Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Formal Definition of the Term "Semantics" as a Foundation for Semantic Interoperability in the Industrial Internet of Things

Tizian Schröder * Christian Diedrich *

**Institute for Automation Engineering, Otto-von-Guericke-University, Universitätsplatz 2, Magdeburg, Germany (e-mail :{tizian.schroeder, christian.diedrich}@ovgu.de).*

Abstract: Semantic interoperability is seen as the key to realize the ideas of the Industrial Internet of Things (IIoT). In order to equip technical systems with such a capability, a precise definition of the term "semantics" is needed. Complex IIoT devices can only be developed properly on a formal foundation. Existing approaches that intend to specify the term "semantics" are often more intuitively motivated. These include, for example, the knowledge pyramid or the Levels of Conceptual Interoperability Model (LCIM). The paper provides a formal definition of the term "semantics" and relates these existing approaches critically to the proposed definition.

Keywords: Semantics, Semantic Interoperability, Network of Interacting Systems, Industrial Internet of Things (IIoT), Knowledge Pyramid, Levels of Conceptual Interoperability Model (LCIM), Ontologies.

1. INTRODUCTION

The vision of the Industrial Internet of Things (IIoT) envisages that a large number of decentralized systems form a decentralized network of systems in which autonomous, intelligent entities act for the purpose of achieving their individual goals. The most basic motivation of this effort can be seen in making the enormous complexity of networks of systems, which is already enormous today and will potentially continue to grow in the future, more manageable by moving from a centralized approach based on the automation pyramid to a decentralized approach. A proper interaction of information processing systems requires semantic interoperability as a crucial prerequisite for the implementation of the Industrial Internet of Things. Following the idea of the cyber-physical (production) systems (CP(P)S), these "Things" as physical systems are extended by a digital representation in order to integrate the physical world into the information world (digitization). This digital representation is an information processing system, in the following only called "system" because the focus lies on these information processing entities. Fig. 1 illustrates the intended purpose of such an (information processing) system.

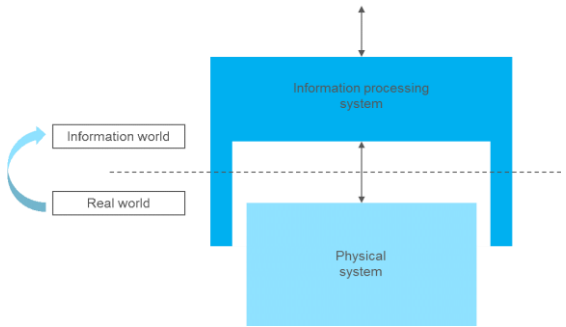

Fig. 1. Digitization by adding information processing system to physical system.

Fig. 2 schematically shows a simple network of systems in which individual systems interact as decentralized entities.

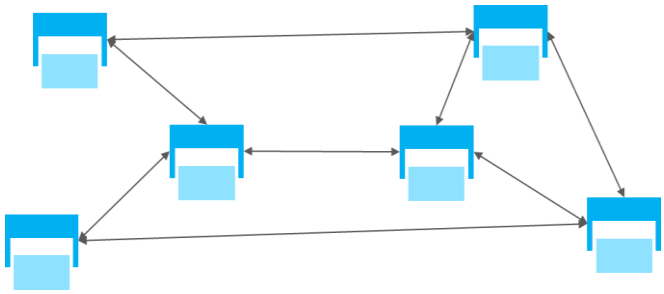

Fig. 2. Network Of interacting systems

The paper provides a formal definition of the term "semantics" which helps to further develop approaches for the semantic interoperability within those networks of systems. At the level of understanding, at least as far as terminology is concerned, this view of "semantics" is currently not established according to our findings. We regard these fundamental considerations as essential in order to solve further problems in practice-oriented research activities in the future. It is not yet about their application to concrete problems, but about a proposal for a formally grounded perspective on the problem of "semantics" and the critical examination of the currently pursued approaches of the knowledge pyramid or the Levels of Conceptual Interoperability Model (LCIM) as well as ontologies. The paper takes a fundamental view and serves primarily to stimulate a scientific discourse on a precise definition of the term "semantics".

2. FUNDAMENTALS AND CRITICISM

The semantics of data plays an important role in many efforts to advance digitization (PI4.0, 2016; PI4.0 2018). Everyone has an intuitive idea of what the meaning of semantics is. This intuitive notion of semantics has produced a number of models

that intend to refine the concept of semantics. These include, for example, the knowledge pyramid or the Levels of Conceptual Interoperability Model (LCIM) (Turnitsa, 2005). However, the implications of these models are sometimes rather weak or, on closer inspection, erroneous. The knowledge pyramid is presented below as a representative of a class of similar models. These models share the claim to provide a definition for semantics and related terms. In addition, further concepts related to formal semantics are presented and related to the definition proposed here.

2.1 Knowledge Pyramid

The transition from data up to knowledge and the actions that can be derived from it is vividly illustrated in the knowledge pyramid after Aamodt and Nygard (Aamodt and Nygård, 1995) (Fig. 3). The lowest level (level 1) represents the set of symbols from which data can be composed according to a - possibly formal - syntax (level 2). This corresponds to the definition of formal languages as a set of words formed from an alphabet as a set of permitted symbols. As soon as data receives a semantics, i.e. its meaning is determined, the transition to information takes place (level 3). According to this model, the crosslinking of several pieces of information results in knowledge (level 4). This crosslinking is also seen as an addition of pragmatics. On this basis, actions can be derived – in other words: decisions can be made (level 5). The transitions between the levels therefore take place according to the triad syntax, semantics and pragmatics, which are also represented in the so-called semiotic triangle.

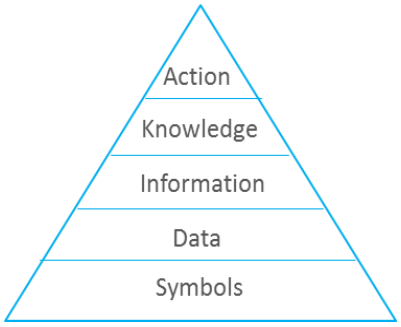

Fig. 3. knowledge Pyramid.

By assigning semantics to data, information is created. Knowledge results from the crosslinking of information. We see actions as information or knowledge in combination with an objective, which is represented as information or knowledge too. The transition from information to knowledge therefore takes place through crosslinking. This is vague, because depending on the perspective, what is regarded as elementary information from one perspective may well contain a substructure - i.e. a crosslinking of information - from another perspective. The transition from knowledge to action takes place by assigning an objective, or the intention, which determines which external effect must unfold in order to fulfil a certain purpose. This objective exists itself in the form of information or knowledge and therefore does not form a category of its own. These explanations support our view that the distinction made by the knowledge pyramid between information, knowledge and action is a phenomenological distinction, i.e. one that cannot be clearly identified from a formal perspective. If the principles of the knowledge pyramid are to be used for the realization of semantic interoperability - i.e. including the actual implementation and not only rough conceptual ideas - these must be sufficiently formalized.

2.2 Ontologies

Another widespread answer in the search for the definition of semantics is the concept of ontologies as one of several approaches for knowledge representation(Brachman and Levesque, 2004; Hildebrandt et al., 2017) Before knowledge can be processed by a system, it must be made available to it in a form of representation. For such forms of representation, a wide variety of approaches exist that differ in their expressiveness, e.g. glossaries, taxonomies, classifications or semantic networks/ontologies. The result of the application of such approaches are models that represent knowledge of one or more domains. Concepts and relations between these form the building blocks of an ontology which represent the vocabulary of one or more domains. A much quoted definition of the term comes from Rudi Studer, who - referring to definitions by Gruber and Borst - regards an ontology as a "formal, explicit specification of a shared conceptualization" (Studer, Benjamins and Fensel, 1998). Ontologies are described e.g. with RDF with tuples of the form <subject> <predicate> <object>. In RDF classes (concept), instances (individual), relations (object property) or literals (datatype property) can be represented. In this way, a graph is created whose nodes and edges can be uniquely identified, e.g. via a URI. The (automated) tracking of the edges makes it possible to browse the shared domain vocabulary. A simple example is shown in Fig. 4.

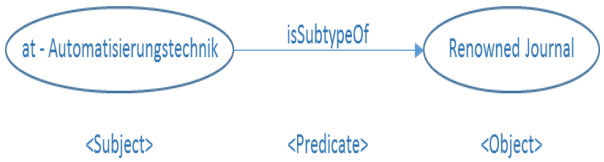

Fig. 4. Example of a simple ontology.

The problem is that ontologies, as representatives of approaches for knowledge representation, suggest that in order to make semantics of data machine-processable, they must be “described”. This means that semantics can be described and thus processed. This leads to the following circular conclusion: If the semantics of data is itself described by other data, who or what exactly describes the semantics of this description? In order to answer the question of what semantics of a term is, ontologies answer with a substitute description of this term. This is always continued without ever answering what “semantics” really means. Ontologies are therefore not an answer to the question “What is semantics?”, but merely a possibility to replace the semantics of one term with the semantics of other terms. However, we do not want to make replacements, but define the term of “semantics” itself.

2.3 Semantics of Programs

Theoretical computer science defines the concept of formal semantics. Its purpose is to make the meaning of programs specified using formal languages formally describable.

In general, a semantic function is constructed as a mapping of a program to the function calculated by this program (Stump, 2013):

$$C: P \rightarrow f$$

C denotes the semantic function, P the set of syntactically correct programs, $f: \Sigma \rightarrow \Sigma$ the function calculated by the program and Σ the set of possible memory allocations.

There are several such formal semantics which differ in the mathematical representation of this semantic function, such as axiomatic semantics, denotational semantics and operational semantics (Stump, 2013; Nielson and Nielson, 2007).

This view of the semantics of programs established in theoretical computer science is similar to the definition of semantics given in this paper. Both definitions explain semantics through an effect realized by a function associated with semantic-related entities.

However, there is a difference between these entities and their association with this function. On the one hand, the formal semantics of programs does not address the meaning of atomic characters, but the meaning of a program resulting from the composition of several characters. On the other hand, the semantics of the program is represented directly by this function. In the definition given in this paper, however, this function is the system function of any system that accepts a character as a function argument and assigns it the calculable function value as a representation of its semantics. Here the semantics is represented by a single function value instead of by a whole function. In addition, this paper also considers the context dependency of the assignment of semantics to data, which is not the case with the aforementioned concepts.

2.4 Semantics of Data Types

In computer science, a data type is the combination of sets of permitted values for instances of that data type and the operations applicable to that data (Dale, 1996). It is defined by a signature. Such a signature initially defines only the names of the value sets and operations. If, in addition, semantics is specified by defining the interpretation of these value sets and operations, a data type in a more specific sense results (Noble et al., 2018).

Obviously, the term semantics also plays a role here. However, this refers to the meaning of a whole class of characters. This class of characters is defined by the semantics of the value sets. Its semantics is then specified for the entire class of these characters by the semantics of the operations applicable to it.

Therefore, the concept for defining the semantics of data types serves a different purpose despite fundamental similarities to the definition of semantics given in this paper. Instead of defining a definition for the semantics of individual characters, i.e. concrete instances of a data type, the semantics of an entire class of such characters is defined.

The similarity is that in both concepts the functions or operations applied to the characters or classes of characters are relevant. In the definition for semantics given here, these are system functions of data processing systems. In order to define the semantics of individual characters, the concrete application of such a function to characters and the resulting calculated function value must be considered instead of merely the general applicability of an operation to the corresponding data type. In addition, when considering the semantics of data types, unlike in this paper, no context dependency of the assignment of semantics is considered.

2.5 Relation to OSI Reference Model

The semantics of data is also relevant in the context of the OSI reference model (ISO/IEC, 1994). The relationship to the definition of semantics given in this paper is that in each layer of the OSI Reference Model, the layer-specific portions of a datagram are processed by functions of an implemented protocol. These functions can often be specified as state machines and correspond to the system functions used in the definition. Obviously, because of its generality, the given definition is applicable to any system function, so it is not relevant which layer of the OSI reference model it belongs to or whether it is part of a communication protocol stack at all. However, the definition can also be used well to explain how communication protocols semantically interpret the header data of a datagram relevant to them. The user data of a datagram is ultimately interpreted in the same way by the application located above the communication stack.

2.6 Comparison of related concepts with proposed definition

Table 1 summarizes the aforementioned relations of selected concepts to the semantics definition proposed herein. Thereby key similarities and differences are highlighted.

Table 1 : Comparison of related concepts

Concept	Similarity	Difference
Knowledge Pyramid	Some layers can be mapped directly to the proposed definition (see chapter 4)	Some layers overlap in their meaning and thus cannot be distinguished (see chapter 4)
Ontologies	Intends to define semantics of arbitrarily complex terms/concepts	Replaces term semantics by other terms semantics without addressing its functional aspect
Semantics of Programs	Relates semantics to mathematical functions representing an entities behavior	Does not regard the context dependency of semantics
Semantics of Data Types	Relates semantics to mathematical functions applicable to (sets of) characters	Does not regard the context dependency of semantics
OSI Reference Model	Proposed definition is applicable to any layer of the OSI model	Relies implicitly on a similar semantics concept without explicitly defining it

3. PREREQUESITES FOR THE FORMAL DEFINITION OF THE TERM "SEMANTICS"

3.1 Systems

The concept of "system" is decisive in this work. Therefore, it should first be formally defined. The definition is a version of the definition of a system usual in the system theory adapted for the purposes pursued in this work (to be found e.g. in (Kilian, 2005)). In general all variables can be vectors.

Any system can be described as a tupel $Sys = (T, X, U, Y, x_0, f)$ with :

- Time Structure $T = (T, succ)$ with :

  Successor function $succ: T \rightarrow T,\ t' = succ(t)$

- State set X
- Input set U
- Output Set Y
- Initial state x_0
- System function $f = (f_{int}, f_{ext})$ with :
  - $f: X \times U \rightarrow X \times Y$ with :

    $$(x(t'), y(t')) = f(x(t), u(t))$$

  - $f_{int}: X \times U \rightarrow X$ with :

    $$x(t') = f_{int}(x(t), u(t))$$

  - $f_{ext}: X \times U \rightarrow Y$ with :

    $$y(t') = f_{ext}(x(t), u(t))$$

For the later definition of the term "semantics" it is introduced that the input/output vector is composed of the input/output components to be considered (cons) for the assignment of semantics to data and the remaining (rest) input/output components:

- $u = (u_{cons}, u_{rest})$
- $y = (y_{cons}, y_{rest})$

3.2 System Setup

Fig. 5 shows a minimal system architecture consisting of the system Sys_S (S for Self) and the environment interacting with it. It depends on the definition of the system boundaries whether this environment consists of one, several or all other systems participating in the network. For simplification, it is assumed that this is one arbitrarily complex environment system Sys_E (E for Environment). The notation for the description of the systems Sys_S and Sys_E corresponds to the general system description introduced above.

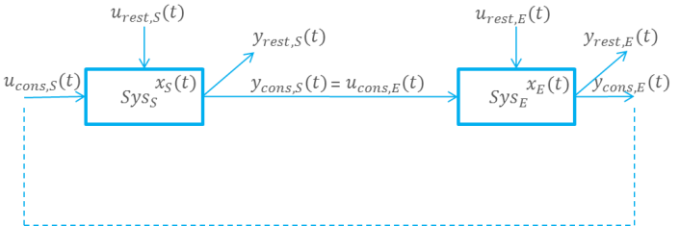

Fig. 5. Considered System and its Environment.

In the general case, the interaction behavior of a system is characterized by:

Asynchronicity: A system is not blocked in its overall behavior by one specific interaction.

Non-determinism: External systems (receivers) behave non-deterministically from the point of view of the system under consideration (senders).

State adhesion: The behavior of a system is stateful.

Of course, networks of systems are conceivable in which the individual systems present themselves differently with regard to their interaction behaviour. But the most complex networks and those enabling the most possible degree of decentralization or autonomy show the above mentioned interaction behaviour.

Here it becomes clear once again that with regard to the inputs/outputs a distinction is made between the currently considered ones ($u_{cons,S}$, $y_{cons,S}$, $u_{cons,E}$, $y_{cons,E}$) and the remaining ones ($u_{rest,S}$, $y_{rest,S}$, $u_{rest,E}$, $y_{rest,E}$). This distinction is important because the knowledge of Sys_S regarding all interactions of Sys_E Is always limited to the information represented by $u_{cons,E}$ and $y_{cons,E}$. Sys_S generally does not know the values of $u_{rest,E}$ and $y_{rest,E}$. Sys_S knows however $u_{rest,S}$ and $y_{rest,S}$, since it enters the corresponding interactions itself. Exactly from this fact results the already mentioned non-determinism of all interaction partners. Because the interactions unknown to Sys_S - those which Sys_E enters into with other external system participants - lead to the fact that the concrete behavior of Sys_S cannot be determined from the point of view of Sys_S. This distinction is important later on in this paper.

4. FORMAL DEFINITION OF THE TERM "SEMANTIC"

Fig. 6 shows the transition from the traditional "Knowledge Pyramid" to the new "Knowledge Flow" proposed here.

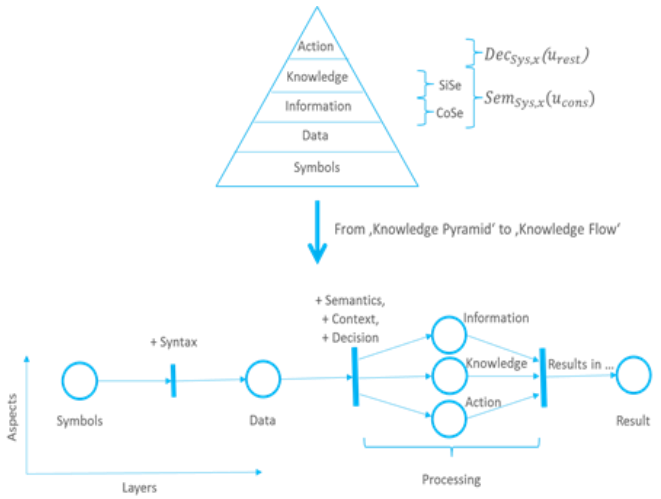

Fig. 6. From "knowledge Pyramid" to "Knowledge Flow".

An essential statement of this model is the recharacterization of information - knowledge - action, which are regarded as layers in the knowledge pyramid. It seems to be more appropriate to describe these components as equally important aspects. This no longer implies that these aspects are completely sequentially or hierarchically related to each other. The components of the knowledge flow, namely symbols, data, information, knowledge, action, are the same as in the knowledge pyramid. The sequential transition from symbols to data is also assumed by adding syntax in the sense of rules for the composition of data from elementary symbols. The subsequent transition from (unprocessed) data is also regarded as a delimitable sequential step. So far this is also in line with the LCIM up to and including the level "Syntactic Interoperability". After this transition a (core) semantics is assigned to the data in the context of the (data) processing (information), this is embedded in a context (knowledge) and a decision is derived from it (action), which ultimately leads to a result of the processing. However, it is generally not possible to sequence these three aspects of the processing. Instead, processing of data in general has these three aspects at the same time. In addition - in contrast to the aforementioned models - precise criteria for the separability of these aspects are to be given here in the following.

The following considerations are valid for any system Sys_S. Therefore, no distinction is made between Sys_S and Sys_E.

According to Fig. 5 u_{cons} is regarded as an input of Sys_S. As mentioned above, Sys_S generally receives further inputs, which are collectively referred to as u_{rest}. The proposed definition for the semantics Sem of u_{cons} from the point of view of Sys_S is:

$$\mathrm{Sem}_{\mathrm{Sys,x(t)}}(\mathrm{u(t)_{cons}}) = \left\{ (\mathrm{x(t')}, \mathrm{y(t')}) \middle| \begin{matrix} \mathrm{x(t')} = \mathrm{f_{int}}(\mathrm{x(t)}, (\mathrm{u(t)_{cons}}, \mathrm{u(t)_{rest}})), \\ \mathrm{y(t')} = \mathrm{f_{ext}}(\mathrm{x(t)}, (\mathrm{u(t)_{cons}}, \mathrm{u(t)_{rest}})), \\ \mathrm{u(t)_{rest}} \in \mathrm{U} \end{matrix} \right\} \quad (1)$$

This is a Sys_S -specific set - called Sem - that contains all the pairs $(x(t'), y(t'))$ that can result from f_{int} and f_{ext} with any u_{rest} and fixed $x(t)$. $x(t)$ is fixed, since the assignment of semantics to u_{cons} is to depend on the context, expressed by the state $x(t)$. u_{rest} is arbitrary, since the semantics of a certain input u_{cons} influences the result of the processing of this input without interdependence on the remaining interactions of Sys_S. t' denotes the next time step after t. With regard to the traditional knowledge pyramid, the semantic assignment thus defined covers the roles of the levels "information" and "knowledge". With regard to the LCIM, the semantic mapping thus defined covers the roles of the levels "Semantic Interoperability" and partly "Pragmatic Interoperability". We want to call the context-independent part of semantics "core semantics" (CoSe), the context-dependent part we call "side semantics" (SiSe). This distinction can be found in various other approaches: Linguistics distinguishes between denotation and connotation or semantics and pragmatics (Bussmann and Lauffer , 2008).

Ultimately, however, the system Sys_S should deliver one concrete result $(x(t'), y(t'))$ instead of a whole set of such results. Therefore, in addition the definition of the decision Dec of Sys_S as reaction to an input u_{cons} is made:

$$\mathrm{Dec}_{\mathrm{Sys,x(t)}}(\mathrm{u(t)_{rest}}) = \left\{ (\mathrm{x(t')}, \mathrm{y(t')}) \middle| \begin{matrix} \mathrm{x(t')} = \mathrm{f_{int}}(\mathrm{x(t)}, (\mathrm{u(t)_{cons}}, \mathrm{u(t)_{rest}})), \\ \mathrm{y(t')} = \mathrm{f_{ext}}(\mathrm{x(t)}, (\mathrm{u(t)_{cons}}, \mathrm{u(t)_{rest}})), \\ \mathrm{u(t)_{cons}} \in \mathrm{U} \end{matrix} \right\} \quad (2)$$

Similar to before, this is a Sys_S - specific set - called Dec - that contains all the pairs $(x(t'), y(t'))$ that can result from f_{int} and f_{ext} at any u_{cons} and fixed $x(t)$. $x(t)$ is fixed, because this decision should depend on the context, expressed by the state $x(t)$. u_{cons} is arbitrary, because the use of the freedom of decision regarding the reaction to a certain input u_{cons} depends only on the remaining interactions u_{rest} of Sys_S. t' denotes the next time step after t.With regard to the traditional knowledge pyramid, the decision making defined in this way is congruent with the role of the "action" level. With regard to the LCIM, the decision making thus defined covers the roles of all levels starting from "Pragmatic Interoperability". For the levels above, LCIM does not provide any clearly defined - i.e. as formal as possible - criteria for distinguishing them from each other.

The result $(x(t'), y(t'))$ realized by Sys_S in response to receiving the input u_{cons} is simply the intersection $(x(t'), y(t')) = Sem \cap Dec$. For deterministic system functions f, the cardinality of the set $Sem \cap Dec$ equals 1 $(x(t'), y(t'))$. This determinism has to be distinguished from the non-deterministic interaction behavior, which results from incomplete knowledge regarding the interaction partner. As already mentioned, nothing restrictive can be said about the order of the calculation of Sem and Dec, i.e. they are aspects, not layers.

This realized result corresponds exactly to the result of the system function: $Sem \cap Dec = f(x(t), (u(t)_{cons}, u(t)_{rest}))$. It is now possible to determine the system behavior by linking semantics and decision instead of using the system function.

This serves at the same time as validation of the given definitions for semantics and decision.

A simple equivalence criterion for determining the equality of semantics of two inputs u_1 and u_2 is the verification of the equation $Sem_{Sys,x}(u_1) = Sem_{Sys,x}(u_2)$. The equivalence relation is the set of all pairs of $u \in U$ that have the same meaning, thus: $ÄR \subseteq U \times U$. The equivalence class of an element $u_1 \in U$ is: $[u_1] = \{u \in U \mid u_1 \sim u\}$. But these equivalence considerations are not necessary to define the meaning of one input $\in U$.

It is evident that this definition is applicable to any system. Because the underlying system concept does not make any restrictions regarding the characterization of the system function. Thus, for example, it is irrelevant whether the system is a continuous system described by differential equations or a discrete system described by state machines.

The formal definitions will now be supported by a few more easily understandable statements: First of all, the idea is that each piece of data in itself is completely semanticless. Only when data is processed by the recipient the semantics is assigned, i.e. data gets that meaning that the recipient attaches to it. The semantics of an input from the recipient's point of view is therefore what "happens" $Sem_{Sys}(u_{cons}(t))$-related after the input has been received by the recipient. "Happen" means: transition to $(x(t'), y(t'))$, in other words: execution of f. The $Sem_{Sys}(u_{cons}(t))$-related part of $(x(t'), y(t'))$ aggregates the semantics assigning processing of the input. The semantics of an output from the sender's point of view, on the other hand, is the intended meaning, that is, the semantics that is assigned by the receiver according to the sender's expectation. Of course, this expectation does not have to apply. In this case, it is a misunderstanding in colloquial terms. This model is also easily transferable to human interaction: Sender and receiver exchange data. These data processing units are here 2 people. If input data arrives at the recipient, e.g. the reading of a character string, a processing occurs as a result. For example, the recipient imagines the real object designated by this character string visually, associates certain emotions with this idea and possibly derives from it a behavior that can be observed from outside.

5. EXAMPLE

Fig.7. shows an example network of interacting systems, based on Fig. 5. In addition, the u_{rest}/y_{rest} processing system Sys_{3rd} is shown in grey.

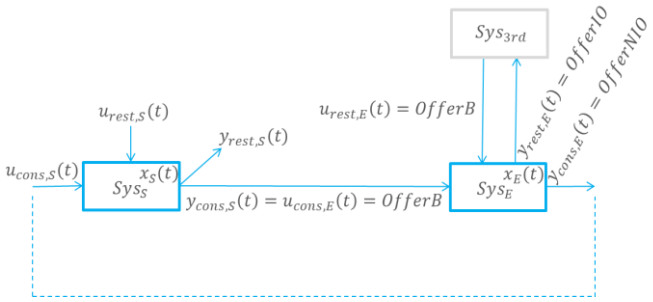

Fig. 7. Example of interacting systems.

In the example, the focus is on the semantic-assigning system with $Sys_E = (T, X, U, Y, x_0, f)$ with:

$$X = \{ready, busy\}$$

$$U = \{OfferA, OfferB\}$$

$$Y = \{OfferIO, OfferNIO\}$$

$$x_0 = ready$$

Table 2: System Function f_{Sys_E}.

$x_E(t)$	$u_{cons,E}(t)$	$u_{rest,E}(t)$	$y_{cons,E}(t)$	$y_{rest,E}(t)$	$x_E(t')$
ready	OfferA	OfferA	OfferNIO	OfferNIO	busy
ready	OfferA	OfferA	OfferNIO	OfferIO	busy
ready	OfferB	OfferB	OfferIO	OfferNIO	busy
ready	OfferB	OfferB	OfferNIO	OfferIO	busy

The time structure T is here regarded as arbitrary, because it has no relevance for the core statements of the example.

Sys_S and Sys_{3rd} can send the input characters $OfferA$ and $OfferB$, which can be processed by Sys_E and each represent a rather bad or rather good contract offer. Sys_E can respond with contract rejection or contract acceptance, which is communicated to the sender by the output characters $OfferNIO$ (for bad $OfferA$) or $OfferIO$ (for good $OfferB$). In the example the capacity of contract acceptance by Sys_E is limited to 1, thus Sys_{3rd} is given higher priority if both interaction partners Sys_S and Sys_{3rd} are offering $OfferB$, which is considered to be a good contract offer. In the concrete example Sys_S sends the character $u_{cons,E} = OfferB$ and Sys_{3rd} the character $u_{rest,E} = OfferB$. Sys_E responds to this with $y_{cons,E} = OfferNIO$ and $y_{rest,E} = OfferIO$ according to the system function f_{Sys_E}, since Sys_{3rd} is preferred for contract acceptance.

As an example, the semantics Sem of the character $y_{cons,S} = u_{cons,E} = OfferB$ sent by Sys_S is considered. According to the previously introduced formal definition of semantics, this corresponds to the set of all possible results after processing of this character by Sys_E with arbitrary influence by Sys_{3rd} :

$$Sem_{Sys_E,ready}(OfferB) =$$

$$\{(busy, (OfferIO, OfferNIO)),$$
$$(busy, (OfferNIO, OfferIO))\}$$

It should be noted that the reply from Sys_E to Sys_S depends on the contractual offer made by Sys_{3rd}, as Sys_{3rd} is considered to be a preferred contracting party.

Sys_E's decision freedom Dec regarding the decision concerning the offer made by Sys_S was defined as the amount of all processing results of $u_{rest,S} = OfferB$ under any influence by Sys_S :

$$Dec_{Sys_E,ready}(OfferB) = \{(busy, (OfferNIO, OfferIO))\}$$

The intersection of semantics Sem and decision Dec finally provides the concrete result of this interaction between Sys_S, Sys_E and Sys_{3rd} as specified in Fig. 7:

$$Sem \cap Dec = ((busy, (OfferNIO, OfferIO)))$$

Thus, the concrete system behaviour of Sys_E can be explained as a conjunction between the semantics assigned to the input characters and the decision taken thereon regarding the acceptance or rejection of the contract.

6. SUMMARY AND OUTLOOK

The paper provides a formal definition of the term "semantics". At the level of understanding, at least as far as terminology is concerned, this view of the concept of semantics is, according to our findings, currently not established. Instead, there are a number of intuitively motivated approaches that intend to specify the concept of semantics, such as the knowledge pyramid or the LCIM. The paper presented these approaches and critically placed them in the context of the proposed formal definition. It serves as an invitation to a scientific discourse on a precise definition of the term "semantics". Definitions which rely on informal intuition only and do not provide precise criteria are likely to be of little value if one wishes to derive something concretely helpful from them for future practical work. This includes the definition of a reference model for the interaction semantics of network systems (Reich and Schröder, 2017) , which forms the basis for understanding the question of semantic interoperability - as the key to the design of future IIoT devices and cyber-physical systems. Similarly, standards that verbally specify the semantics of domain vocabulary do not meet criteria such as "uniqueness" or "machine interpretability". Instead, the semantics-assigning system function of the receiver system under consideration, which processes the received character, would have to be specified. In many practical cases, this is extremely time-consuming. Therefore, semantics cannot always be reasonably defined in this way. However, existing standards are often subject to restrictions regarding "machine interpretability". In order to increase this, the higher effort in accordance with the semantics definition given in this paper may be worthwhile. A few standardization projects are already implementing the ideas shown, e.g. VDI/VDE Guideline 2193/2: Language for I4.0 components - Interaction protocol for bidding procedures [17].

REFERENCES

Aamodt, A. and Nygård, M. (1995). *'Different roles and mutual dependencies of data, information, and knowledge* - An AI perspective on their integration', Data and Knowledge Engineering, 16(3), pp. 191–222. doi: 10.1016/0169-023X(95)00017-M.

Ariel, M. (2012). *Defining Pragmatics.* Cambridge University Press.

Brachman, R. & Levesque, H. (2004). *Knowledge Representation and Reasoning.* 1st Edition ed. s.l.:s.n.

Dale, N. B. (1996). *Abstract data types: Specifications, implementations, and applications.* D.C. Heath.

Hildebrandt, C. et al. (2018). '*Semantic modeling for collaboration and cooperation of systems in the production domain*', IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, pp. 1–8. doi: 10.1109/ETFA.2017.8247585.

ISO/IEC. (1994). *ISO/IEC 7498-1 - Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model*.

Kilian, C. (2005). *Modern Control Theory.* Delmar Cengage Learning.

Nielson, H. R. and Nielson, F. (2007). *Semantics with Applications: An Appetizer.* Springer-Verlag London.

Nobel, J. et Al. (2018). *Abstract and Concrete Data Types vs Object Capabilities.* In: Müller, P.; Schaefer, I. (eds) Principled Software Development. Springer, Cham.

Platform Industrie 4.0 (PI4.0) (2016). *'Aspects of the Research Roadmap in Application Scenarios*', Platform Industrie 4.0, pp. 3, 10–11.

Platform Industrie 4.0 (PI4.0) (2018). '*The Structure of the Administration Shell: Trilateral Perspective from France, Italy and Germany'*, p. 64. Available at: https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/hm-2018-trilaterale-coop.pdf?__blob=publicationFile&v=5.

Reich, J. and Schröder, T. (2017). *'A reference model for interaction semantics'.* Available at: http://arxiv.org/abs/1801.04185.

Studer, R., Benjamins, V. R. and Fensel, D. (1998). '*Knowledge engineering: Principles and methods',* Data & Knowledge Engineering, 25(1), pp. 161–197. doi: https://doi.org/10.1016/S0169-023X(97)00056-6.

Stump, A. (2013). *Programming Language Foundations.* John Wiley & Sons.

Turnitsa, C. (2005). *Extending the levels of conceptual interoperability model*. Proceedings IEEE summer computer simulation conference, IEEE CS Press.

VDI/VDE. (2019). *Guideline 2193/2 - Language for I4.0 components - Interaction protocol for bidding procedures.*