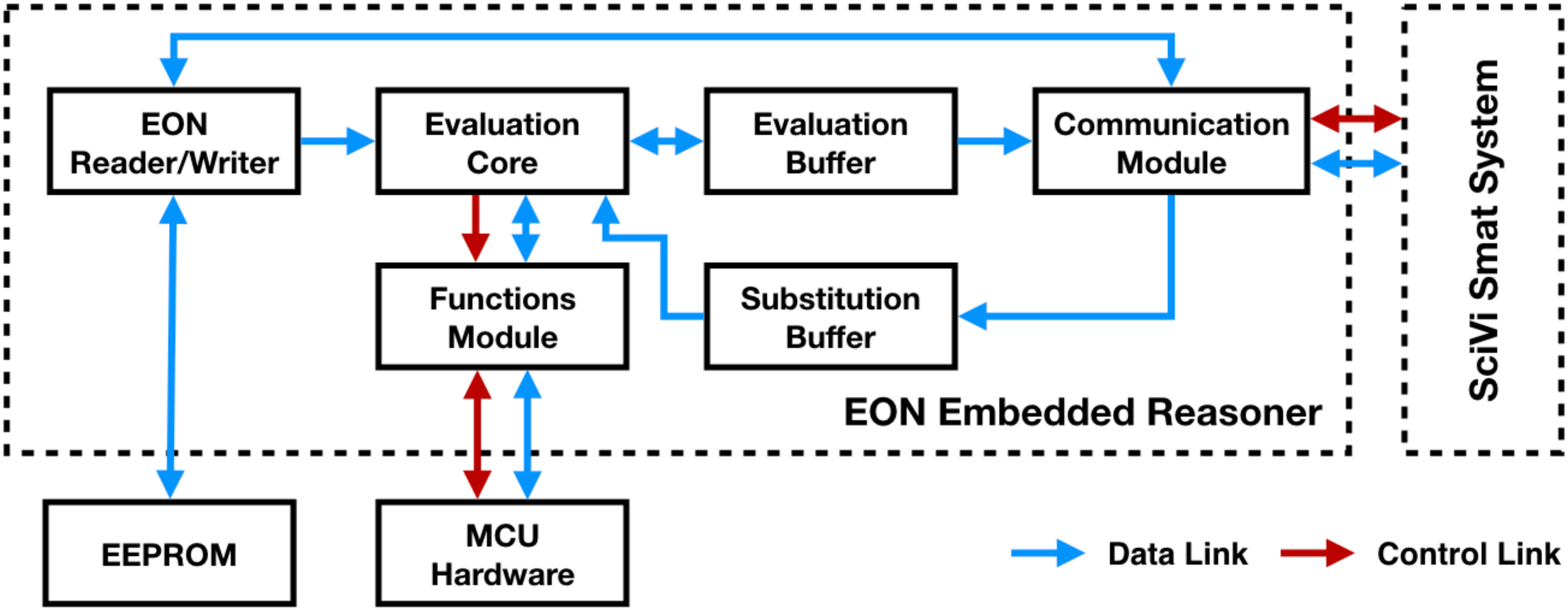
$$\sigma(Output\ Pin) = \text{“Output Pin@IBool@SEnum>Number”}$$

$$\pi(Input\ Pin) = 19218$$

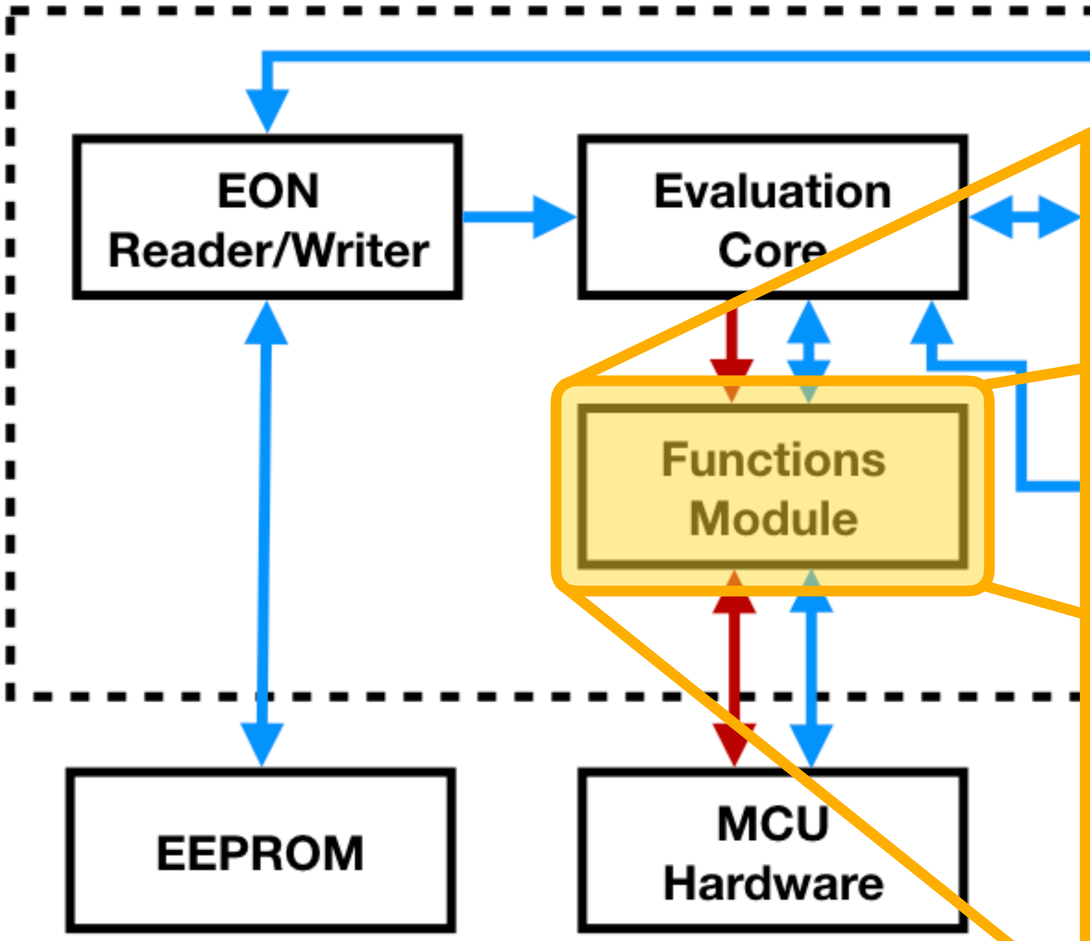
$$\pi(Output\ Pin) = 57372$$

Stored in the embedded reasoner and referenced in the task ontology

Generation of Embedded Reasoner

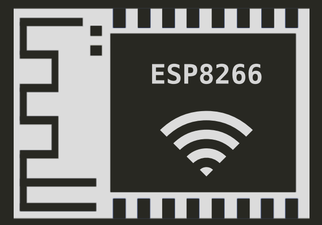


Generation of Embedded Reasoner

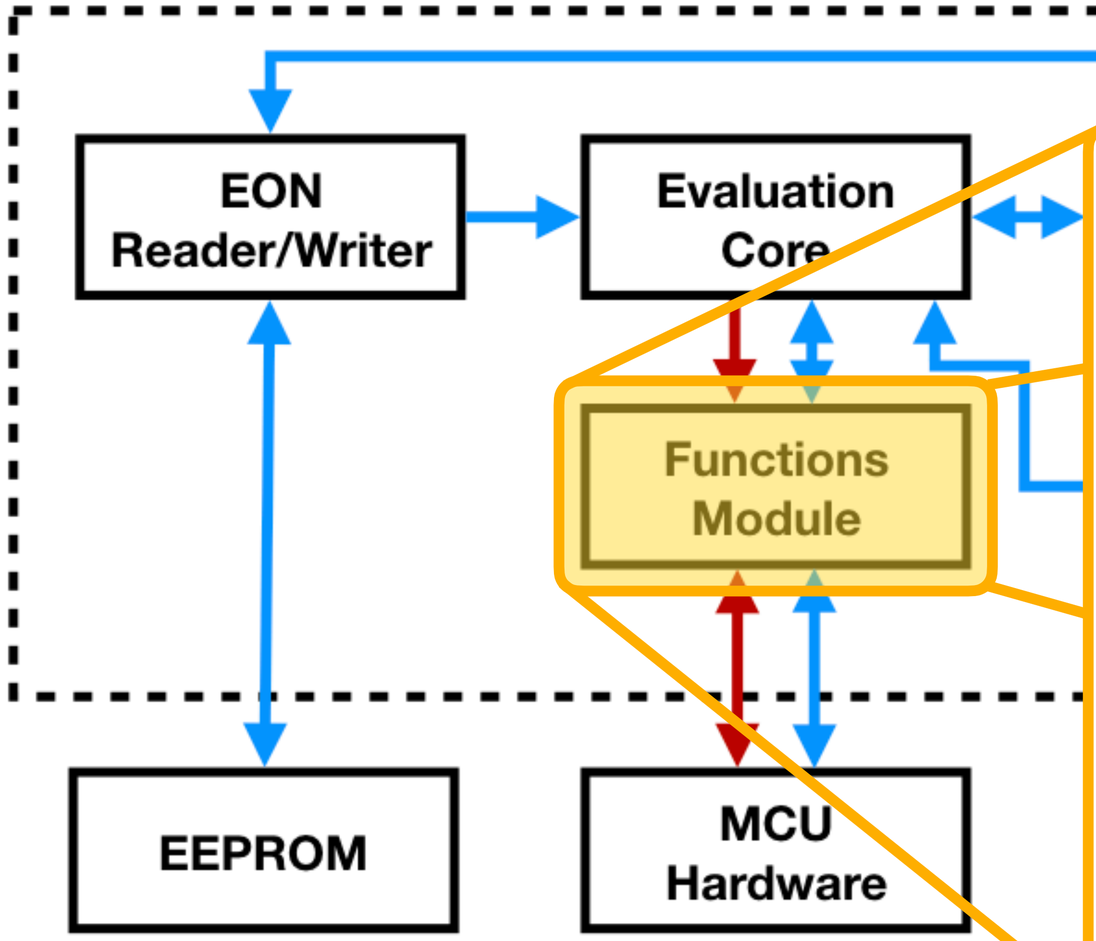



```
1   namespace EON
2   {
3       static void inputPin(OpInstance &opInst)
4       {
5           opInst.output(0) = digitalRead((int)opInst.setting(0)) == HIGH;
6       }
7
8       static void outputPin(OpInstance &opInst)
9       {
10          digitalWrite((int)opInst.setting(0), (bool)opInst.input() ? HIGH : LOW);
11      }
12
13      static void (*m_operators[])(OpInstance &) =
14      {
15          &inputPin,
16          &outputPin
17      };
18
19      static MotherNodeID m_operatorsTableLen = 2;
20
21      static MotherNodeID m_operatorsTable[] =
22      {
23          19218,
24          57372
25      };
26  };
```

Code auto-generated
by SciVi for

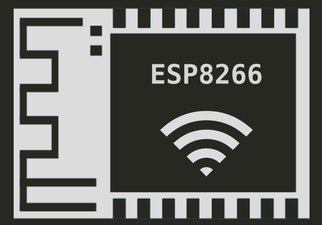


Generation of Embedded Reasoner

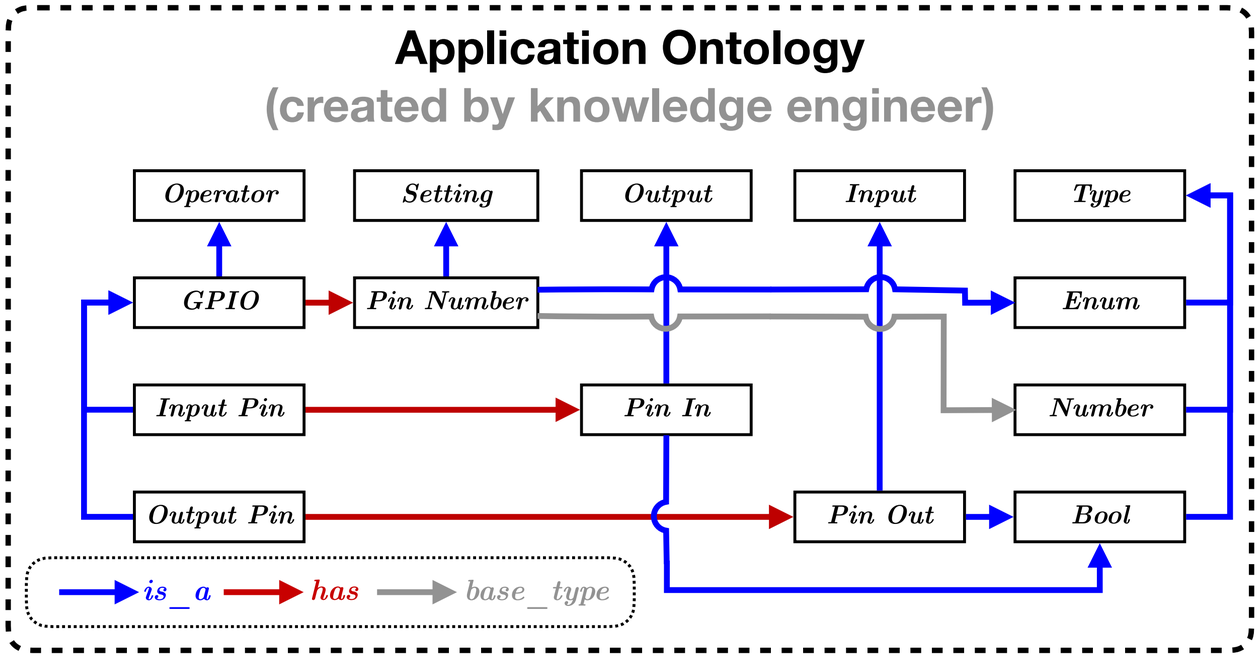



```
1   namespace EON
2   {
3       static void inputPin(OpInstance &opInst)
4       {
5           opInst.output(0) = digitalRead((int)opInst.setting(0)) == HIGH;
6       }
7
8       static void outputPin(OpInstance &opInst)
9       {
10          digitalWrite((int)opInst.setting(0), (bool)opInst.input() ? HIGH : LOW);
11      }
12
13      static void (*m_operators[])(OpInstance &) =
14      {
15          &inputPin,
16          &outputPin
17      };
18
19      static MotherNodeID m_operatorsTableLen = 2;
20
21      static MotherNodeID m_operatorsTable[] =
22      {
23          19218,
24          57372
25      };
26  };
```

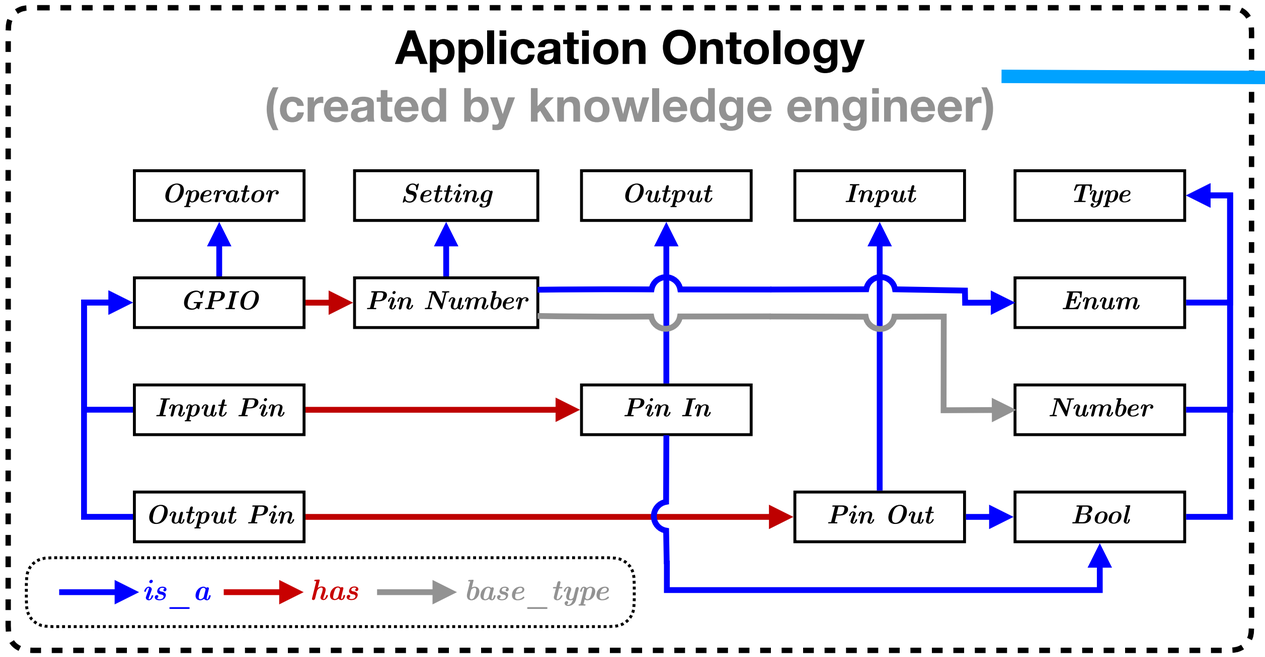
Code auto-generated
by SciVi for

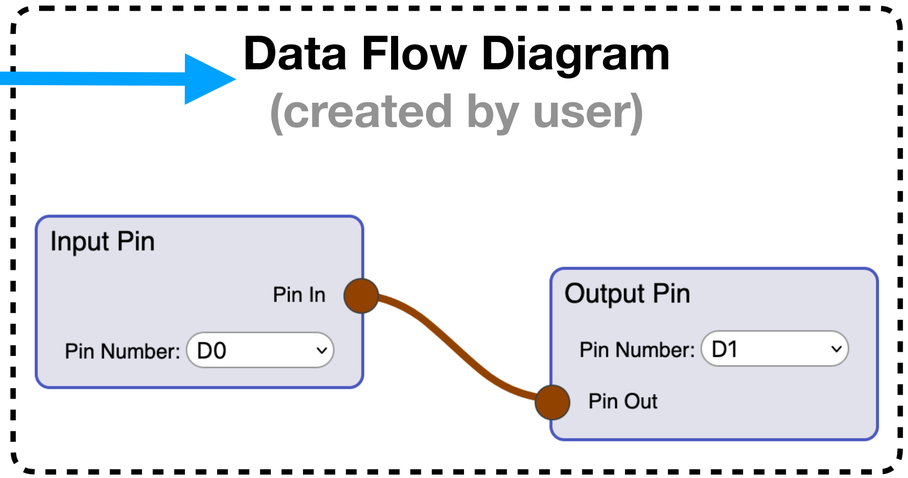


Generation of Task Ontology

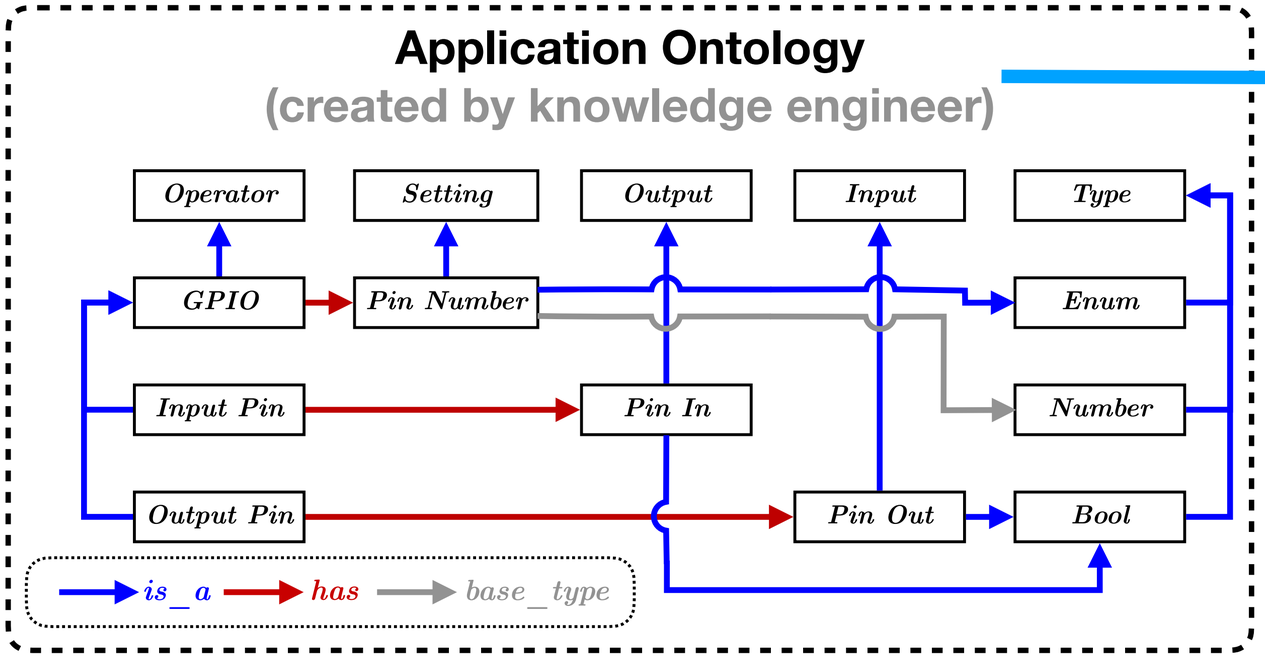


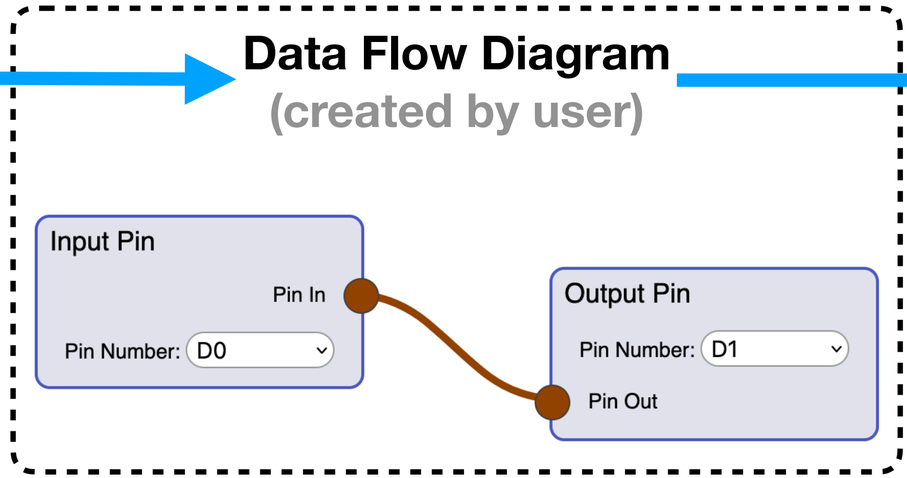
Generation of Task Ontology

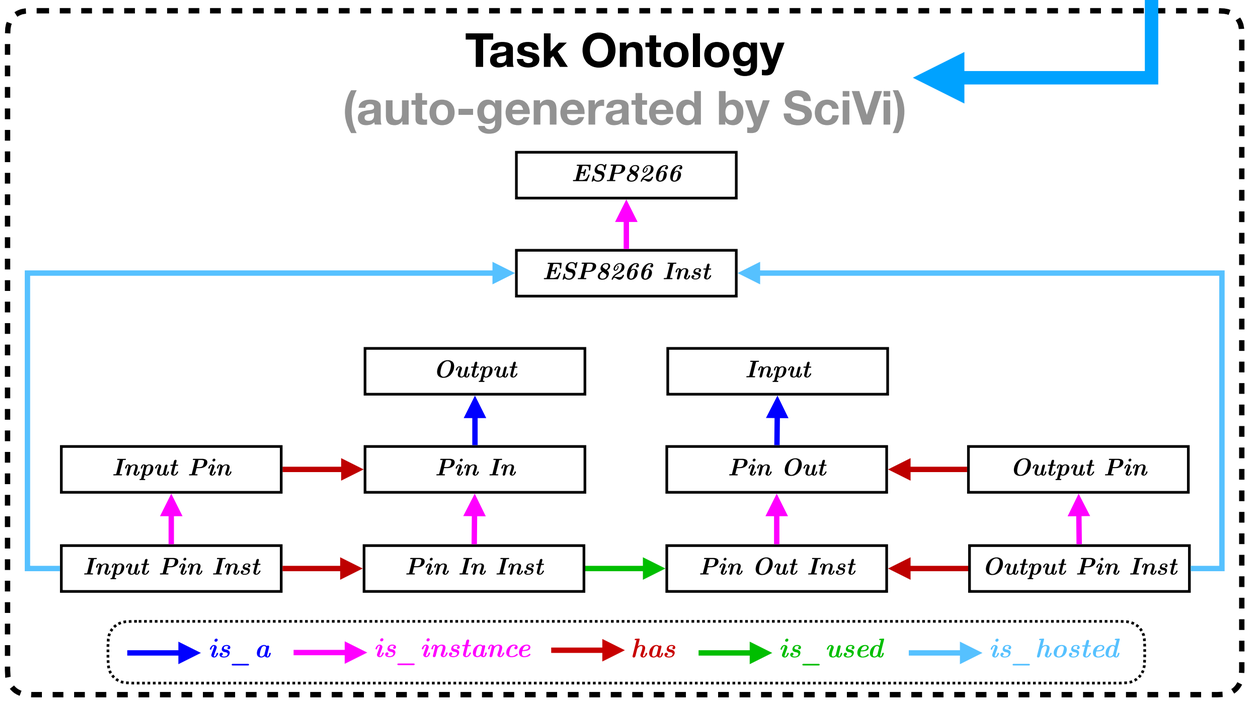




Generation of Task Ontology



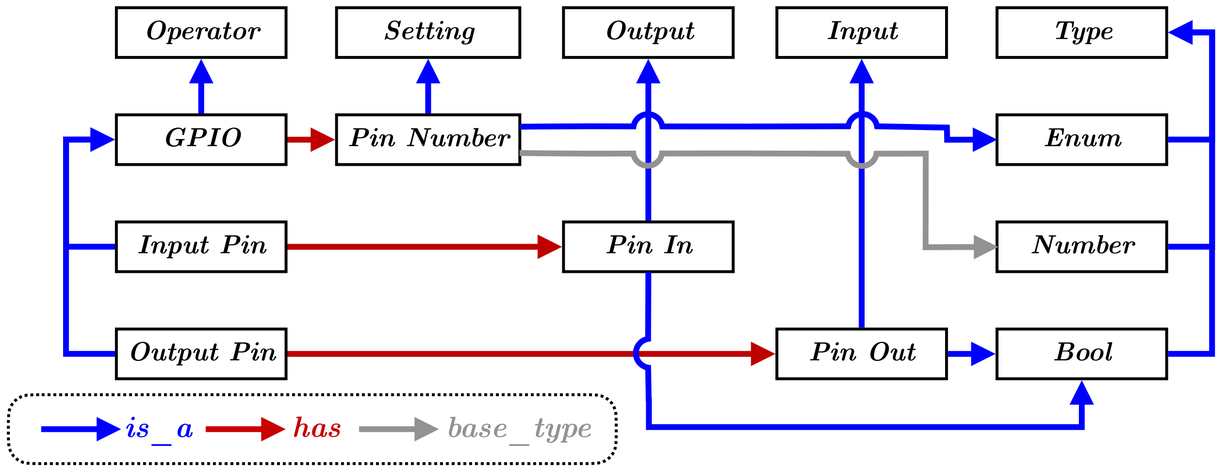




Generation of Task Ontology

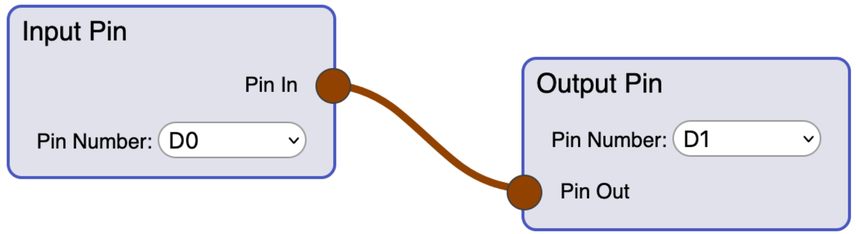
Application Ontology

(created by knowledge engineer)



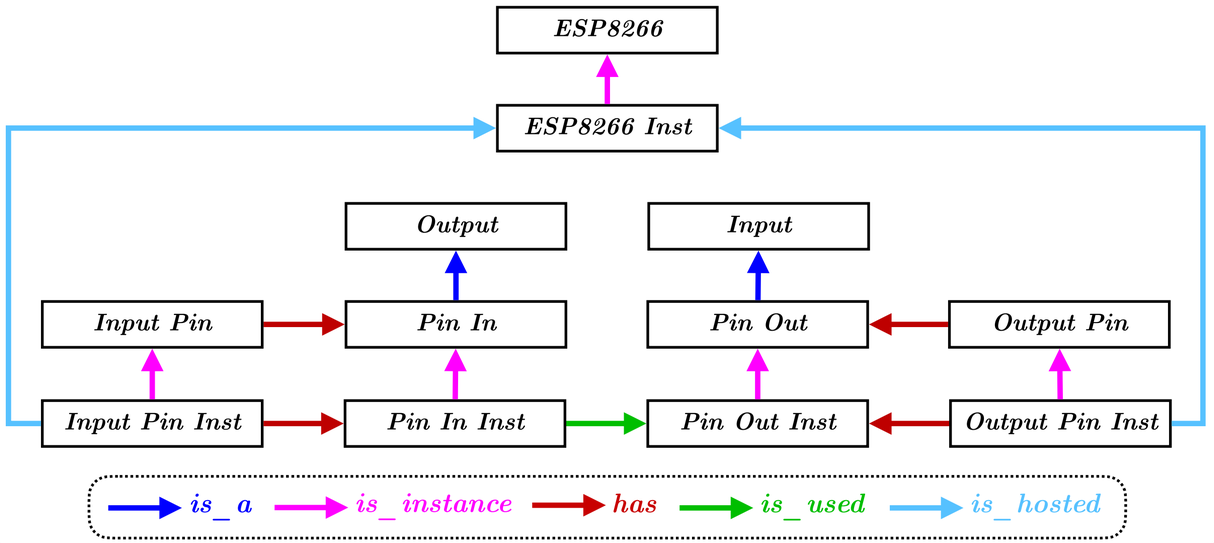
Data Flow Diagram

(created by user)



Task Ontology

(auto-generated by SciVi)



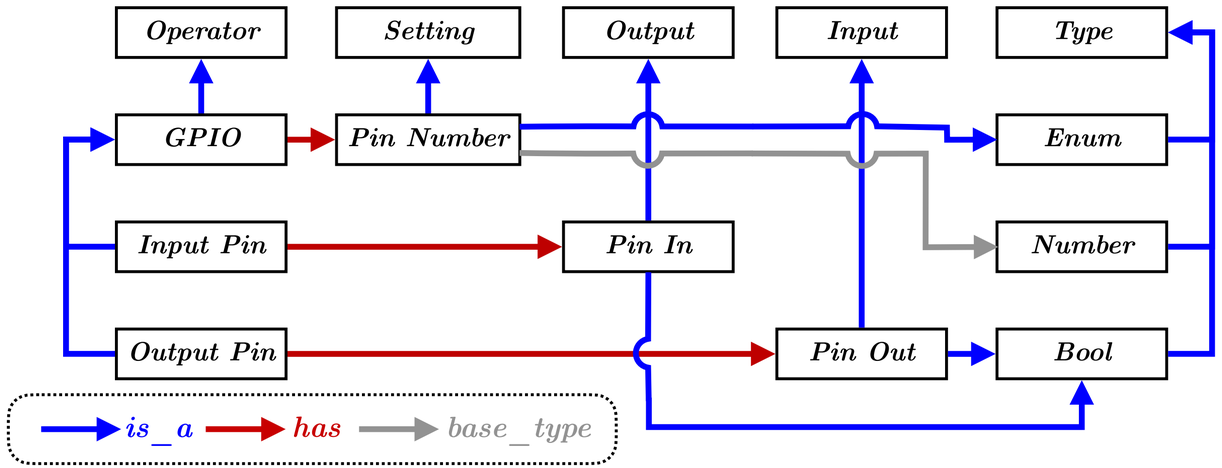
Embedded Ontology

(auto-generated by SciVi; 20 bytes only)

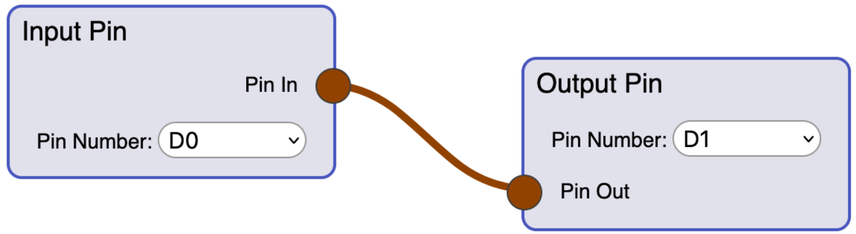
```
{ 0x01, 0x01, 0x00, 0x02,
  0x00, 0x06, 0x01, 0x00,
  0x00, 0x02, 0x00, 0x01,
  0x4b, 0x12, 0x01, 0x00,
  0xe0, 0x1c, 0x02, 0x00 }
```

Generation of Task Ontology

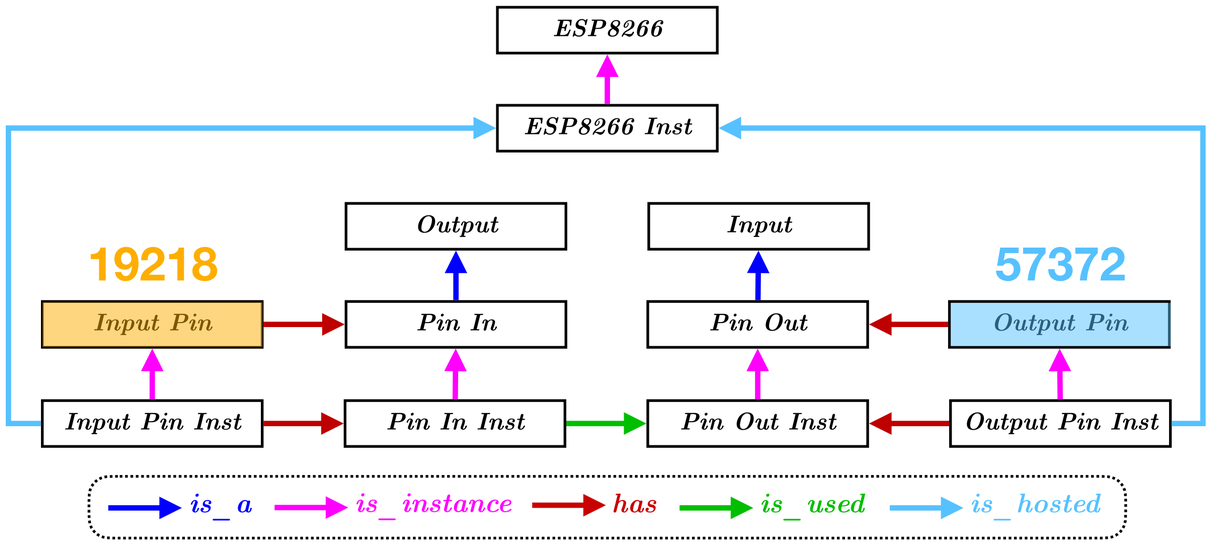
Application Ontology
(created by knowledge engineer)



Data Flow Diagram
(created by user)



Task Ontology
(auto-generated by SciVi)

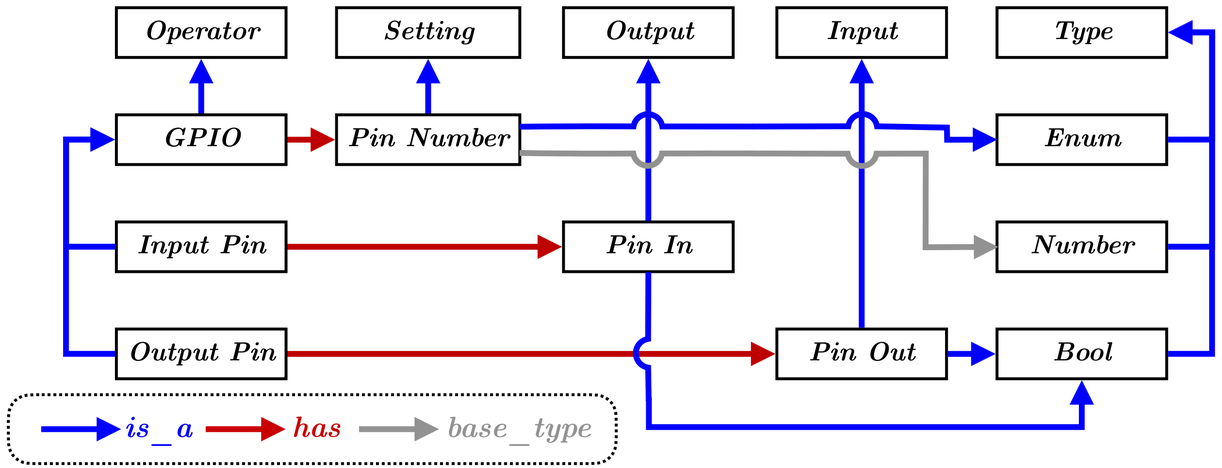


Embedded Ontology
(auto-generated by SciVi; 20 bytes only)

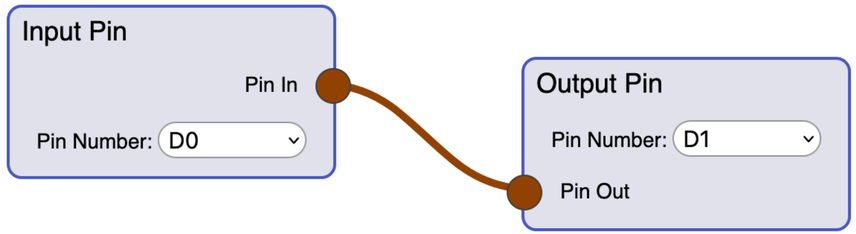
```
{ 0x01, 0x01, 0x00, 0x02,
  0x00, 0x06, 0x01, 0x00,
  0x00, 0x02, 0x00, 0x01,
  0x4b, 0x12, 0x01, 0x00,
  0xe0, 0x1c, 0x02, 0x00 }
```

Generation of Task Ontology

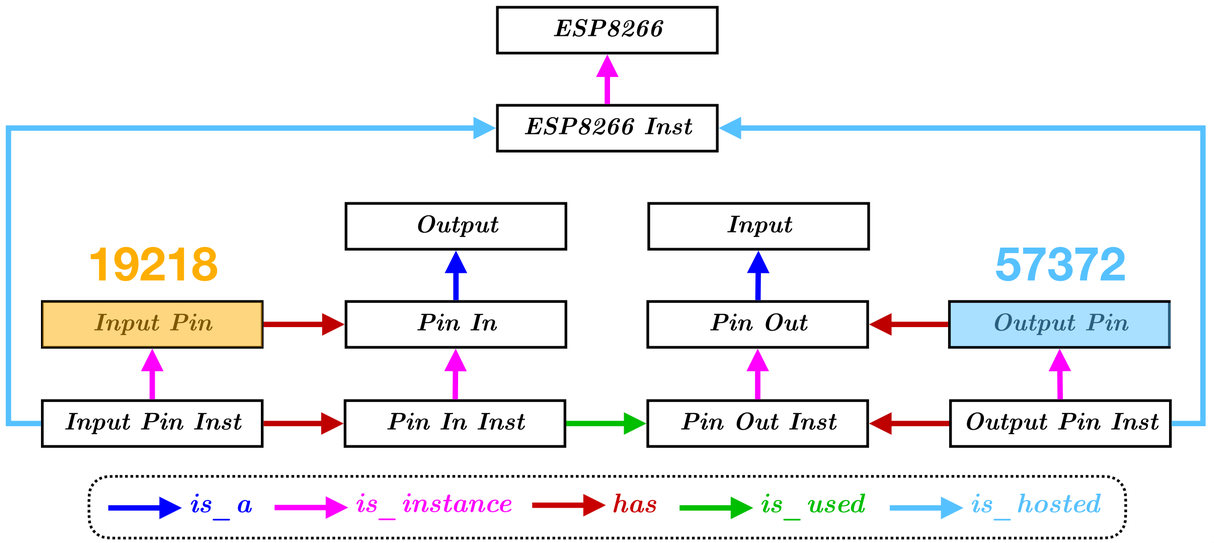
Application Ontology
(created by knowledge engineer)



Data Flow Diagram
(created by user)



Task Ontology
(auto-generated by SciVi)

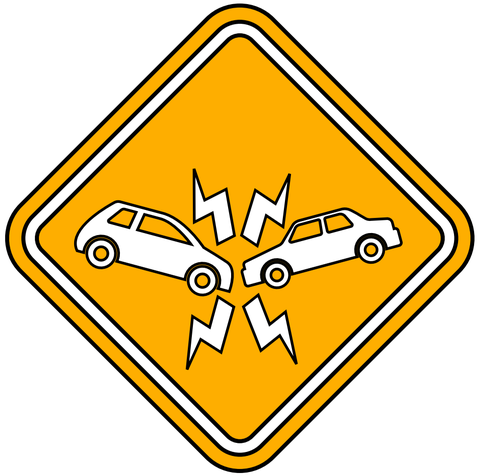


Embedded Ontology
(auto-generated by SciVi; 20 bytes only)

NEW

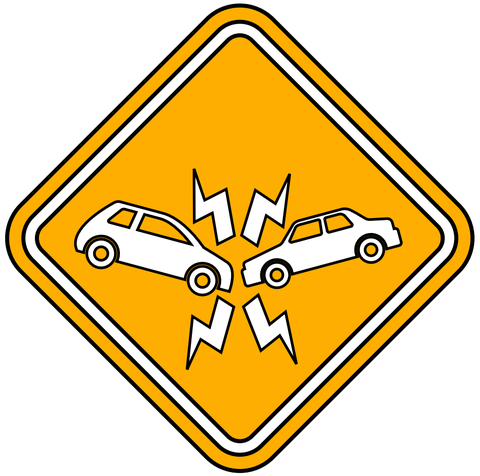
```
{ 0x01, 0x01, 0x00, 0x02,
  0x00, 0x06, 0x01, 0x00,
  0x00, 0x02, 0x00, 0x01,
  0x4b, 0x12, 0x01, 0x00,
  0xe0, 0x1c, 0x02, 0x00 }
```

Collisions of Semantic Hash

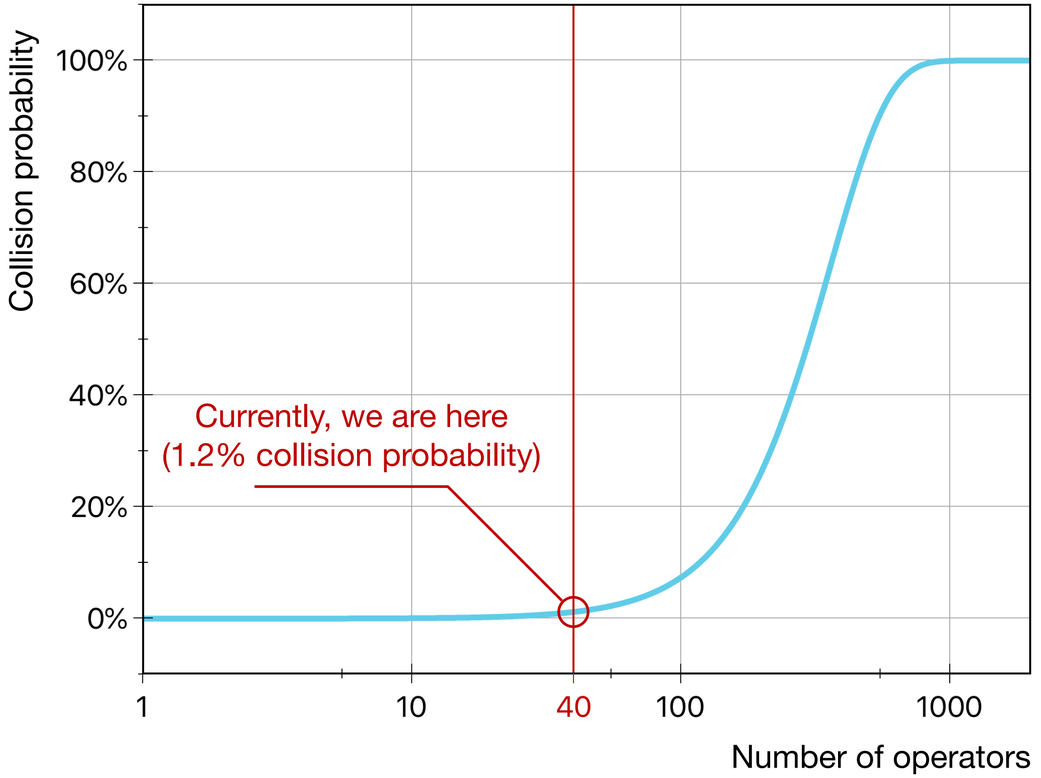


Collision Danger:
2-Bytes Hash

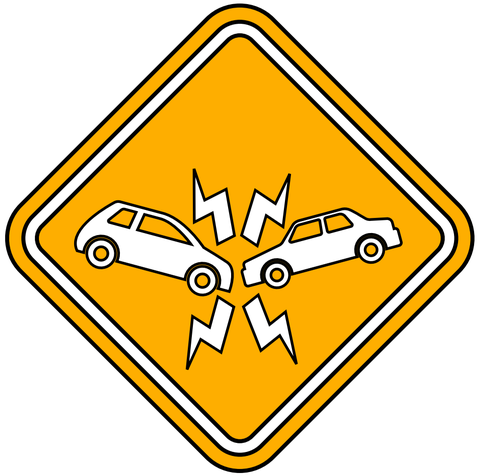
Collisions of Semantic Hash



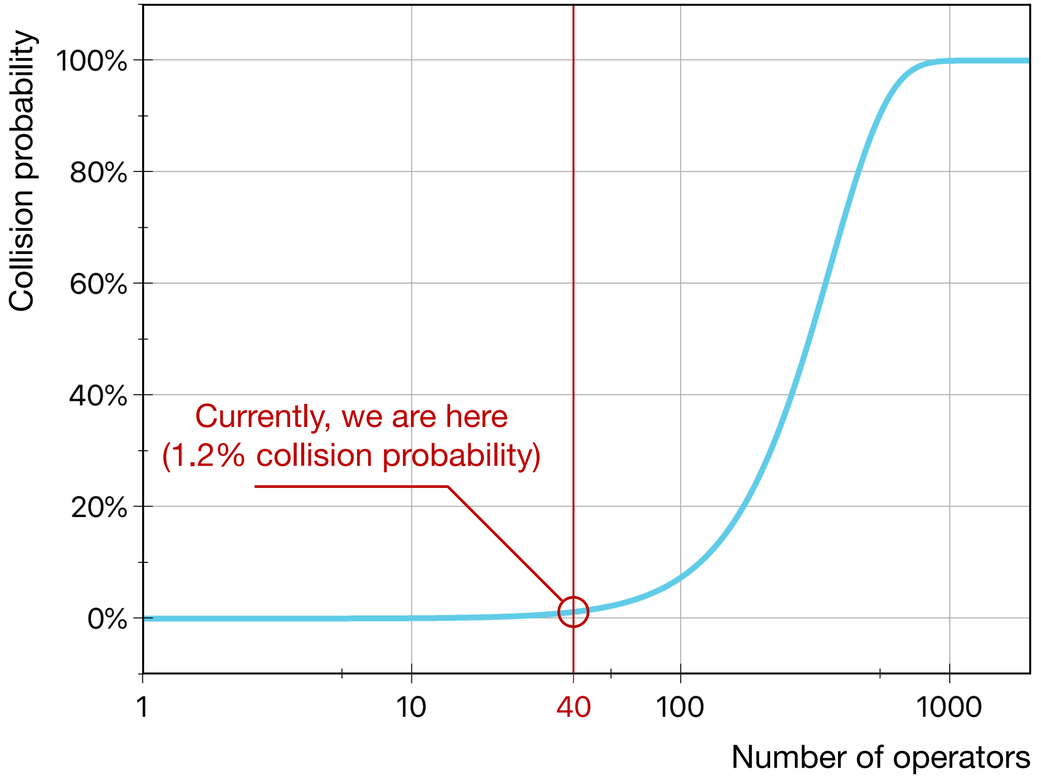
Collision Danger:
2-Bytes Hash



Collisions of Semantic Hash

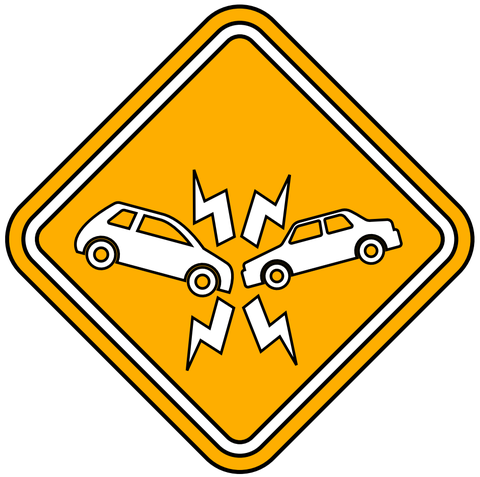


Collision Danger:
2-Bytes Hash

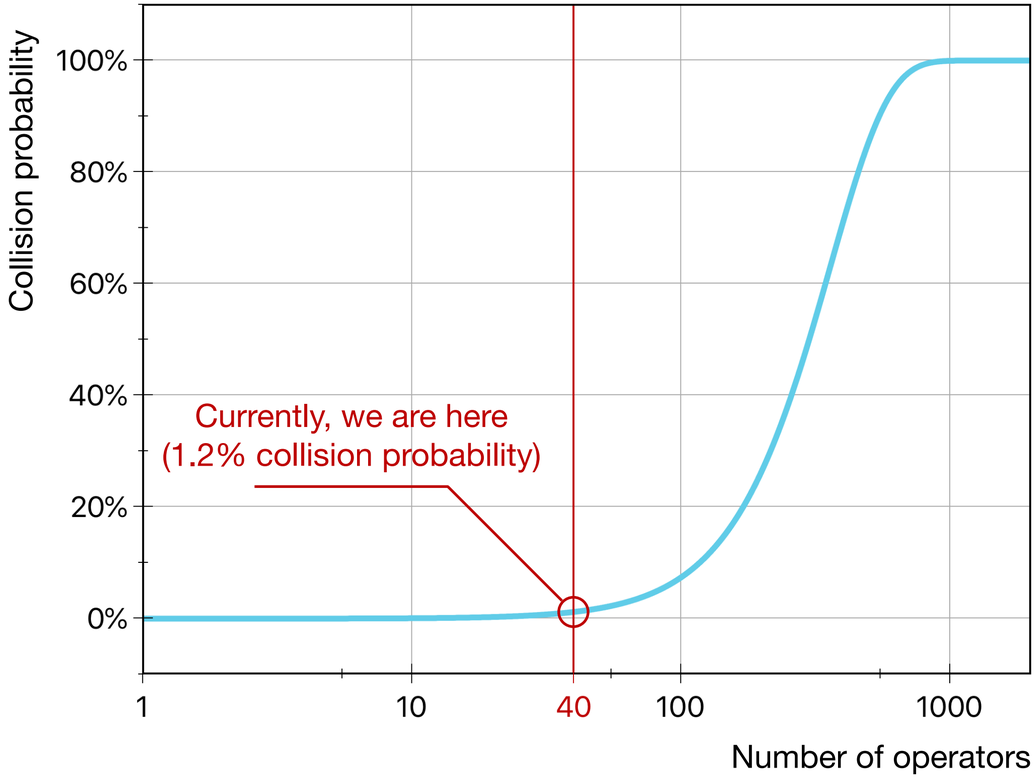


$$\mu = \mathrm{MD5}\left(\sum_{i=1}^{m} \sigma(\Delta_i)\right)$$

Collisions of Semantic Hash



Collision Danger:
2-Bytes Hash



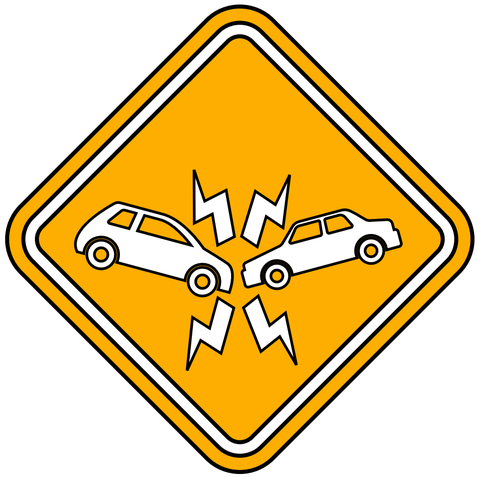
$$\mu = \mathrm{MD5}\left(\sum_{i=1}^{m} \sigma(\Delta_i)\right)$$

number of operators in the embedded reasoner

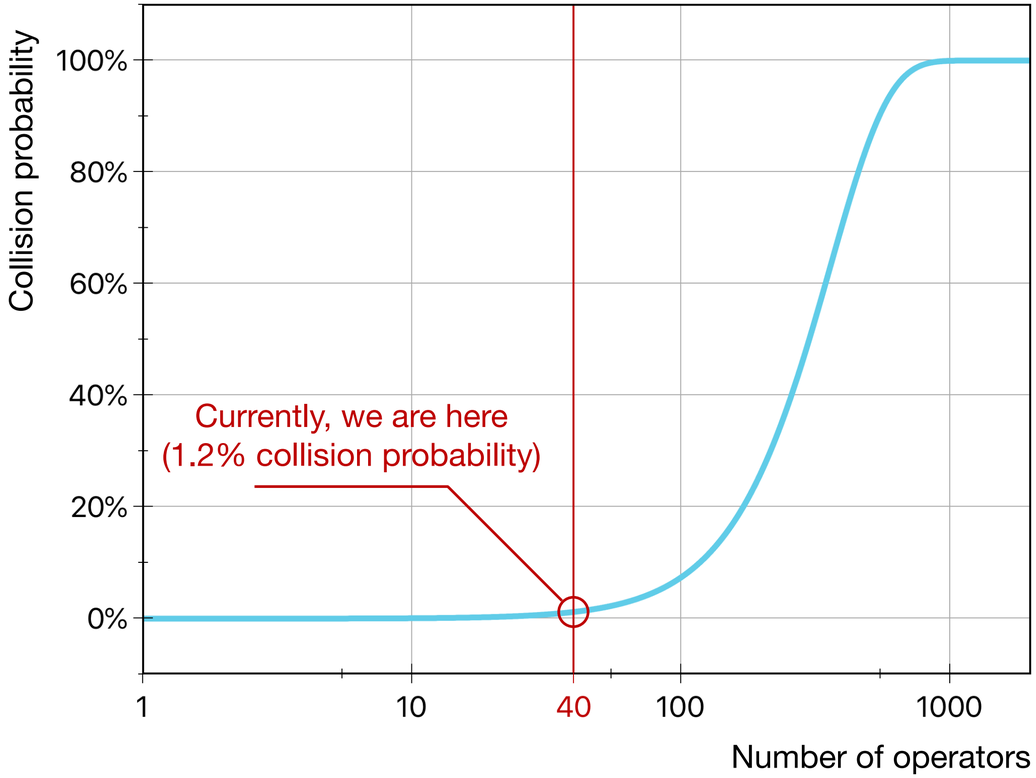
signature of operator

";"-delimitered string concatenation

Collisions of Semantic Hash



Collision Danger:
2-Bytes Hash



$$\mu = \mathrm{MD5}\left(\sum_{i=1}^{m} \sigma(\Delta_i)\right)$$

number of operators in the embedded reasoner

signature of operator

";"-delimitered string concatenation

This MD5 hash is stored in the embedded reasoner (taking 16 bytes)

Compatibility Check

Compatibility Check

1. Reasoner sends its operators' lookup table
   (set of Pearson-hased signatures of operators)
   and MD5 hash

Compatibility Check

1. Reasoner sends its operators' lookup table
   (set of Pearson-hased signatures of operators)
   and MD5 hash

2. SciVi server searches operators with corresponding hashes
   in the application ontology

Compatibility Check

1. Reasoner sends its operators' lookup table
   (set of Pearson-hased signatures of operators)
   and MD5 hash

2. SciVi server searches operators with corresponding hashes
   in the application ontology

3. If at least one operator has no correspondence, compatibility check is failed

Compatibility Check

1. Reasoner sends its operators' lookup table
   (set of Pearson-hased signatures of operators)
   and MD5 hash

2. SciVi server searches operators with corresponding hashes
   in the application ontology

3. If at least one operator has no correspondence, compatibility check is failed

4. Else, SciVi server reconstructs MD5 hash by the application ontology
   and compares it with the one received from reasoner

Compatibility Check

1. Reasoner sends its operators' lookup table
   (set of Pearson-hased signatures of operators)
   and MD5 hash

2. SciVi server searches operators with corresponding hashes
   in the application ontology

3. If at least one operator has no correspondence, compatibility check is failed

4. Else, SciVi server reconstructs MD5 hash by the application ontology
   and compares it with the one received from reasoner

5. If MD5 hashes do not match, compatibility check is failed

Compatibility Check

1. Reasoner sends its operators' lookup table
   (set of Pearson-hased signatures of operators)
   and MD5 hash

2. SciVi server searches operators with corresponding hashes
   in the application ontology

3. If at least one operator has no correspondence, compatibility check is failed

4. Else, SciVi server reconstructs MD5 hash by the application ontology
   and compares it with the one received from reasoner

5. If MD5 hashes do not match, compatibility check is failed

6. Else, compatibility check is passed

Performance Comparison

Performance Comparison

Testing environment:

Performance Comparison

Testing environment:

- SciVi server: **MacBook Pro 2.3 GHz 8-Core Intel Core i9 CPU, 16 Gb RAM**

Performance Comparison

Testing environment:

- SciVi server: **MacBook Pro 2.3 GHz 8-Core Intel Core i9 CPU, 16 Gb RAM**
- Application ontology: **328 nodes and 845 relationships**

Performance Comparison

Testing environment:

- SciVi server: **MacBook Pro 2.3 GHz 8-Core Intel Core i9 CPU, 16 Gb RAM**
- Application ontology: **328 nodes and 845 relationships**
- Edge device: **custom ESP8266-based controller for VR scene**

Performance Comparison

Testing environment:

- SciVi server: **MacBook Pro 2.3 GHz 8-Core Intel Core i9 CPU, 16 Gb RAM**
- Application ontology: **328 nodes and 845 relationships**
- Edge device: **custom ESP8266-based controller for VR scene**

Testing results:

Performance Comparison

Testing environment:

- SciVi server: **MacBook Pro 2.3 GHz 8-Core Intel Core i9 CPU, 16 Gb RAM**
- Application ontology: **328 nodes and 845 relationships**
- Edge device: **custom ESP8266-based controller for VR scene**

Testing results:

- Semantic hash calculation time: **2.15 ms / operator (average)**

Performance Comparison

Testing environment:

- SciVi server: **MacBook Pro 2.3 GHz 8-Core Intel Core i9 CPU, 16 Gb RAM**
- Application ontology: **328 nodes and 845 relationships**
- Edge device: **custom ESP8266-based controller for VR scene**

Testing results:

- Semantic hash calculation time: **2.15 ms / operator (average)**
- Firmware memory footprint: **16 bytes**

Performance Comparison

Testing environment:

- SciVi server: **MacBook Pro 2.3 GHz 8-Core Intel Core i9 CPU, 16 Gb RAM**
- Application ontology: **328 nodes and 845 relationships**
- Edge device: **custom ESP8266-based controller for VR scene**

Testing results:

- Semantic hash calculation time: **2.15 ms / operator (average)**
- Firmware memory footprint: **16 bytes**
- Device behavior updating time:

Performance Comparison

Testing environment:

- SciVi server: **MacBook Pro 2.3 GHz 8-Core Intel Core i9 CPU, 16 Gb RAM**
- Application ontology: **328 nodes and 845 relationships**
- Edge device: **custom ESP8266-based controller for VR scene**

Testing results:

- Semantic hash calculation time: **2.15 ms / operator (average)**
- Firmware memory footprint: **16 bytes**
- Device behavior updating time:

Development case: type of changes in ontology	No changes	Changes of related operators' structure	Changes of related operators' parameters naming	Changes of unrelated operators	Average
Conventional versioning	16 ms	30000 ms	30000 ms	30000 ms	22504 ms
Semantic hashing	16 ms	30000 ms	16 ms	16 ms	7512 ms

Performance Comparison

Testing environment:

- SciVi server: **MacBook Pro 2.3 GHz 8-Core Intel Core i9 CPU, 16 Gb RAM**
- Application ontology: **328 nodes and 845 relationships**
- Edge device: **custom ESP8266-based controller for VR scene**

Testing results:

- Semantic hash calculation time: **2.15 ms / operator (average)**
- Firmware memory footprint: **16 bytes**
- Device behavior updating time:

Development case: type of changes in ontology	No changes	Changes of related operators' structure	Changes of related operators' parameters naming	Changes of unrelated operators	Average
Conventional versioning	16 ms	30000 ms	30000 ms	30000 ms	22504 ms
Semantic hashing	16 ms	30000 ms	16 ms	16 ms	7512 ms

3 times faster

Conclusion

Result:

New level of ODEC maturity by mitigating the compatibility uncertainty with semantic hashing:

1. **Average performance boost: x3**
2. **Memory footprint: 16 bytes per firmware**
3. **Implementation available on GitHub: https://github.com/scivi-tools/**

Future plan:

Further development of ODEC by creating an ontology-driven bus for joining hardware components of edge devices on plug-and-play principles

1 St Petersburg University

2 Perm Sate University

3 UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Thank you for attention!

This study is supported by the research grant No. ID92566385 from Saint Petersburg University, "Text processing in L1 and L2: Experimental study with eye-tracking, visual analytics and virtual reality technologies"

Konstantin Ryabinin [1,2,3], kostya.ryabinin@gmail.com

Svetlana Chuprina [2], chuprinas@inbox.ru

Prague – 2023