Functional Component Descriptions for Electrical Circuits based on Semantic Technology Reasoning

Johannes Bayer[a], Mina Karami Zadeh[b], Markus Schröder[c] and Andreas Dengel[d]
Deutsches Forschungszentrum für Künstliche Intelligenz, Trippstadter Str. 122, Kaiserslautern, Germany

Keywords: RDF, Forward Chaining, Electrical Network, Circuit Diagram.

Abstract: Circuit diagrams have been used in electrical engineering for decades to describe the wiring of devices and facilities. They depict electrical components in a symbolic and graph-based manner. While the circuit design is usually performed electronically, there are still legacy paper-based diagrams that require digitization in order to be used in CAE systems. Generally, knowledge on specific circuits may be lost between engineering projects, making it hard for domain novices to understand a given circuit design. The graph-based nature of these documents can be exploited by semantic technology-based reasoning in order to generate human-understandable descriptions of their functional principles. More precisely, each electrical component (e.g. a `diode`) of a circuit may be assigned a high-level function label which describes its purpose within the device (e.g. `flyback diode for reverse voltage protection`). In this paper, forward chaining rules are used for such a generation. The described approach is applicable for both CAE-based circuits as well as raw circuits yielded by an image understanding pipeline. The viability of the approach is demonstrated by application to an existing set of circuits.

1 INTRODUCTION

Graphs are a traditional mean to describe electrical circuits (Rücker, 2012). Computer-aided engineering (CAE) systems are already used to capture, maintain, simulate and verify these circuits by exploiting their underlying graph structure. However, circuits also incorporate engineering knowledge. During the migration of circuits between CAE systems or during the digitization of circuits from paper sources, maintaining the plain syntactic features of the graph structure is often focused while high-level functional principles are not properly processed. Likewise, errors in the migration process often need to be traced manually.

In order to help developers (and other agents) understand circuit functionality and to automatically search for engineering concepts, additional means are required. This can also be achieved by exploiting graph structures: For example, a diode which has its anode connected to a terminal of an inductor and its cathode connected to the opposite terminal of the same inductor can be considered a flyback diode (functional description of the diode component). Likewise, a direct electrical connection between the two terminals of a voltage source can be considered a shortcut (fault description). Modeling these engineering concepts by the semantic technologies in order to reason about their presence in arbitrary circuits is the objective of the paper at hand.

[a] https://orcid.org/0000-0002-0728-8735
[b] https://orcid.org/0000-0002-6965-5190
[c] https://orcid.org/0000-0001-8416-0535
[d] https://orcid.org/0000-0002-6100-8255

2 RELATED WORK

Electrical Rule checker are already existing for CAE software (e.g. (Beard, 2021)).

Efforts have already been made to model electrical power systems for the purpose of interoperability (Gaha et al., 2013) as well as fault diagnosis (Bernaras et al., 1996). In contrast, the paper at hand focuses on electronic circuits.

(Liu and Farley, 1990) describe a system that integrates physical aspects and statements about circuits, hence allow to question about the low-level behavior of the system. Conversely, (Kitamura and Mizoguchi, 1998) describe an approach for assigning functions to components of technical systems. However, the approach is rather generic way and demonstrated by the

Bayer, J., Zadeh, M., Schröder, M. and Dengel, A.
Functional Component Descriptions for Electrical Circuits based on Semantic Technology Reasoning.
DOI: 10.5220/0011322000003269
In *Proceedings of the 11th International Conference on Data Science, Technology and Applications (DATA 2022)*, pages 528-532
ISBN: 978-989-758-583-8; ISSN: 2184-285X

example of a power plant.

(Kleer, 1979) (De Kleer, 1984) describes a comprehensive system for automatically deriving high-level insights on electronic circuits. Due to the time of their implementation, the described systems lack support of modern semantic technologies like RDF (rdf, 2014).

(Kunal et al., 2020) describe an approach for sub-circuit annotation to based on graph neural networks. While this approach is powerful, it comes with the typical limitation of ANNs like a fixed class set the requirement for a (large and usually non-public) dataset as well as a lack of perfect accuracy and explainability.

Systems for the manipulation of piping and instrumentation diagrams (which are structurally similar to circuit diagrams) have been proposed by (Grüner et al., 2014) and (Bayer and Sinha, 2020).

An extended version of a public dataset of hand-drawn circuit diagrams (Thoma et al., 2021) has been used to evaluate the approach described in this paper.

Wikidata is an open knowledge database, in which entities of many domains as well as general-purpose properties are available(van Veen, 2019). As it contains entries for describing electric and electronic components, the RDF circuit representations described in this paper is linked against this database in order to allow for later inference of additional knowledge.

3 APPROACH

The complete approach consists of the following steps (see Figure 1):

- Conversion from CAE file formats or recognition results to a uniform raw RDF representation.
- **Preprocessing Rules** are applied in order to abstract from the circuit's optical features like junctions and to compensate structural weaknesses like directed component connections.
- **Component Annotation Rules** are applied to generate functional descriptions within the circuit.
- For avoiding redundancies, the resulting RDF representation is generated from a unification of the raw RDF and the functional descriptions.
- The enriched RDF is converted back to CAE or image files

Both preprocessing rules and component annotation rules are denoted in Apache Jena (Siemer, 2019) forward chaining syntax.

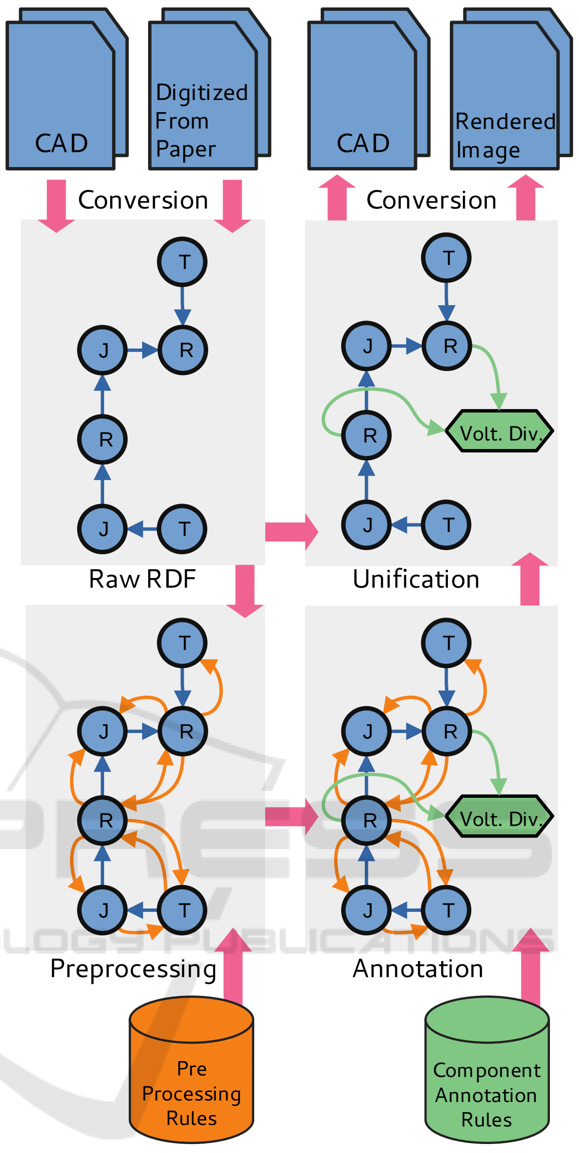

Figure 1: System Overview.

3.1 Ontology

In oder to have a common ontology for describing the circuits, the following terminology is used:

A **circuit** is described as a graph structure with **components** as nodes and electrical **connections** as edges. Components can be either connected directly or via **ports** which represent the individual terminals of a component (e.g. the anode of a diode). Additionally, **junctions** and **crossover** symbols are introduced to support the graphical representation of engineering diagrams as well as their digitization from analogue sources. Junctions are used both to indicate corners in wiring lines and to allow for connecting multiple components (to form hyperedges in

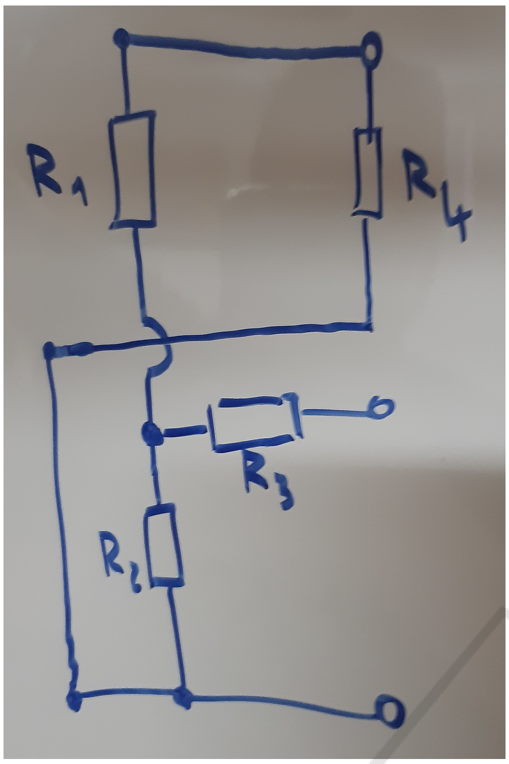

Figure 2: Sample Circuit Image.

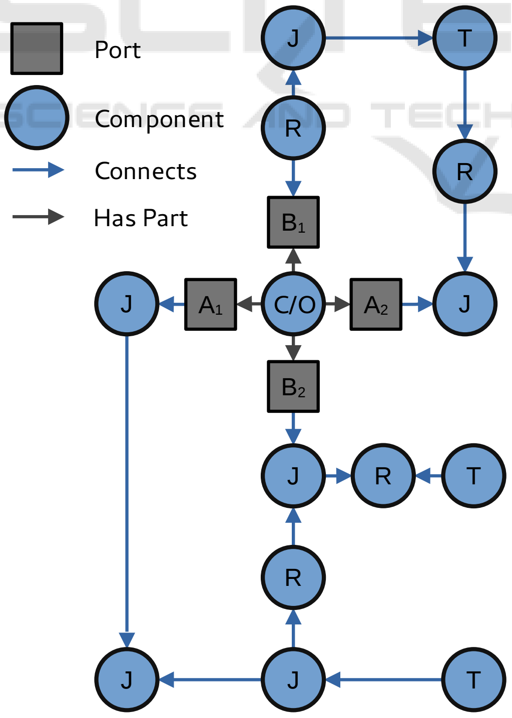

Figure 3: Sample Circuit in Proposed Representation. For visibility reasons, relations to the owing graph resource and the positioning are omitted; port names as well as component types are denoted as abbreviated labels. Note that the direction of component connections is unordered.

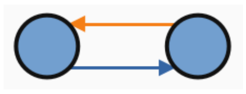

Figure 4: Electrical Symmetry Rule (result in orange).

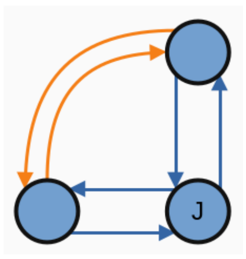

Figure 5: Junction Resolution Rule (results in orange; after prior symmetry rule application).

graph logic). Crossover symbols indicate the crossing of lines without a electrical connection between them. Components annotation rules add **functions** to indicate a component's purpose in the circuit. All mentioned terms (including the specific **compoment classes** and **function classes**) are related to Wikidata (van Veen, 2019) entries in the circuit's RDF representation.

3.2 Preprocessing Rules

The preprocessing rules create a normalized electrical view by augmenting the raw RDF structure. Therefore, they allow a simplified and flexible design of the annotation rules:

3.2.1 Electrical Symmetry

The RDF triples which express the electrical connections between the components form a directed graph, while there is no physical justification for the level at which the annotation rules are applied. In fact, the position of subject and object resource is considered not well defined in the respective triples (i.e. up to the implementation of the graph generation), resulting in an directed yet unordered relation. In order to implement an undirected graph structure and consequently allow for an abstraction before the annotations rule application, connecting triples of opposite direction are added (see also Figure 4):

```
[electSymm: (?a w:connects ?b)
          -> (?b w:connects ?a)]
```

3.2.2 Junction Resolution

As junctions are both optical features and a mean to describe hyperedges (which are tedious to encode in forward chaining rules), they also need to be broken down to direct connections between components.

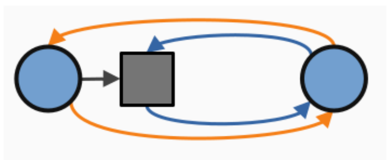

Figure 6: Port Resultion Rule (results in orange; after symmetry rule application).

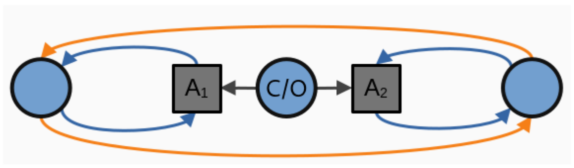

Figure 7: Crossover Resolution Rule (results in orange; after prior symmetry rule application).

This can be achieved by a simple transitive relation (see Figure 5):

```
[byJ:  (?a w:connects ?junction),
       (?junction w:connects ?c),
       (?junction rdf:type w:JUNCTION)
      -> (?a w:connects ?c)]
```

3.2.3 Port Resolution

As information model allows electrical connections between either **components** or **ports** or a mixture of them, rules may or may not address them. As circuits of different granularity should be supported and some rules don't make use of the port attributes, it is crucial to add connections so that components are always connected directly. Note that this rule is used in conjunction with the electrical symmetry (see Figure 6):

```
[res:  (?owner w:has_part ?port),
       (?port rdf:type w:PORT),
       (?a w:connects ?port)
      -> (?a w:connects ?owner)]
```

3.2.4 Crossover Resolution

Crossover symbols are resolved by connecting the connection partners of the opposite crossover symbol ports (the rule below only describes the resolution of one pair of opposite crossover ports while a crossover symbol usually has two pairs, see Figure 7):

```
[resCro:  (?a w:connects ?co_a1),
          (?b w:connects ?co_a2),
          (?co rdf:type w:CROSSOVER),
          (?co_a1 w:name a_1),
          (?co_a2 w:name a_2),
          (?co_a1 rdf:type w:PORT),
          (?co_a2 rdf:type w:PORT),
          (?co w:has_part ?co_a1),
          (?co w:has_part ?co_a2),
          (?junction rdf:type w:JUNCTION)
         -> (?a w:connects ?b)]
```

3.3 Component Annotation Rules

While the preprocessing rules are considered to be a fixed set, the component annotation rules are an intended to be extensible by domain experts and knowledge workers. The component annotation rules used in this paper are:

Name	Description
Emitter Common Amplifier	A bipolar transistor amplifier using a voltage divider biasing to keep base bias voltage at a constant level.
Coupling Capacitor	Connects the AC part of a signal between two parts of the circuit while blocking DC Parts.
Electronic Switch	Component in either open or closed state.
Flyback Diode	A diode that is connected inversely to an energy storage component for protecting against voltage spikes.
Oscillator Crystal	Provide a constant stable frequency and can be used as clock in digital circuits
PullUp Resistor	Provides a well-defined voltage level in case of absence of other connections (e.g. open switches).
Voltage Divider	provides an intermediate voltage level between the surrounding voltage levels.

3.4 Implementation

Complete circuits diagrams are loaded from KiCad (Kanagachidambaresan, 2021) schematic files and internally captured as NetworkX (Hagberg and Conway, 2020) graphs before being converted to RDF Turtle (World Wide Web Consortium, 2014) representations. The preprocessing as well as the component annotation itself is performed as forward chaining rules in Apache Jena (Siemer, 2019), where the component annotation rules are implemented as individual files for extensibility purposes. The source code and the circuit dataset is made publicly available[1].

[1]https://github.com/DFKI/circuitgraph-insights

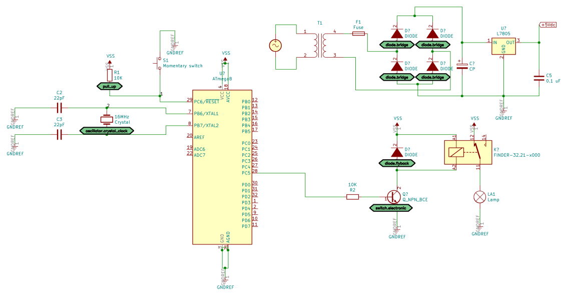

Figure 8: Component Annotations (Green Hexagons) on a Sample Circuit.

4 EVALUATION

In order to validate the approach, the results are demonstrated on a sample circuit (see Figure 8).

5 CONCLUSION

An RDF-based system for automatically deriving functional annotations of individual components inside circuits has been described. By incorporating support for an openly available CAE system as well as referencing ressources from the also openly available wikidata knowledge base, it connects the world of circuit modeling with the world of image annotation and the world of circuit understanding.

6 OUTLOOK

So far, many of the rules needed to be formulated in multiple, rather specific ways in order to deal with all desired situations. Further preprocessing steps are required to allow for more general formulations. For example, voltage sources as well as vcc and gnd symbols need to be resolved to a uniform representation.

ACKNOWLEDGMENT

This work was funded by the BMBF project SensAI (grant no. 01IW20007).

REFERENCES

(2014). Resource description framework (rdf). https://www.w3.org/RDF/. [Online; accessed 14-April-2022].

Bayer, J. and Sinha, A. (2020). Graph-based manipulation rules for piping and instrumentation diagrams.

Beard, J. (2021). KiBot (formerly KiPlot). https://github.com/INTI-CMNB/KiBot. [Online; accessed 4-April-2022].

Bernaras, A., Laresgoiti, I., Bartolome, N., and Corera, J. (1996). An ontology for fault diagnosis in electrical networks. In *Proceedings of International Conference on Intelligent System Application to Power Systems*, pages 199–203.

De Kleer, J. (1984). How circuits work. *Artificial intelligence*, 24(1-3):205–280.

Gaha, M., Zinflou, A., Langheit, C., Bouffard, A., Viau, M., and Vouligny, L. (2013). An ontology-based reasoning approach for electric power utilities. In *International Conference on Web Reasoning and Rule Systems*, pages 95–108. Springer.

Grüner, S., Weber, P., and Epple, U. (2014). Rule-based engineering using declarative graph database queries. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pages 274–279. IEEE.

Hagberg, A. and Conway, D. (2020). Networkx: Network analysis with python.

Kanagachidambaresan, G. (2021). Introduction to kicad design for breakout and circuit designs. In *Role of Single Board Computers (SBCs) in rapid IoT Prototyping*, pages 165–175. Springer.

Kitamura, Y. and Mizoguchi, R. (1998). Functional ontology for functional understanding. In *Twelfth International Workshop on Qualitative Reasoning (QR-98), Cape Cod, USA, AAAI Press*, pages 77–87.

Kleer, J. D. (1979). Causal and teleological reasoning in circuit recognition.

Kunal, K., Dhar, T., Madhusudan, M., Poojary, J., Sharma, A., Xu, W., Burns, S. M., Hu, J., Harjani, R., and Sapatnekar, S. S. (2020). Gana: Graph convolutional network based automated netlist annotation for analog circuits. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 55–60. IEEE.

Liu, Z.-Y. and Farley, A. M. (1990). Shifting ontological perspectives in reasoning about physical systems. In *AAAI*, pages 395–400.

Rücker, G. (2012). Network meta-analysis, electrical networks and graph theory. *Research synthesis methods*, 3(4):312–324.

Siemer, S. (2019). Exploring the apache jena framework.

Thoma, F., Bayer, J., Li, Y., and Dengel, A. (2021). A public ground-truth dataset for handwritten circuit diagram images. In *International Conference on Document Analysis and Recognition*, pages 20–27. Springer.

van Veen, T. (2019). Wikidata. *Information technology and libraries*, 38(2):72–81.

World Wide Web Consortium (2014). RDF 1.1 Turtle.