Modeling Classes of Body Sensor Networks

Marc Carwehl[1], Wolfgang Reisig[2]

Abstract: Computer-embedded systems frequently manifest in diverse variants, featuring slight differences in interfaces and functionalities, yet fundamentally grounded in a shared functional kernel. To address this variability, we propose to employ a schematic model of the functional kernel, from which concrete system instances are derived. This modeling methodology leverages well-established principles from predicate logic and Petri nets, augmented with the dynamic extensions provided by the HERAKLIT infrastructure. As a practical case study, we explore the realm of Body Sensor Networks (BSNs), a domain increasingly pivotal in the realm of medical diagnosis. Our work showcases the versatility and adaptability of our modeling framework in the context of BSNs, offering insights into its potential applications in the broader landscape of embedded systems and beyond.

Keywords: Petri nets; HERAKLIT; Body Sensor Network

1 Introduction

Body Sensor Networks (BSNs) have emerged as a vital component in modern healthcare [Ro18; YY06], providing real-time monitoring and evaluation of essential patient data. These battery-powered, wearable devices employ a network of sensors attached to a patient's body, establishing connections through wired or wireless technology. Depending on the patient's current state, different vital data are to be controlled, in different frequencies, causing different alarm patterns. To this end, a BSN may cover a number of sensors that a nurse may switch on or off during a session. Characteristic patterns of measured data may then solicit alarms. In the realm of BSNs, various versions exist, each tailored to meet the unique needs of patients with varying medical conditions. While the sensors and alarm patterns may differ for individual BSNs, the underlying structural and operational principles remain consistent. This uniformity across BSNs underscores the significance of a standardized modeling approach. With this standardized model, we can perform verification tasks on a higher level. Among the many different versions of BSN; here we focus on versions where:

- a set S of sensors is given; however, only a subset of sensors is activated in a concrete session of the BSN. A human supervisor (such as the nurse) may activate or deactivate sensors,

[1] Humboldt-Universität zu Berlin, Inst. für Informatik, Unter den Linden 6, 10099 Berlin, carwehl@cs.hu-berlin.de
[2] Humboldt-Universität zu Berlin, Inst. für Informatik, Unter den Linden 6, 10099 Berlin, reisig@cs.hu-berlin.de

- a BSN operates in rounds; each round starts and ends in an idle state. Although typically initiated by the patient or a nurse, rounds can also be triggered externally,
- data collection occurs during these rounds, with alarms triggered when the collected data deviates from normal ranges.

In our approach, we represent a BSN architecture schematically, emphasizing both static and dynamic aspects. The static characteristics are specified using a signature, a concept akin to those found in first-order logic [Ba77] and algebraic specifications [Wi90]. Any instantiation of this signature encapsulates the static attributes of a specific BSN. Dynamic aspects, on the other hand, are depicted through predicates, which apply to a set of elements subject to updates. These updates are defined using terms derived from the underlying signature, reminiscent of high-level Petri nets. Thus, we rely on three core pillars: static (signature), dynamic (predicates), and architecture (modules). To model BSNs comprehensively and systematically, we employ the Heraklit modeling technique [FR21], [FR22b], [FR22a], [FR22c] which offers a unique combination of detailed structure, abstraction capabilities, operational behavior representation, and verification methods. In this paper, we demonstrate how Heraklit enables us to represent concrete BSN systems and schemata for sets of BSN systems concurrently in one unified notation. Specifically, we show how Heraklit can be used to formally reason about safety properties of BSNs. By harnessing the power of formal verification, we elevate the precision and reliability of BSN modeling, thus ensuring the safe and effective application of this technology in healthcare. This work introduces a robust framework for BSN design, paving the way for more advanced and dependable healthcare solutions.

2 The Concept of Heraklit Modules

The architecture of a BSN is effectively modeled using Heraklit modules, a core concept in our modeling approach. In the following section, we briefly introduce the fundamental concept of Heraklit modules. Afterwards, we show modules of the BSN. At its core, a Heraklit *module* consists of a graph-like *inner structure* (which can sometimes be empty), and an *interface*. The power of Heraklit modules lies in their ability to be *composed* along their interfaces to create complex systems. One of the most vital properties of module composition, especially when dealing with extensive systems composed of numerous modules, is *associativity*. In simple terms, for three modules L, M, and N, the following compositions must yield the same module:

$$(L \cdot M) \cdot N \text{ and } L \cdot (M \cdot N). \tag{1}$$

To maintain a high level of generality, the interface of a Heraklit module consists of labeled *gates*. Intuitively, two modules M and N are composed by merging equally labeled gates from their respective interfaces. Each of these merged gates then becomes an inner element of the composed module, $M \cdot N$. However, it's worth noting that this naive composition is not associative which poses a challenge for modeling complex systems. HERAKLIT addresses this challenge by introducing a modified composition method that retains the benefits of the original idea while ensuring associativity. Specifically, the interface of a HERAKLIT module comprises two sets of distinguished labeled nodes, $^*N$ and N^*, the left and the right gates of N. All other nodes of N are considered inner nodes. Visually, a module is represented as a box, with the *left* and *right* gates positioned on its left and right margin, respectively. Figure 1 shows a module with four left gates and one right gate. No inner node is shown.

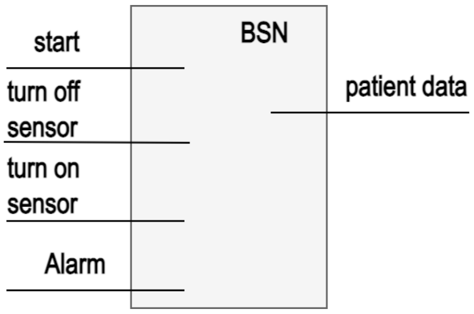

Fig. 1: Abstract view of a BSN.

Two such modules M and N can be composed, resulting in the module $M \cdot N$. Equally labeled gates of M^* and of $^*N$ are merged to an inner node within $M \cdot N$. Each inner node found in M or N also becomes an inner node of $M \cdot N$. The left interface $^*(M \cdot N)$ of $M \cdot N$ includes $^*M$ and all gates of $^*N$ without an equally labeled partner in M^*. Likewise, the right interface $M \cdot N^*$ of $M \cdot N$ includes N^* and all gates of M^* without an equally labeled partner in $^*N$. The result is a modular composition that retains associativity, allowing us to build complex systems from simpler modules. In Figure 2, we provide visual examples to illustrate this composition process in more detail. For additional information on this composition operator, please refer to [Re19; Re20].

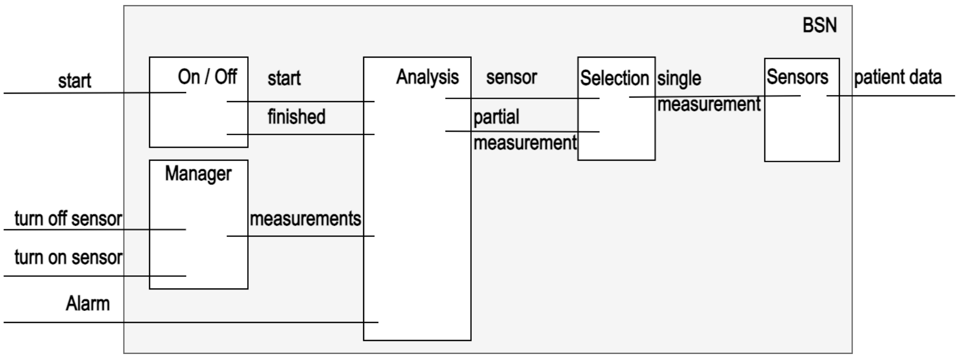

Fig. 2: Composition of a BSN module.

In summary, modules form the foundation of HERAKLIT, providing a structured and associative approach to representing complex systems. These modules enable us to break down the intricacies of BSNs into manageable components, fostering both clarity and scalability in our modeling approach.

2.1 An abstract view of the BSN

In Fig. 1, we present an abstract representation of a BSN module, as defined within the scope of our modeling approach. This abstract module encapsulates the essential elements of a BSN, facilitating a comprehensive understanding of its interaction with its environment. The gates depicted in Fig. 1 serve as the interfaces connecting the BSN module to its environment. The left gates denote buttons for stakeholders to control and adjust the BSN, and to receive alarms from the BSN; the right gate receives data from the sensors attached to the BSN's wearer. Each gate has an individual functionality. The five gates are:

- **start button:** This gate allows for the initiation of a new operational round within the BSN;
- **turn off sensor button:** Intended for human operators of the BSN, this gate enables to turn off one of the sensors;
- **turn on sensor button:** Similarly, this gate is designed for human operators and allows them to turn on sensors;
- **alarm:** The alarm gate functions as a visual or auditory alert mechanism, providing notifications or warnings as necessary.
- **patient data:** The right gate serves as the entry point for data from the sensors attached to the patient. Each sensor transmits patient data at sensor-specific intervals to the BSN, which is then processed and utilized as part of the BSN's functionality.

To be more specific, we consider BSNs that have a set of sensors S, can trigger alarms A, and handle patient data V. Internally, we employ $measurements$. A $measurement$ is a set of *key-value pairs* (s, v), where each pair assigns some patient data $v \in V$ to a sensor $s \in S$[3]. Note, that a measurement does not necessarily contain a value for each sensor in S.

2.2 Refinement of the BSN

In this section, we delve into the modules that constitute the core functionality of the BSN. Fig. 2 presents abstract versions of five modules, offering a glimpse into their roles and interactions within the broader BSN architecture:

- **The On / Off module:** This module serves as the initiator for a new operational round of the BSN. In practical terms, when the BSN is in an idle state, a nurse can press a control button.

[3] We assume here that each key is unique, i.e., there is at most one value for any sensor in a measurement.

- **The Selection module:** For each sensor integrated into the BSN, the Selection module plays a pivotal role in storing the most recent sensor value, providing the analysis module (gate *sensor*) with the most recent partial measurement for each sensor.
- **The Sensors module:** This module encapsulates the behavior of each sensor within the BSN: its most recent sensor value is updated by an external event. The module provides updated single measurements to the Selection module. Thus, one instance of this module is needed for each sensor for a particular BSN.
- **The Analysis module:** The analysis module is central to the BSN's operation as it is responsible for conducting new measurements and computing alarms. In every operational round, the Analysis module fetches new partial measurements. An alarm is computed based on this new measurement.
- **The Manager module:** As introduced in the earlier sections, a nurse can switch sensors on or off in the BSN. The Manager module handles this behavior, updating the sensors the Analysis module can rely on.

From the arrangement of these modules depicted in Fig. 2, it becomes apparent how these modules can be composed to form the cohesive system illustrated in Fig. 1. When these modules are composed, the resulting system offers the same interfaces as the abstract view, underscoring the seamless integration and compatibility of the various components. Hence, Fig. 2 can be conceived as a refinement of Fig. 1.

3 Modeling the Behavior of Modules in Heraklit

To model the behavior of a module, we employ Petri nets. More information on Petri nets can be found, e.g., in the first Chapter of [Re16]. In this section, we discuss how the modules shown previously in Fig. 2 can be modeled to exhibit the desired behavior of a BSN. To this end, we refine the general notion of a module's gate and depict each gate either as a state or a transition. We discuss the different notions of modeling a gate as a state or as a transition in this section. Additionally, we follow Heraklit's notion of using *algebraic structures* to model items and data. We will show that the same module with two different structures describes two different systems.

3.1 A BSN with Three Sensors

Fig. 3 visually represents a specific BSN configuration. In the following, we will discuss the behavior of each module:

- **On / Off:** The On / Off module contains the *start* gate which allows a nurse to start an operational round of the BSN. Such a round can be started at any point when there is a token on place *finished*, resembling that the previous round has concluded. This resembles an *idle* state. To start a new round, the token is consumed by the *start* transition, and a token is emitted on the *start* place in the module's right interface.

- **Manager:** The Manager module enables a nurse to turn sensors on or off. To this end, the module features two transitions in its left interface. Both transitions can only be enabled when the latest round has concluded, i.e., the system is in an idle state. To turn a sensor on, a measurement for the sensor is added to the Analysis module (place *measurements*). However, the value associated with the newly added sensor may be outdated. Therefore, we replace it with a dummy value ⊥. Similarly, a sensor can be turned off when the system is in an idle state by taking the corresponding measurement out of the Analysis module (place *measurements*). The places *sensors on* and *sensors off* store information about the sensors that are turned on or off, respectively.
- **Sensors:** Moving on to its right interface, a BSN is connected to real sensors. The Sensors module handles interaction with these Sensors. The transition *patient data* can be invoked anytime by a sensor to update its measurement (place *single measurements*).
- **Selection:** The measurements (place *single measurements*) are made available to the Analysis module by the Selection module. In essence, the Selection module can fetch a measurement (place *partial measurement*) for any of the sensors (place *sensor*).
- **Analysis:** Finally, the Analysis module is started by the On / Off module and requests new measurements for the sensors using its interface to the Selection module. Once all measurements are gathered (place *partial measurements*), they are merged into one measurement (transition *d*). An alarm is computed at transition *e* and output to the nurse at place *Alarm* which resembles one of the left gates of a BSN. Finally, a token is put on the *finished* place to depict the end of the round.

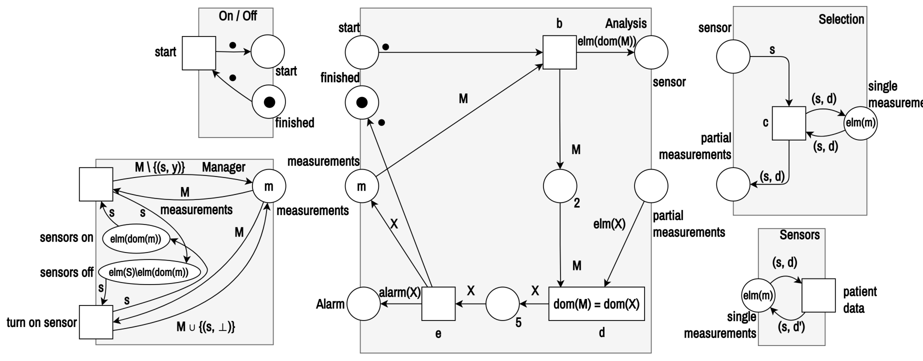

Fig. 3: Petri nets depicting behavior inside the five modules.

To improve the patient's comfort and system efficiency, not all sensors should be engaged in every round. For instance, sensors like blood pressure sensors, known for their potential discomfort to the patient, are selectively chosen. A refinement of the analysis module utilizes a function f to determine which activated sensors to engage in a round. The refined module is shown in Fig. 4 while Fig. 5 shows a structure for a BSN with three sensors.:

- **Refined Analysis:** In the refined analysis, a function f is invoked at transition a to dissect the set of activated sensors. Those that are engaged in this round (place *1*) are used to request new measurements from the Selection module. For disengaged sensors, their measurements (place *3*) are added to the freshly collected measurement at transition e to compute an alarm. Afterward, the measurement is changed once more at transition f where values for disengaged sensors are re-set to the dummy value $\perp$. The round concludes with a token on the *finished* place.

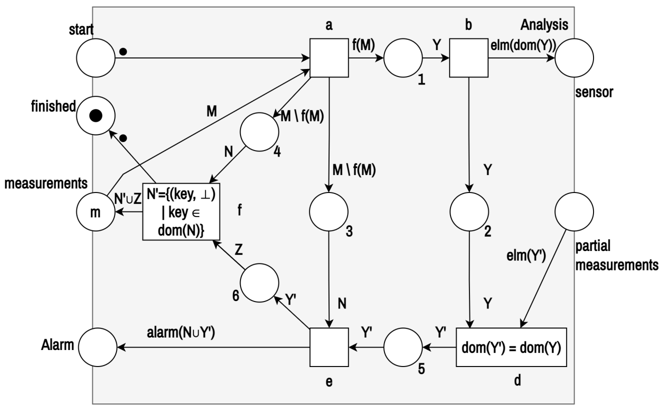

Fig. 4: Refined version of the Analysis Module.

This BSN system (cf. Fig. 3, 4) adheres to the abstract view of BSNs discussed earlier (cf. Fig. 2). It serves as a concrete instantiation of how the system's interfaces can be interacted with by the external environment. Specifically, the *start* gate of the On/Off module is modeled as a transition, symbolizing the system's responsiveness to external triggers. In this case, the nurse can initiate a new round by triggering this transition. Similarly, the *turn sensor on* and *turn sensor off* gates are depicted as transitions, representing the activation or deactivation of sensors. Importantly, these transitions can only be executed when the BSN is in an idle state, as there must be a measurement present on the *measurements* gate. This condition is inherently satisfied only during the BSN's idle state, a claim that we will formally prove in Section 6. Finally, the *alarm* gate is modeled as a place, depicting it as a light that can be on or off, depending on the patient's health status. It's noteworthy that HERAKLIT permits the modeling of gates as both places and transitions. When a gate is represented as a transition, it signifies the capability to trigger specific system behaviors, as exemplified by the *start* gate. Conversely, when a gate is depicted as a place, it signifies the capacity to convey data, as demonstrated by the *alarm* gate.

3.2 A BSN with 10 sensors

It is evident that various use cases may necessitate different BSN configurations. For instance, another patient may require a different set of sensors compared to the BSN previously described. It's important to note that altering the set of sensors does not necessitate changes to the modules or their composition. However, the system's structure must adapt to accommodate the additional sensors within the corresponding domain. Functions, constants, and data representations must also be adjusted accordingly.

Fig. 6 shows one possible structure for a BSN with 10 sensors. This depiction underscores HERAKLIT's adaptability in accommodating diverse variants of a system model. While, in a BSN, the sensors and their properties may vary, the core modules and their interactions remain consistent with the previously discussed BSN structure (cf. Fig. 4).

This BSN configuration serves as a testament to the flexibility of the HERAKLIT modeling approach, as it can seamlessly represent various BSN scenarios with distinct sets of sensors, each tailored to address specific patient needs and medical requirements. This versatility ensures that the HERAKLIT model can effectively capture the complexity and nuances of real-world systems.

Domains
Sensor = {thermometer, pulse, blood pressure, glucometer}
Alarm = $\{true, false\}$
patient data = $\mathbb{N}$
control = $\{\bullet\}$

Derived Domains
measurements: $\mathcal{P}(\{(key, value)|$
$key \in$ Sensor and
$value \in$ patient data $\cup \{\perp\}$
and each key is unique })

Functions
alarm : measurements $\rightarrow$ Alarm
alarm(m) = $true$, iff
$(x, d) \in m$ with $d = \perp$ or
($x =$ thermometer and $d > 40$) or
($x =$ pulse and $d > 180$) or
($x =$ glucometer and $d > 200$);
$false$ otherwise
f : measurements $\rightarrow$ measurements
$f(m) = m \setminus \{(thermometer, d)\}$
if $(thermometer, d) \in m$ and $d < 40$;
m otherwise

Constants
m : measurements
$m = \{(thermometer, 38), (pulse, 70)\}$
S : $\mathcal{P}(Sensor)$ S = {thermometer, pulse, blood pressure, glucometer}

Variables
M, N, Y, Y', Z : measurements
s : Sensor
d : patient data $\cup\{\perp\}$
d' : patient data

Fig. 5: Structure S1.

Domains
Sensor = {thermometer1, pulse1, blood pressure1, glucometer1, oxygen saturation1,
thermometer2, pulse2, blood pressure2, glucometer2, oxygen saturation2}
Alarm = $\{true, false\}$
patient data = $\mathbb{N}$
control = $\{\bullet\}$

Derived Domains
measurements: $\mathcal{P}(\{(key, value)|$
$key \in$ Sensor and
$value \in$ patient data $\cup \{\perp\}$
and each key is unique})

Functions
alarm : measurements $\rightarrow$ Alarm
alarm(m) = $true$, iff
$(x, d) \in m$ with $d = \perp$ or
($x =$ thermometer1 and $d > 40$) or
($x =$ pulse1 and $d > 180$) or
($x =$ glucometer1 and $d > 200$);
$false$ otherwise
f : measurements $\rightarrow$ measurements
$f(m) = m \setminus \{(thermometer1, d)\}$
if $(thermometer1, d) \in m$ and $d < 40$;
m otherwise

Constants
m : measurements
$m = \{(thermometer1, 38), (pulse2, 70)\}$
S : $\mathcal{P}(Sensor)$
S = {thermometer1, pulse1, blood pressure1, glucometer1, oxygen saturation1, thermometer2, pulse2, blood pressure2, glucometer2, oxygen saturation2}

Variables
M, N, Y, Y', Z : measurements
s : Sensor
d : patient data $\cup\{\perp\}$
d' : patient data

Fig. 6: Structure S2.

4 Using Signatures in HERAKLIT

HERAKLIT employs the well-established spirit of predicate logic [Ba77] and Algebraic specifications [Wi90] to further generalize system models. To this end, we define a signature, Σ, i.e. a collection of symbols for sets of sensors, measurements, and symbols for functions that map from one domain to another. Finally, the signature captures symbols for constants, including initial markings, and variables in the signature that can be employed when modeling with Petri nets.

4.1 Signature for BSNs

Domain Symbols
Sensor
Alarm
patient data
control
Derived Domain Symbols
measurements
Function Symbols
alarm : measurements $\rightarrow$ Alarm
f : measurements $\rightarrow$ measurements
Constant Symbols
m : measurements
S : $\mathcal{P}(Sensor)$
Variables
M, N, Y, Y', Z : measurements
s : Sensor
d, d' : patient data $\cup\{\perp\}$

Fig. 7: Σ. Signature for BSNs.

Fig. 7 shows the signature Σ. Σ has symbols for the sensors, alarms, patient data used in the BSN, and control tokens. Additionally, Σ contains symbols for functions that determine the alarm, and which sensors to engage in a round. Finally, the constant symbol *m* depicts the initial measurement with which the BSN starts and the symbol *S* depicts the set of all sensors.

4.2 Instantiations of Σ

As usual in logic, an instantiation of Σ is a structure, that replaces each symbol in Σ by a corresponding set, function, or element (Predicate logic denotes such instantiations as models). For example, the structures S1 and S2 of Section 3 are instantiations of Σ. Obviously, Σ generates infinitely many instantiations.

5 Making use of the concept of modularity in HERAKLIT

With HERAKLIT's focus on modules, it becomes easy to perform behavioral changes locally, without having to change the entire system model. In particular, changes can often be constrained to a small number of modules. In this section, we discuss different variants of BSNs and show how the system model can be changed to accommodate them. To showcase how locally constrained the changes are, we only concentrate on instances of modules that have been changed, while unchanged modules w.r.t. the system from Fig. 3 with the Analysis module from Fig. 4 remain abstract. This shall again highlight to the reader that changes are made only locally. In the remainder of this section, we will show different variants of modules in the BSN. To highlight the changed modules, we keep an abstract representation of all modules that have not changed. Incidentally, the algebraic specification Σ remains intact for any of the changes we discuss in this section.

5.1 Displaying the idle states

For the On / Off module, it is intuitively useful to accompany the *start* button with a display in the left interface, showing whether the BSN is in an idle state. The place *finished* can serve this role. The module's right interface contains this place already; so the place will now be located in both the left and the right interface. In the left interface, we will label it *ready*. The HERAKLIT framework allows a gate to serve coincidentally in both interfaces of a module. This raises the problem of the graphical representation of a node that belongs to both interfaces. Graphically satisfying are two instances of the node representation, linked by a double line. Fig. 8 shows an example.

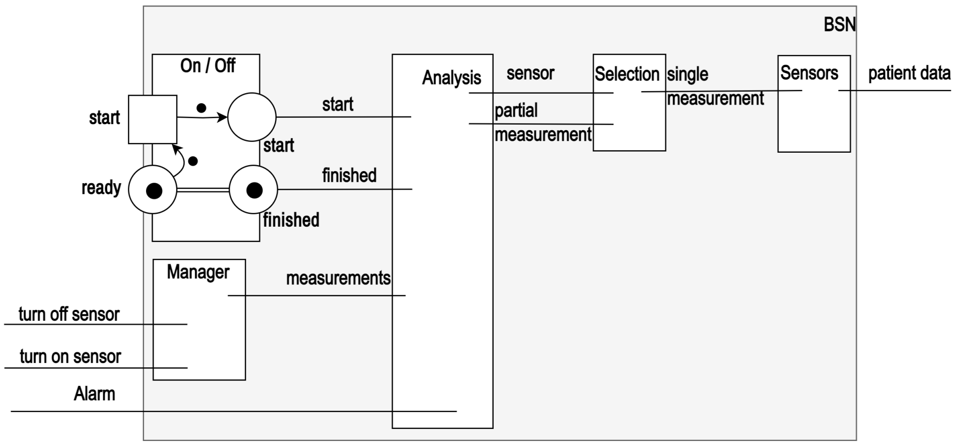

Fig. 8: Refined version of the On / Off module.

5.2 Displaying the activated sensors

The left interface of the abstract models of Fig. 1 and Fig. 2 includes the two gates labeled *turn off sensor*, and *turn off sensor*. However, before pressing such a button, a user would prefer to understand whether or not pressing the buttons will have an effect, i.e., to understand whether or not the corresponding sensor is already active or not. To this end, the places *sensors off* and *sensors on* of Fig. 4 are moved from the inside of the module to the left interface, as shown in Fig. 9. These places can be conceived of as displays for the activation status of the corresponding sensors. Using the idle state display described above, the user will also see if the BSN is in an idle state and whether the activation will be handled or not.

Intuitively, places are adequate for this purpose as the goal is to provide information to the environment, in contrast to the buttons we model as transitions.

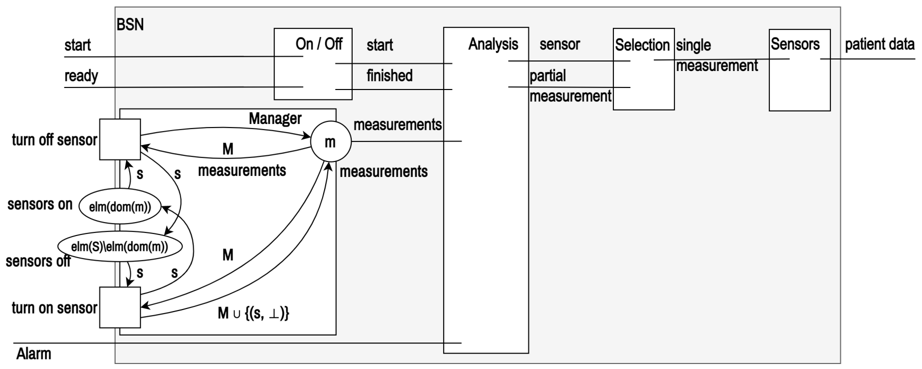

Fig. 9: Refined version of the Manager module.

5.3 Enforced Updates

The system schema and model in Figs. 1, 4 model the case where the most recent sensor data are employed. This includes the case of multiple uses of the same measurements. In some cases, however, it may be interesting to have a system that can trigger a sensor to generate a fresh value. Fig. 10 models this case by forcing the sensor to get new data in the right interface.

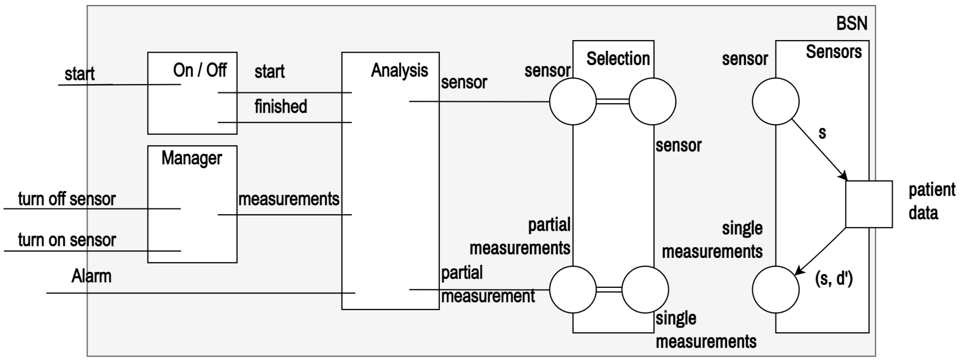

Fig. 10: Refined version with enforced updates.

5.3.1 Timeouts

An enforced update causes a deadlock in case the sensor is broken or does not react for whatever reason. For this case, a timeout mechanism may be installed, returning the previous value or the *dummy value* ⊥, and informing the users about the timeout event. Fig. 11 models this case, employing an external *trigger*, such as a timer, to depict a timeout.

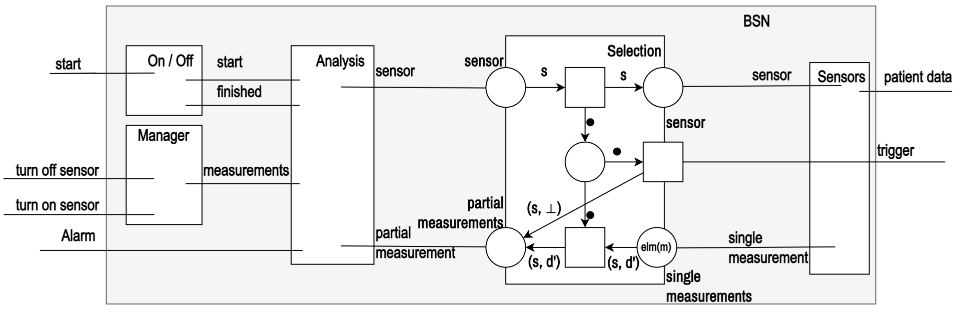

Fig. 11: Refined version with timeouts.

6 Verification

In the realm of safety-critical systems like BSNs, ensuring that the system behaves precisely as specified is of paramount importance. Stakeholders often demand rigorous proofs to guarantee the system's reliability. Fortunately, HERAKLIT leverages well-established concepts, including predicate logic and Petri nets, providing a structured framework for systematic verification.

In this section, we illustrate the power of HERAKLIT's foundation by showcasing a pivotal property of the BSN model. Before delving into the details, it's essential to establish some standard terminology and notations for Petri nets and Petri net schemata, which are introduced in [Re98]. Our earlier discussion highlighted the cyclic nature of BSN operations, where each round starts and concludes in an idle state. We now turn our attention to the verification of this crucial property.

To rigorously verify the idle state in BSNs, we introduce the concept of *idle markings* within the HERAKLIT framework. An instantiation of the module *Analysis* in Fig. 4 is considered idle if: (i) The place *measurements* contains a token (of type measurement), (ii) either the place *finished* or the place *start* contains a token (represented as a black dot), and (iii) no other place within the *Analysis* module contains a token.

Surprisingly, it turns out that property (i) alone is sufficient to characterize idle markings effectively. This leads us to the following theorem:

Theorem 1. *Let I be an instantiation of the module in Fig. 4 of the system shown in Fig. 3. A reachable marking M of I is* idle *iff the place* measurements *contains a token.*

Tab. 1: Matrix A.

	start	a	e	b	g	d	Turn off sensor	Turn on sensor	c	patient data
finished	-·				+·					
start	+·	-·								
measure-ments		-M			+N’∪Z		-M, +M\{(s. y)}	-M, +M ∪ {(s, ⊥)}		
alarm			+alarm(N∪Y’)							
1		+f(M)		-Y						
2				+Y		-Y				
3		+M\f(M)	-N							
4		+M\f(M)			-N					
5			-Y’			+Y’				
6			+Y’		-Z					
sensor				+elm(dom(Y))					-s	
p. mea-surem.						- elm(Y’)			+(s, d)	
sensors on							-s	+s		
sensors off							+s	-s		
s. mea-surem.									-(s, d), +(s, d)	-(s, d), +(s, d’)

To prove this theorem at the schematic level, we exploit a structural property of the schema. We represent S1 as a matrix, A, where each place p of S1 contributes a line, and each transition contributes a column. An entry A(p,t) in this matrix collects the inscriptions of the arcs that connect place p and transition t. Table 1 shows this matrix.

Tab. 2: Invariants and initial marking.

	M_0	i1	i2	i3	i4	i5	i6
finished	•					|x|	
start						|x|	
measurements							
alarm							
1				|x|			pr1(x)
2							pr1(x)
3			|x|				pr1(x)
4		|x|					
5				|x|			pr1(x)
6			|x|	|x|			pr1(x)
sensor					|x|		
partial measurements					|x|		
sensors on	elm(dom(m))						
sensors off	elm(S) \elm(dom(m))						
single measurements	elm(m)						

We formulate an equational system, $A^{\tau} \cdot x = 0$, where each entry of a solution vector x (referred to as an invariant, found in Table 2) is a multiset of terms with one variable. This system is instrumental in proving the fundamental property of invariants, which claims a property of all reachable markings:

Theorem 2. *For each invariant x, each instantiation I, each reachable marking M of I, and M_0 denoting the initial marking as shown in Table 2 holds:*

$$x \cdot M = x \cdot M_0.$$

Proof of Theorem 2 can be found in the literature. To illustrate, one of the invariants, $i1$, implies that the number of tokens on places *measurement* and *place 4* sums up to 1. This, in turn, guarantees that a token on *place measurement* implies *place 4* contains no token. Corresponding arguments with the invariants $i2$, $i3$, and $i4$ imply property (iii) of the definition of invariants. We also introduce an intriguing invariant, $i5$, which implies property (ii) of the definition of idle markings. It signifies the balance between tokens on *finished* and *start* relative to *measurement*. Lastly, a more intricate invariant, $i6$, demonstrates that each reachable marking accurately represents each sensor exactly once in the Analysis module. These compelling proofs not only validate the integrity of the BSN model but also exemplify the robustness of HERAKLIT in facilitating systematic and rigorous verification processes.

7 Related Work

Rodrigues et al. [Ro18] introduced a valuable approach by modeling the requirements of a BSN using a Contextual Goal Model (CGM). Their methodology involves the construction of explicit goals that the BSN aims to fulfill, followed by the design of tasks and sub-tasks to achieve these goals. Subsequently, the authors developed a model of the BSN, represented as timed automata, aligning with their CGM. Using timed automata, they conducted property verification through temporal logic using model-checking techniques. While Rodrigues et al.'s work successfully models and verifies a specific BSN instance, it is important to note that their formalism does not readily facilitate abstraction to a class of similar systems. Similar advancements employing timed automata have been made by Vogel et al. [Vo23]. Their work contributes to the modeling and verification of BSNs using timed automata, adding to the body of research in this domain. An additional approach has been conducted by Araújo [Ar23] who modeled a BSN with OpenModelica. This approach provides novelty regarding the continuous data flow. However, it is crucial to highlight that while these approaches excel in modeling and verifying individual BSN instances, they lack the capacity for abstract representation applicable to a broader class of systems, as HERAKLIT achieves through its use of structures and signatures. Carwehl et al. [Ca23] recently explored the verification of a dynamically changing BSN, focusing on changes specified by their *Property Adaptation Patterns*. Their work addresses the important aspect of adaptability within BSNs. Nevertheless, it is worth noting that despite their valuable contributions, the use of runtime verification in their approach requires a running system, while the verification shown in this paper can be performed earlier in the development cycle.

In summary, while prior research has made significant strides in modeling and verifying BSNs, our HERAKLIT framework distinguishes itself by its unique ability to abstract to classes of similar systems through structures and signatures, modules and their composition. This feature empowers HERAKLIT with a high degree of versatility and generality, making it an advantageous choice for modeling and analyzing complex applications with dynamic, adaptive, and evolving characteristics.

8 Conclusion

In this paper, we've demonstrated the effectiveness of HERAKLIT's architectural, static, and dynamic modeling approach in capturing a class of BSNs. By leveraging established techniques such as Petri nets, algebraic structures, and modules, we successfully conducted formal reasoning about the safety properties of these systems. For future work, we aim to enhance our model by incorporating more insights from healthcare professionals. Moreover, the verification techniques we've applied can serve as a valuable tool in discussions with experts. They can aid in precisely assessing the behavior of BSNs and identifying potential safety issues. This can facilitate productive dialogues between modelers and domain experts, leading to more effective and trustworthy BSN systems for healthcare.

References

[Ar23] Araújo, J. P. C. d.: Enhancing runtime monitors of cyber-physical systems using negative selection./, 2023.

[Ba77] Barwise, J.: An introduction to first-order logic. In: Studies in Logic and the Foundations of Mathematics. Vol. 90, Elsevier, pp. 5–46, 1977.

[Ca23] Carwehl, M.; Vogel, T.; Rodrigues, G.; Grunske, L.: Runtime Verification of Self-Adaptive Systems with Changing Requirements. In: 2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). IEEE Computer Society, Los Alamitos, CA, USA, pp. 104–114, May 2023, URL: https://doi.ieeecomputersociety.org/10.1109/SEAMS59076.2023.00024.

[FR21] Fettke, P.; Reisig, W.: Handbook of HERAKLIT, tech. rep., Heraklit working paper, v1. 1, 2021.

[FR22a] Fettke, P.; Reisig, W.: Breathing Life into Models: The Next Generation of Enterprise Modeling. In (Fill, H.; van Sinderen, M.; Maciaszek, L. A., eds.): Proceedings of the 17th International Conference on Software Technologies, ICSOFT 2022, Lisbon, Portugal, July 11-13, 2022. SCITEPRESS, pp. 7–14, 2022, URL: https://doi.org/10.5220/0011376600003266.

[FR22b] Fettke, P.; Reisig, W.: HERAKLIT. In: Gronau, Norbert ; Becker, Jörg ; Kliewer, Natalia ; Leimeister, Jan Marco ; Overhage, Sven (Herausgeber): Enzyklopädie der Wirtschaftsinformatik – Online-Lexikon. [Online; Stand 16. October 2023], Berlin : GITO, 2022, URL: https://wi-lex.de/index.php/lexikon/entwicklung-und-management-von-informationssystemen/systementwicklung/software-engineering/heraklit/.

[FR22c] Fettke, P.; Reisig, W.: Modellieren mit Heraklit. In (Riebisch, M.; Tropmann-Frick, M., eds.): Modellierung 2022, 27. Juni - 01. Juli 2022, Hamburg, Deutschland. Vol. P-324. LNI, Gesellschaft für Informatik e.V., pp. 77–92, 2022, URL: https://doi.org/10.18420/modellierung2022-005.

[Re16] Reisig, W.: Understanding petri nets. Springer, 2016.

[Re19] Reisig, W.: Associative composition of components with double-sided interfaces. Acta Informatica 56/3, pp. 229–253, 2019.

[Re20] Reisig, W.: Composition of component models-a key to construct big systems. In: International Symposium on Leveraging Applications of Formal Methods. Springer, pp. 171–188, 2020.

[Re98] Reisig, W.: Elements of distributed algorithms: modeling and analysis with Petri nets. Springer Science & Business Media, 1998.

[Ro18] Rodrigues, A.; Caldas, R. D.; Rodrigues, G. N.; Vogel, T.; Pelliccione, P.: A learning approach to enhance assurances for real-time self-adaptive systems. In: Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems. Pp. 206–216, 2018.

[Vo23] Vogel, T.; Carwehl, M.; Rodrigues, G. N.; Grunske, L.: A property specification pattern catalog for real-time system verification with UPPAAL. Information and Software Technology 154/, p. 107100, 2023.

[Wi90] Wirsing, M.: Algebraic specification. In: Formal models and semantics. Elsevier, pp. 675–788, 1990.

[YY06] Yang, G.-Z.; Yang, G.: Body sensor networks. Springer, 2006.