



**HAL**  
open science

# A framework for modeling and analyzing cyber-physical systems using statistical model checking

Abdel-Latif Alshalalfah, Otmane Ait Mohamed, Samir Ouchani

► **To cite this version:**

Abdel-Latif Alshalalfah, Otmane Ait Mohamed, Samir Ouchani. A framework for modeling and analyzing cyber-physical systems using statistical model checking. *Internet of Things*, 2023, 22, pp.100732. 10.1016/j.iot.2023.100732 . hal-04108550

**HAL Id: hal-04108550**

**<https://hal.science/hal-04108550>**

Submitted on 11 Jan 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

1 **A Framework for Modeling and Analyzing**  
2 **Cyber-Physical Systems using Statistical Model**  
3 **Checking**

4 **Abdel-Latif Alshalalfah**  
5 **Otmane Ait Mohamed · Samir Ouchani**

6  
7 Received: date / Accepted: date

8 **Abstract** The trustworthiness of a cyber-physical system is essential for it  
9 to be qualified for utilization in most real-life deployments. This is espe-  
10 cially critical for systems that deal with precious human lives. Although these  
11 safety-critical systems can be investigated using both experimental testing and  
12 model-based verification, accurate models have the potential to permit risk-  
13 free mimicking of the system behavior even in the most extreme scenarios.  
14 To overcome the CPS modelling and design challenges, the INCOSE/OMG  
15 standard System Modeling Language (SysML) is utilized in this work to ac-  
16 curately specify cyber-physical systems. For that, a bounded set of SysML  
17 constructs are defined to precisely capture the semantics of continuous-time  
18 and discrete-time system behaviors. Then, the SysML constructs are substi-  
19 tuted by developing a new algebra, called Enhanced Activity Calculus (EAC).  
20 So, EAC helps construct equivalent priced timed automata models by develop-  
21 ing a new systematic procedure to correctly translate the SysML models into  
22 the statistical model checking tool UPPAAL-SMC inputs. The latter checks  
23 whether the system is correct and safe or not. Moreover, the soundness of the  
24 developed translation mechanism has been proved and its effectiveness has  
25 been shown on a real use case, namely the artificial pancreas.

26 **Keywords** System Modeling Language · Enhanced Activity Calculus ·  
27 Cyber-Physical Systems · Model Transformation · Model-Based Verification ·  
28 Safety-Critical · Statistical Model Checking · Priced Timed Automata

---

A.-L. Alshalalfah  
Department of Electrical & Computer Engineering, Concordia University, Montreal, Canada.  
E-mail: abdellatif.alshalalfah@mail.concordia.ca

O. Ait Mohamed  
Department of Electrical & Computer Engineering, Concordia University, Montreal, Canada.

S. Ouchani  
CESI Lineact, Aix-en-Provence, France.

## 29 Nomenclature

30	<i>CPS</i>	Cyber-Physical Systems
31	<i>EAC</i>	Enhanced Activity Calculus
32	<i>HA</i>	Hybrid Automata
33	<i>MITL</i>	Metric Interval Temporal Logic
34	<i>ODE</i>	Ordinary-Differential Equations
35	<i>ODESCD</i>	ODE of SysML Constraint Diagram
36	<i>PID</i>	Proportional-Integral-Derivative
37	<i>PTA</i>	Priced Timed Automata
38	<i>SMC</i>	Statistical Model Checking
39	<i>SysML</i>	Systems Modeling Language

## 40 1 Introduction

41 Whether human-operated or autonomous, embedded systems are designed to  
42 improve the quality of life for people. From embedded computing to distributed  
43 systems, Cyber-Physical Systems (CPS) refer to computing systems that inter-  
44 act with control and management objects [53]. As technology advances,  
45 CPS is being used in a wide range of applications [30]. With the reduction  
46 in size and cost of hardware, along with accelerating innovation and advance-  
47 ment in sensor and computational technologies, CPS has been able to spread  
48 to all types of applications. Through horizontal expansion, CPS has gained  
49 popularity in all types of application. Also, CPS flourished vertically to find a  
50 foot in more complex and dependable applications. From daily applications,  
51 the various success stories have encouraged designers to develop CPS for au-  
52 tonomous control compared to the early systems which required some degree  
53 of human interaction [23, 41]. Nowadays, wireless body area networks are uti-  
54 lized to connect devices that observe the status of the physiological dynamics  
55 [16]. As a result, health conditions can be monitored and treated in a timely  
56 manner. Patients with chronic diseases will particularly benefit from this. For  
57 example, with around half a billion diabetes worldwide [42], an automatic glu-  
58 cose controller is necessary for them to live a normal life while still avoiding  
59 the health complications related to their situation.

60 In order to get approval certificates from the appropriate authorization  
61 entities, these systems must prove their safety and robustness under all sce-  
62 narios [11]. However, for real-life deployments, only qualified systems must  
63 meet these safety requirements. From the first prototype to the final fabri-  
64 cated product, verifying the safety of CPS is a vital step in the development  
65 process. The system-level analysis provides feedback early in the design pro-  
66 cess, and by identifying safety issues early, time and resources are not wasted  
67 [29]. Additionally, the system-level analysis helps understand CPS limitations  
68 and define the requirements of CPS components for safe operation. Further-  
69 more, CPS can be verified under extreme scenarios that would be impossible

70 to conduct in real life without taking extraordinary risks by using appropriate  
71 realistic models.

72 Analyzing systems at the system level is either accomplished through sim-  
73 ulation testing or through formal methods. In the former approach, specific  
74 input scenarios can be used to evaluate CPS behavior. Yet, it does not give  
75 confidence on the state space coverage. On the other hand, formal methods  
76 such as model checking [10] provide exhaustive coverage for the whole state  
77 space. Unfortunately, formal techniques do not scale well for realistic hybrid  
78 systems and suffer from the infamous state space explosion problem [21].

79 As a compromise between these two approaches, Statistical Model Check-  
80 ing (SMC) can be used for verification. Although it does not provide exhaustive  
81 coverage for the state space, SMC can be used to introduce statistical guaran-  
82 tees for safety properties with feasible computational resources. In a nutshell,  
83 the following are the main contributions of this work.

- 84 – Proposing a novel systematic procedure to capture the semantics of SysML-  
85 based diagrams and to construct its equivalent PTA models for SMC anal-  
86 ysis.
- 87 – The effectiveness of the proposed framework to analyze a medical CPS is  
88 demonstrated on an artificial pancreas case study. In particular, the safety  
89 of the system is verified using SMC to evaluate the ability of three control  
90 configurations to mitigate message errors.

91 Below is an outline of the remainder of the article. The literature review is  
92 presented in Section 2, and then Section 3 demonstrates, through an artificial  
93 pancreas example, the SysML graphical and textual modeling. Afterwards,  
94 Section 4 introduces the new proposed automatic construction of equivalent  
95 Priced Timed Automata (PTA) models and proves the soundness of the de-  
96 veloped approach. Section 5 illustrates the experiments conducted for model  
97 validation and safety verification procedure by an example experiment, and  
98 section 6 concludes the article.

## 99 2 Literature Review

100 With the growing demand for CPS applications, several research works have  
101 investigated the verification and safety analysis problems related generally to  
102 CPS. Based on our surveyed initiatives, we have identified two main categories:  
103 *Formal verification* and *Simulation* based approaches.

### 104 2.1 Simulation based approaches

105 Even before the advent of modern computer systems, the term *Simulation* is  
106 known as the process of designing a model of a real system to conduct exper-  
107 iments [52]. These experiments aim at understanding the system's behavior  
108 or evaluating a strategy associated with the system. Simulation software tools

109 have flourished with the advent and availability of low-cost computational  
110 systems.

111 Liu et al. [37] have used the open-source toolkit MATSim [55] to inves-  
112 tigate large-scale transportation patterns for shared autonomous vehicles. In  
113 their work, agent-based modelling is applied to estimate mode choices between  
114 human-driven vehicles, shared autonomous vehicles, and public transit. Fol-  
115 lowing a cost function that takes into account, the out-of-pocket, the trip time,  
116 and the waiting time, each driver chooses one of the three options of travel  
117 mode. The analysis is done for different fare levels, demographic settings, and  
118 shared autonomous vehicles availability to give implications on sustainability.

119 In [32], an assessment of the safety of leader-follower configurations for  
120 autonomous radar semi-trucks is made based on different environmental condi-  
121 tions. The simulation model is developed with the commercial platforms  
122 AmeSim, PreScan, and Matlab-Simulink to study the effect of environmental  
123 conditions on safety margins in semi-truck convoy platooning. The autonomy  
124 in their simulated vehicles is enabled by adopting sensors for radar, global  
125 positioning systems, and short-range inter-vehicle communication.

126 Instead of fully autonomous vehicles, the work in [5] addressed semi-autonomous  
127 vehicles implementing adaptive cruise control coexisting with regular vehicles  
128 and trucks. The vehicles enter the four-lane highway with a user-predefined  
129 arrival rate in the microscopic Java-based F.A.S.T. traffic simulator. Their  
130 findings show that a high penetration of semi-autonomous vehicles can in-  
131 crease traffic performance, especially under high traffic conditions.

132 Connected and autonomous vehicles and their impact on road safety are  
133 discussed in [47]. Initially, the simulation software VISSIM is utilized to study  
134 a test-bed that mimics a three-lane motorway with traffic statistics measured  
135 from a real one in England. A lateral and longitudinal control algorithm is then  
136 tested for its ability to reduce traffic conflicts at different market penetration  
137 rates.

138 From a healthcare perspective, a falsification approach is presented in  
139 [48, 9] to simulate and verify the artificial pancreas controller in a simulation  
140 environment. The S-Taliro tool which applies falsification simulations termi-  
141 nates with either finding a safety violation or failure to find, without the  
142 explicit guarantee that such one does not exist. Instead, the tool uses robust-  
143 ness metric to predict the distance between simulation outcomes and safety  
144 margins.

## 145 2.2 Formal based approaches

146 Unlike the numerical simulation approaches which mimic the behavior of real  
147 systems, formal methods apply analytical reasoning to derive mathematically-  
148 proven properties that characterize the system behavior. These characteristics  
149 are not always attainable, but when achieved they provide guaranteed out-  
150 comes which is an asset that helps verify safety-critical systems.

151 In [28], piece-wise affine hybrid automata was used to analyze the wind  
152 turbine dynamics in SpaceEx verification platform [20]. Even though Kekatos  
153 et al. reduced some blocks for better scalability, the resulting model contained  
154 around 16 million locations, which would hinder the ability to analyze more  
155 elaborate systems. However, classical hybrid automata (HA) tools and method-  
156 ologies suffer from this limitation [49].

157 The problem of formally analyzing a swarm of robots is handled by Schupp  
158 et al. [50]. The cooperative decentralized robots are modeled as a hybrid sys-  
159 tem and investigated by *flowpipe* analysis where the sets of reachable states  
160 are iteratively over-approximated [19]. Although the work in [50] deals with  
161 a simple model of distributed synchronization, it still causes some scalabil-  
162 ity challenges that are partially encountered by compositional analysis and  
163 optimized transition emulation.

164 Using a combination of simulations and formal analysis, [46] examines  
165 patient-controlled analgesia's safety. So, to analyze the resulting CPS, its de-  
166 tailed behavior is modeled in Simulink. Then, to qualify the CPS for model  
167 checking, the continuous dynamics are abstracted away from the system model  
168 and then replaced by simple timing constraints with the target to be analyzed  
169 in UPPAAL model checker [7]. Additionally, UPPAAL is also used in [26] to  
170 verify control algorithms in a dual chamber implantable pacemaker. Mean-  
171 while, a timed automata representation of the heart and the pacemaker are  
172 used to specify the ability of the algorithms to avoid unsafe regions of the state  
173 space. The proposed approach covers the whole state space, yet only the state  
174 space that is modeled. Thus, this excludes certain control and physiological  
175 behaviors that are beyond the expressive power of the modeling language and  
176 the computational feasibility of the verification technique. In fact, these be-  
177 haviors can be skipped in some systems but are essential to correctly analyze  
178 hybrid systems with continuous-time variables.

### 179 2.3 Statistical model checking based approach

180 SMC consists of observing a number of simulation runs or system executions  
181 and using statistical methods to reason about formal properties [35].

182 After some preliminary works such as the hypothesis testing of modal prop-  
183 erties in process algebra [33], initial results for SMC had witnessed progress  
184 since 2002 [58] with the corresponding term introduced for the first time in  
185 2004 [51]. Reasoning about reachability problems with SMC algorithms pro-  
186 vides mainly guarantees on the probability error bound. Depending on the type  
187 of reachability expression being dealt with, the error bound can be calculated  
188 by utilizing the appropriate classical mathematics such as Monte Carlo with  
189 Chernoff-Hoeffding error bounds [43, 24] or hypothesis testing using Wald's  
190 sequential analysis [56].

191 Different tools exist that implement SMC algorithms such as PRISM [31],  
192 UPPAAL-SMC [12, 15], BIP [39], and Ymer [57]. Since their inception, SMC  
193 tools have been utilized to study many discrete-time and continuous-time sys-

194 tems. To list a few: airplane cabin communication system [6], distributed sen-  
195 sor network [36], energy-aware house heating [13], biological mechanisms of  
196 the genetic oscillator [14], real-time streaming protocol [44], artificial pancreas  
197 [2, 4], anesthesia control [3], and coordinated emergency braking system [1].

## 198 2.4 Model Construction

199 In order to analyze the system, it is necessary to first convert the specifications  
200 into the modeling language used by the analysis tool. Furthermore, an ade-  
201 quate level of expertise is required to model the system properly when done  
202 manually. Furthermore, formal modeling languages tend to be more error-  
203 prone due to their low readability. Therefore, the need arises to facilitate the  
204 process of constructing formal models by automatically translating high-level  
205 models that incur better readability.

206 In [28], the system modeled in Simulink is translated into SpaceEx mod-  
207 eling language in four steps. After the Simulink model is modified to comply  
208 with the verification standards, the tool SL2SX [40] is employed to handle the  
209 main translation step and construct a SpaceEx model. Afterwards, composi-  
210 tional syntactic hybridization [27] and validation are conducted to achieve a  
211 model ready to be analyzed.

212 An approach to transform Simulink models into UPPAAL-SMC is proposed  
213 in [18]. The work is employed on two automotive use cases for brake-by-wire  
214 and an adjustable speed limiter. The Simulink models are first reduced by  
215 the flattening procedure. Then, each block is replaced by an equivalent timed  
216 automaton composed of three locations: start, offset, and operate. Still, their  
217 approach does not implement complex real-valued blocks in UPPAAL-SMC  
218 but addresses them in Simulink instead.

219 Instead of commercial modeling tools, System Modeling Language (SysML)  
220 [54] can be used to specify CPS. SysML is the defacto standard modeling lan-  
221 guage for systems engineering with rich semantics and expressive power suffi-  
222 cient to describe system structures and behaviors at various levels of abstrac-  
223 tion [25]. Ouchani et al. [45] constructed probabilistic automata by converting  
224 SysML models. The resulting models were incurred to analyze security prop-  
225 erties of the real-time streaming protocol using the probabilistic model checker  
226 PRISM [31].

227 Compared to the studied initiatives, the main objective of this work is to  
228 develop a framework that enables efficient modeling and analysis for CPS. The  
229 proposed framework takes system behavior specified using SysML diagrams  
230 as input. The novelty of this proposed work is summarized by the following  
231 contributions.

- 232 – Defining a bounded set of SysML constructs that are sufficient to capture  
233 the behaviors of the CPS discrete-time and continuous-time dynamics.
- 234 – Defining textual specification language for SysML by extending the seman-  
235 tics initially developed in [17, 45].

- 236 – Proposing a novel systematic procedure to transform the SysML behavioral  
237 specifications into PTA. Compared with the previous works that processed  
238 models specified in the commercial tool Simulink [28, 40, 18] or did not  
239 support modeling physical processes [18, 45], this new proposed approach  
240 defines a systematic procedure to process SysML models for the CPS and  
241 to construct an equivalent PTA model for analysis by supporting more  
242 features and expressive powers to specify physical properties like *time*, *rate*  
243 and *real-numbers related measurements*.
- 244 – The soundness of the proposed approach has been proven and its effective-  
245 ness to analyze CPS is demonstrated on an artificial pancreas system.

### 246 3 The Proposed Framework

247 Fig. 1 provides a brief overview of the proposed framework that runs on the  
248 following steps.

- 249 ① The process starts with the initial identification of the CPS to explore the  
250 nature of its application. This step helps specify the system's requirements  
251 including the safety properties that have to be met.
- 252 ② The topology of the system is defined by specifying the functional compo-  
253 nents of the system which are used to create the SysML block definition  
254 diagram. Also, the interactions between the CPS components are used to  
255 define the SysML flow diagram. Integrated CPS are formed from continu-  
256 ous real-time components describing physical processes and discrete-time  
257 components describing cyber processes.
- 258 ③ By relying on the existing topologies, behavioral models for the physical  
259 components are imported in the form of Ordinary-Differential Equations  
260 (ODE). Similarly, the cyber components of the system are imported from  
261 design specifications in the form of discrete variations.
- 262 ④ Physical and cyber components are represented using SysML parametric  
263 constraint diagrams and activity diagrams, respectively.
- 264 ⑤ To automate further processing, each of the SysML diagrams are written in  
265 textual format. For a constraint diagram describing the physical dynamics,  
266 the representation is done using the proposed syntax named Ordinary-  
267 Differential Equations of SysML Constraint Diagram (ODESCD). For an  
268 activity diagram describing a component's behavior, the representation is  
269 done using the proposed Enhanced Activity Calculus (EAC).
- 270 ⑥ A new systematic algorithm is proposed to convert ODESCD and EAC  
271 blocks into equivalent PTA blocks. The SysML block definition diagram,  
272 describing the system's structure, specifies the input/output connections  
273 of each PTA block.
- 274 ⑦ The various PTA blocks for physical dynamics and component behaviors  
275 are mapped as described by the flow diagram. The parallel composition of  
276 all the PTA blocks form the integrated CPS that is processed.



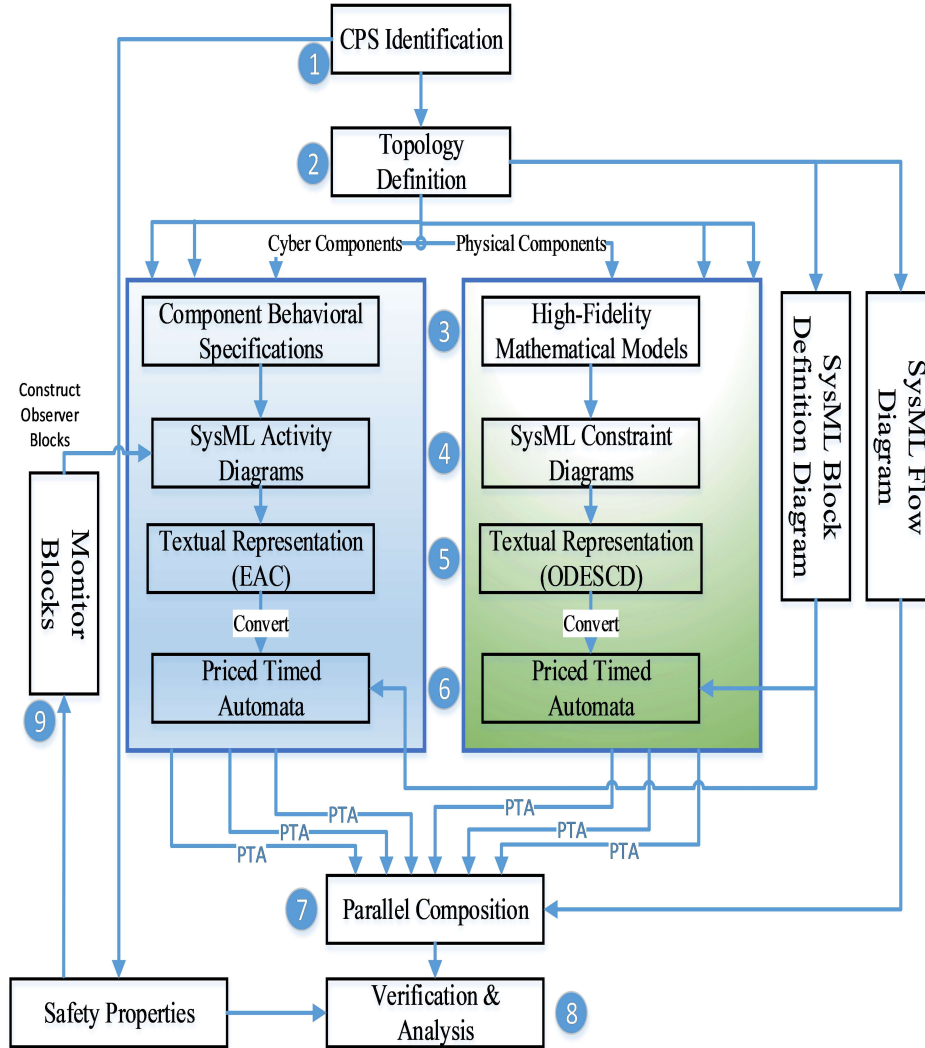


Fig. 1: The Proposed Framework Workflow.

- 277 ⑧ The analysis tool UPPAAL-SMC is used to analyze the system and verify  
 278 the safety properties. The framework is demonstrated on an artificial pan-  
 279 creas system alongside a proposed representation of continuous-time and  
 280 discrete-time dynamics.
- 281 ⑨ For safety properties that are beyond the expressive power of the query  
 282 language in use, dedicated monitor blocks are employed to observe specific

283 phenomena. An observer block is then added to the system by developing a  
 284 behavioral model for that block which is specified using a SysML activity  
 285 diagram. The new block is then processed as component of the CPS to  
 286 construct an equivalent PTA model. By adding these monitor blocks, more  
 287 complex safety properties are simplified and expressed easily in order to  
 288 be examined for safety.

### 289 3.1 SysML Graphical and Textual Modeling

#### 290 3.1.1 SysML for *continuous-time dynamics*

291 The dynamics of physical processes describe the flow of physical quantities  
 292 in the real world. These quantities are represented by real-valued real-time  
 293 variables where the derivative of a variable is equivalent to the change on  
 294 its associated physical quantity. Therefore, it is common for continuous-time  
 295 dynamics to be specified by a system of ODEs. SysML constraint diagrams  
 296 can be used to model ODEs.

297 **Notation 1** (ODE of SysML Constraint Diagram (ODESCD)). ODESCD  
 298 is defined as a tuple  $(X, X^0, K, P, R, F, I, O)$ , where:

- 299 –  $X$  is a set of real-time real-valued differentiable variables,
- 300 –  $X^0$  is a set of initial values,
- 301 –  $K$  is a set of real-valued equation coefficients,
- 302 –  $P$  is a set of constant real-valued parameters,
- 303 –  $R$  is a set of real-time real-valued variables,
- 304 –  $F(X, R)$  is a set of real-valued functions,
- 305 –  $I \in X \cup K \cup P \cup R$  is a set of input variables, and
- 306 –  $O \in X$  is a set of output variables.

**Definition 1** (Semantics of ODESCD). Let  $(X, X^0, K, P, R, F, I, O)$  be  
 a ODESCD, its semantics is defined as the dynamics of a physical system  
 described by a set of ODEs as follows (in this context a subscript in the form  
 of  $a_1 \times a_2$  indicates the matrix dimensions).

$$X'_{n \times 1}(t) = K_{n \times n}(X, P, R, t) X_{n \times 1} + F_{n \times 1}(X, R) \quad (1)$$

$$X_{n \times 1}(t = 0) = X^0_{n \times 1} \quad (2)$$

307  $X'_{n \times 1} = [x_1 \ x_2 \ \dots \ x_n]$  is the set of differential variables to be solved,  
 308  $X^0_{n \times 1} = [x_1^0 \ x_2^0 \ \dots \ x_n^0]$  is the set of initial values for the differential variables,  
 309  $K_{n \times n}(X, P, R, t)$  is the set of differential equation coefficients which can be  
 310 constants or functions of constant parameters, real-time variables or time,  $P$   
 311 is the set of additional constant parameters for the equation,  $R$  is the set of  
 312 additional real-time variables,  $F_{n \times 1}(X, R)$  is the additional terms of the ODE,  
 313  $I \in X \cup K \cup P \cup R$  is the set of input variables which can be parameters or  
 314 real-time variables, and  $O \in X$  is the set of output variables which is a subset  
 315 of the ODE solution.

316 In this system,  $I$  is **defined** to utilize variables and parameters that are  
 317 **provided as** input to the ODESCD definition, and  $O$  is used to export the  
 318 desired variables from the solution of ODESCD.

❖ **ODESCD example: meal glucose absorption model**

$X$  is a vector representing carbo-hydrate measures in the stomach where  $Q_{sto1}$  and  $Q_{sto2}$  are the stomach glucose amounts in solid state and liquid state, respectively, and  $rag$  is the blood glucose rate of appearance. These physical quantities are initially nulled as assigned in  $X^0$ . Fig. 2 depicts the SysML constraints block diagram for meal absorption variations measures.

$$X = [Q_{sto1} \ Q_{sto2} \ rag]^T \quad (3)$$

$$X^0 = [0 \ 0 \ 0]^T \quad (4)$$

$$[K] = \begin{bmatrix} -k_{gri} & 0 & 0 \\ k_{gri} & -k_{empt}(Q_{sto1}(t) + Q_{sto2}(t), D_{meal}) & 0 \\ 0 & \frac{f \cdot k_{abs}}{BW} k_{empt}(Q_{sto1}(t) + Q_{sto2}(t), D_{meal}) & -k_{abs} \end{bmatrix}$$

$$k_{empt}(Q, D_{meal}) = \begin{cases} k_{min} + \frac{k_{max} - k_{min}}{2} (\tanh(\alpha(Q - b \cdot D_{meal})) - \tanh(\beta(Q - c \cdot D_{meal})) + 2) & D_{meal} > 0 \\ 0 & D_{meal} = 0 \end{cases}$$

$$\alpha = \frac{5}{2 \cdot D_{meal} \cdot (1 - b)} \quad (5)$$

$$\beta = \frac{5}{2 \cdot D_{meal} \cdot c} \quad (6)$$

$$P = \{k_{gri}, k_{abs}, f, BW, b, c, k_{min}, k_{max}\} \quad (7)$$

$$R = \{cur\_Meal, D_{meal}\} \quad (8)$$

$$F(X, R) = [cur\_Meal(t) \ 0 \ 0]^T \quad (9)$$

$$I = \{cur\_Meal, D_{meal}\} \quad (10)$$

$$O = \{rag\} \quad (11)$$

❖ **ODESCD example: glucose-insulin dynamics**

$X$  is a vector representing the various physical quantities for the glucose and insulin dynamics all over the body compartments.  $I_{sc1}$  and  $I_{sc2}$  are the insulin levels in the subcutaneous tissues,  $X_1$  is the insulin in the interstitial fluid,  $\{G, G_s, G_t\}$  are the glucose levels in the blood, subcutaneous tissues, and slowly equilibrating tissues respectively.  $I_p$  is the plasma insulin,  $I_l$  is the portal vein insulin, and  $I_d$  is the delayed insulin signal. These physical quantities are initialized as in the vector  $X^0$ . Fig. 3 depicts the SysML Constraint Block diagrams for Glucose-Insulin variations measures.

$$X = [I_{sc1} \ I_{sc2} \ X_1 \ G_s \ I_1 \ I_d \ I_l \ I_p \ G \ G_t]^T \quad (12)$$

$$X^0 = [I_{sc1_{ss}} \ I_{sc2_{ss}} \ 0 \ G_i \ I_b \ I_b \ I_b \ I_{pb} \ G_i \ G_{ti}]^T \quad (13)$$

«constraint» Meal Absorption : Equality
$Q_{sto1}' == -k_{gri} \cdot Q_{sto1} + cur\_Meal(t)$ $Q_{sto2}' == -k_{empt}(Q_{sto1}+Q_{sto2}, D_{meal}) \cdot Q_{sto2} + k_{gri} \cdot Q_{sto1}$ $rag' == -k_{abs} \cdot rag + f \cdot k_{abs} \cdot k_{empt}(Q_{sto1}+Q_{sto2}, D_{meal}) \cdot Q_{sto2} / BW$
<p><i>Parameters:</i>            Input: cur_Meal [mg/min] (real time glucose intake) , D<sub>meal</sub> [g] (meal carbs)            Output: rag [mg/Kg/min] (real time glucose rate of appearance in the blood)</p>

Fig. 2: SysML Constraint Block for Meal Absorption

$$[K] = \begin{bmatrix} -(k_d + k_{a1}) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ k_d & -k_{a2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -p_{2u} & 0 & 0 & 0 & 0 & \frac{p_{2u}}{V_I} & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{T_s} & 0 & 0 & 0 & 0 & \frac{1}{T_s} & 0 & 0 \\ 0 & 0 & 0 & 0 & -k_i & 0 & 0 & \frac{k_i}{V_I} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & k_i & -k_i & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -(m_1 + \frac{m_6 \cdot m_1}{1 - m_6}) & m_2 & 0 & 0 & 0 \\ k_{a1} & k_{a2} & 0 & 0 & 0 & 0 & m_1 & -(m_2 + m_4) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-k_{p3}}{V_G} & 0 & 0 & 0 & -(k_{p2} + k_1) \frac{k_2}{V_G} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & K_1 \cdot V_G & -k_2 \end{bmatrix}$$

$$P = \{I_{sc1_{ss}}, I_{sc2_{ss}}, G_i, I_b, I_{p_b}, G_{t_b}, G_b, k_d, k_{a1}, k_{a2}, p_{2u}, V_I, I_b, T_s, k_i, V_G, m_1, m_6, m_2, m_4, k_{p1}, k_{p2}, k_{p3}, F_{cns}, k_{e1}, k_{e2}, k_1, k_2, V_{m0}, V_{mx}, K_{m0}, K_{mx}\} \quad (14)$$

$$R = \{rag, IIR\} \quad (15)$$

$$F(X, R) = [IIR, 0, -p_{2u} \cdot I_b, 0, 0, 0, 0, 0, 0, \frac{rag + k_{p1} - F_{cns}}{V_G}, -k_{e1} \cdot \max(0, G - \frac{k_{e2}}{V_G}), -\frac{(V_{m0} + V_{mx} \cdot X_1) \cdot G_t}{K_{m0} + K_{mx} \cdot X_1 + G_t}]^T \quad (16)$$

$$I = \{rag, IIR\} \quad (17)$$

$$O = \{G_s, G\} \quad (18)$$

«constraint» Glucose-Insulin Dynamics : Equality
$I_{sc1}' = -(k_d + k_{a1}) \cdot I_{sc1} + IIR$ $I_{sc2}' = k_d \cdot I_{sc1} - k_{a2} \cdot I_{sc2}$ $X_1' = -p_{2u} \cdot X_1 + p_{2u} \cdot (I_p / V_1 - I_b)$ $G_s' = -(G_s - G) / T_s$ $I_1' = -k_i \cdot (I_1 - I_p / V_i)$ $I_d' = -k_i \cdot (I_d - I_1)$ $I_l' = -(m_1 + (m_6 \cdot m_1 / (1 - m_6))) \cdot I_l + m_2 \cdot I_p$ $I_p' = -(m_2 + m_4) \cdot I_p + m_1 \cdot I_l + k_{a1} \cdot I_{sc1} + k_{a2} \cdot I_{sc2}$ $G' = (k_{p1} - k_{p2} \cdot G \cdot V_G - k_{p3} \cdot I_d - F_{cns} - k_{e1} \cdot \max(0, G \cdot V_G - k_{e2}) - k_1 \cdot G \cdot V_G + k_2 \cdot G_t + rag) / V_G$ $G_t' = -(G_t \cdot (V_{m0} + V_{mx} \cdot X_1)) / (K_{m0} + K_{mx} \cdot X + G_t) + k_1 \cdot G \cdot V_G - k_2 \cdot G_t$
<p><i>Parameters:</i></p> <p>Input: IIR [pmol/Kg/min] (subcutaneous insulin infusion rate)</p> <p>Input: rag [mg/Kg/min] (meal glucose rate of appearance in the plasma)</p> <p>Output: <math>G_s</math> [mg/dL] (real time subcutaneous glucose level)</p> <p>Output: <math>G</math> [mg/dL] (real time blood glucose level)</p>

Fig. 3: SysML Constraint Block for Glucose-Insulin Dynamics




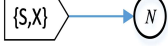
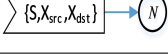



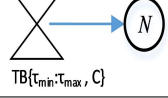

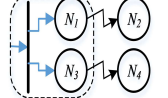
### 3.1.2 SysML for discrete-time dynamics

Discrete-time dynamics are described by SysML activity diagrams. So, **in order to precisely describe CPS and capture exactly its underlying semantics**, we develop Enhanced Activity Calculus (EAC) to formally describe SysML activity diagrams by extending NuAC presented in [17, 45]. These enhancements include redefining existing nodes as well as proposing new nodes for time-bounded delay, constraint-bounded delay, and competing events. The list of the used activity nodes and their textual EAC representation is shown in Table 1.

#### ❖ EAC example: artificial pancreas

The artificial pancreas is composed of a sensor (Fig. 4) that periodically measures the glucose level, sends it over wireless channel (Fig. 5) to the controller. **Then**, the controller (Fig. 6) calculates the required amount of insulin, and the actuator (Fig. 7) applies the control action. Lastly, the SysML activity diagram describing the meal scenario is shown in Fig. 8.

Table 1: SysML Enhanced Activity Calculus Nodes Syntax.

SysML Term	SysML Activity Diagram Structure	EAC Syntax
Activity Initial Node		$l \mapsto N$
Action Node		$l : ACT(A) \mapsto N$
Call Procedure		$l : CALL_P(A) \mapsto N$
Send Node		$l : \{S, X\}! \mapsto N$
Receive Node		$l : \{S, X_{src}, X_{dst}\} ? \mapsto N$
Merge Node		$l : Mrg \mapsto N$
Guarded Branch		$l : BC(l_{i_1} : (C = C_1) \mapsto N_1, \\ l_{i_2} : (C = C_2) \mapsto N_2, \dots)$
Probabilistic Branch		$l : BP(l_{i_1} : (P = P_1) \mapsto N_1, \\ l_{i_2} : (P = P_2) \mapsto N_2, \dots)$
Time-Bounded Delay Node		$l : DTB(\tau_{min} : \tau_{max}, C) \mapsto N$
Constraint-Bounded Delay Node		$l : DCB(C_{ter}, C) \mapsto N$
Competing Events		$l : Comp\_Events(N_1 \mapsto N_2, \\ N_3 \mapsto N_4, \dots)$

By substituting the SysML nodes with their textual equivalents following Table 1, the EAC representation of these activity diagrams is shown below.

$$Act\_Sensor = l \mapsto l_1 : Mrg \mapsto N_1$$

$$N_1 = l_2 : DTB(T_p) \mapsto l_3 : ACT(meas\_var = phy\_var) \mapsto l_4 : \{S_{et}, meas\_var\}! \mapsto l_1$$

$$Act\_Channel_{lossy} = l \mapsto l_1 : Mrg \mapsto N_1$$

$$N_1 = l_2 : \{S_{et1}, var\_in, var\_out\} ? \mapsto l_3 : BP(l_4 : (P = P_S) \mapsto N_2, l_5 : (P = P_F) \mapsto l_6 : Mrg \mapsto l_1)$$

$$N_2 = l_7 : \{S_{et2}, var\_out\}! \mapsto l_6$$

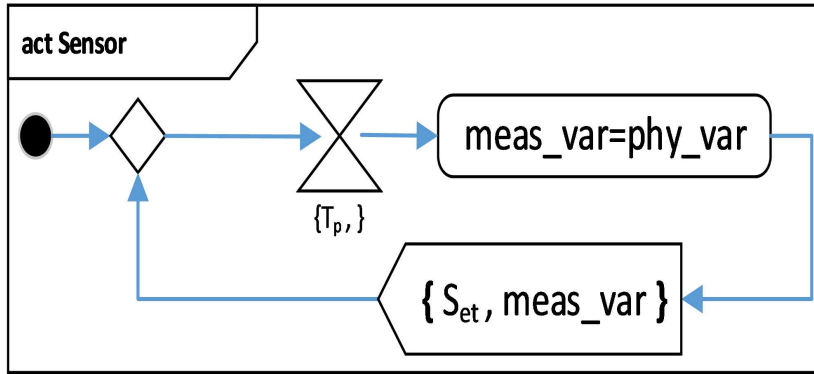


Fig. 4: SysML Activity Diagram of the Sensor.

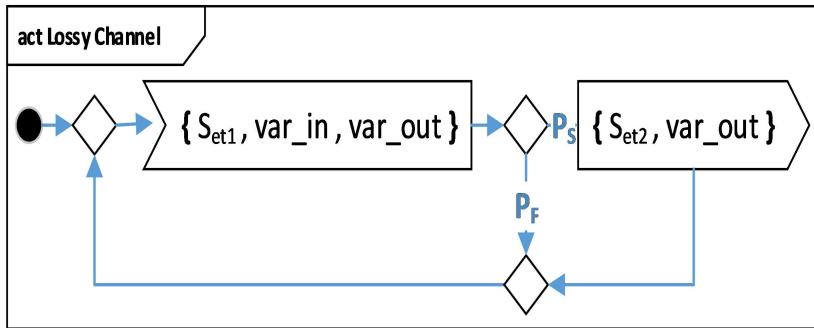


Fig. 5: SysML Activity Diagram of the Lossy Channel.

$$Act\_Ctrl = l \mapsto l_1 : Mrg \mapsto N_1$$

$$N_1 = l_2 : Comp\_Events(l_3 : \{S_{et1}, G, G_r\}? \mapsto N_2, l_4 : D_{TB}(Tp,) \mapsto N_3)$$

$$N_2 = l_5 : CALL_P(IIR = Act\_Calc\_IIR(t)) \mapsto l_6 : Mrg \mapsto N_4$$

$$N_3 = l_7 : CALL_P(IIR = Act\_Calc\_IIR\_missing(t)) \mapsto l_6$$

$$N_4 = l_8 : \{S_{et2}, IIR\}! \mapsto l_1$$

$$Act\_Actuator = l \mapsto l_1 : Mrg \mapsto l_2 : \{S_{et}, IIR_c, IIR_r\}? \mapsto l_3 : ACT(IIR = IIR_r) \mapsto l_1$$

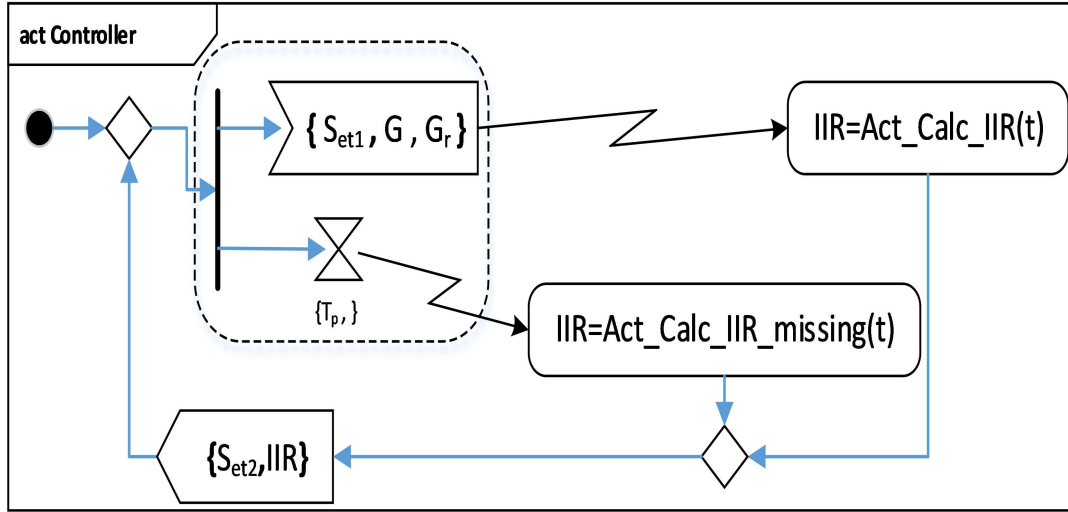


Fig. 6: SysML Activity Diagram of the Controller.

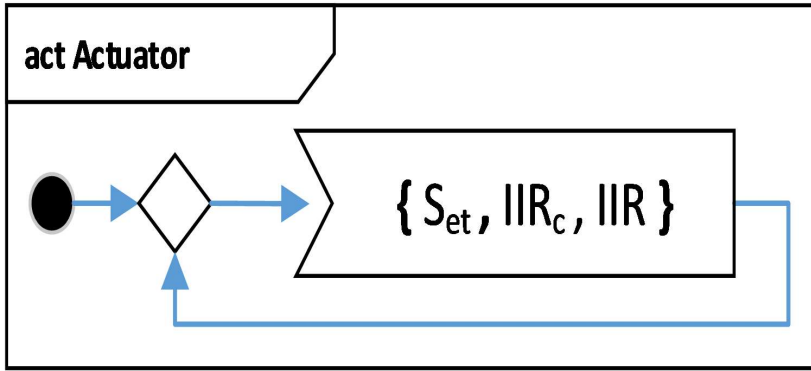


Fig. 7: SysML Activity Diagram of the Actuator.

$$Act\_meal\_scenario = l \mapsto l_1 : Mrg \mapsto l_2 : D_{TB}(inter\_meal\_time,) \mapsto N_1$$

$$N_1 = l_3 : ACT(cur\_meal = 1000 * meal\_carbs/meal\_dur, D\_meal = (Q_{sto1} + Q_{sto2})/1000) \mapsto N_2$$

$$N_2 = l_4 : D_{TB}(meal\_dur, D\_meal' == cur\_meal/1000) \mapsto l_5 : ACT(cur\_meal = 0) \mapsto l_1$$

#### ❖ CPS architecture and flow for artificial pancreas

334 The SysML block definition diagram shown in Fig. 9 defines the blocks and  
 335 their input/output ports. Also, the mapping of the blocks and [the](#) variables  
 336



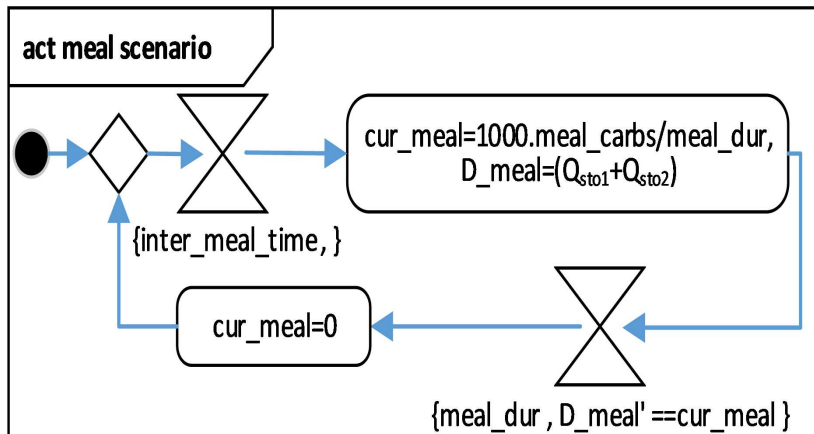


Fig. 8: SysML Activity Diagram of the Meal Scenario.

337 as well as the flow of information among these blocks are defined in the flow  
 338 internal block diagram shown in Fig. 10.

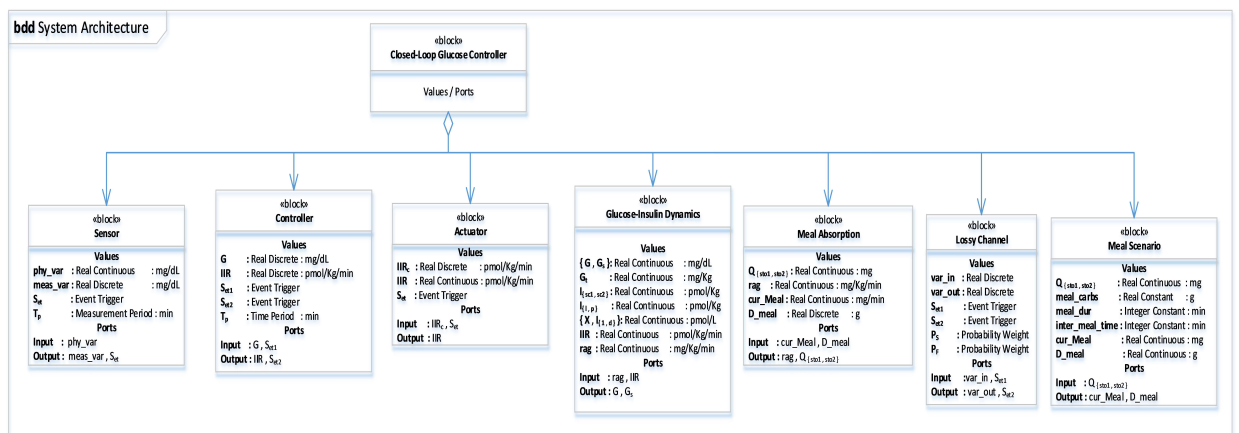


Fig. 9: SysML Architectural Block Definition Diagram of the Closed-Loop Glucose Control System.

339 **4 CPS Semantics**

340 The system behavior should be represented in the suitable formality that  
 341 matches the language of the analysis tool. To do so, the SysML components are  
 342 converted into a network of equivalent PTA models. In the following, the PTA  
 343 is defined and the new proposed automated conversion procedure is presented.

344 **Definition 2** (PTA). A PTA for CPS is a tuple  $(L, l_0, L_{lbl}, L_{IP}, L_{OP}, E, X, V_g,$   
 345  $INV(X, VAR), A(V_g), G(X, V_g), S_{et}, P_r)$ , where:

- 346 –  $L$  is a finite set of locations,
- 347 –  $l_0 \in L$  is the initial location,
- 348 –  $L_{lbl}$  is a set of labels,
- 349 –  $L_{IP}$  is a finite set of input ports,
- 350 –  $L_{OP}$  is a finite set of output ports,
- 351 –  $E$  is a finite set of edges,
- 352 –  $X$  is a finite set of clocks,
- 353 –  $VAR$  is a finite set of general-type variables,
- 354 –  $INV(X, V_g)$  is a finite set of invariants over PTA clocks  $X$  and variables  
 355  $V_g$ ,  $A(V_g)$  is a finite set of actions on the variables  $V_g$ ,
- 356 –  $G(X, V_g)$  is a finite set of atomic propositions on PTA clocks  $X$  and vari-  
 357 ables  $V_g$ ,  $S_{et}$  is a finite set of synchronization event triggers, and
- 358 –  $P_r$  is a finite set of probabilistic weights.

359 **Definition 3** (Semantics of CPS). Let  $(L, l_0, L_{lbl}, L_{IP}, L_{OP}, E, X, V_g,$   
 360  $INV(X, VAR), A(V_g), G(X, V_g), S_{et}, P_r)$  be a PTA for CPS. The semantics  
 361 are defined as a hybrid transition system composed of a set of locations  $L$   
 362 interconnected by a set of edges  $E$  through sets of input ports  $IP$  and output  
 363 ports  $OP$ , where:

- 364 – Locations  $L = \{l_1, l_2, \dots, l_{n1}\}$ , where the  $i^{th}$  location  $l_i \in L$  labelled  
 365  $label_i \in L_{lbl}$  having the invariant constraints  $inv_i \in INV$  and connected to  
 366 the input port  $x_{ip}$  and the output ports  $X_{op}$  is referred as  $l_i(label_i, inv_i, x_{ip}, X_{op})$ .
- 367 – Edges  $E = \{e_1, e_2, \dots, e_{n2}\}$ , where the  $i^{th}$  edge running the action  $a \in A$   
 368 and triggering the synchronization event  $s_{et} \in S_{et}$ , and connected to the  
 369 output port  $x_{op}$  and input port  $x_{ip}$  is referred as  $e_i = \{a, s_{et}, x_{op}, x_{ip}\}$ .
- 370 – Input ports  $L_{IP} = \{l_{ip1}, l_{ip2}, \dots, l_{ip_{n1}}\}$ , where the  $i^{th}$  input port  $l_{ip_i} \in L_{IP}$   
 371 sourcing from incoming edges  $X_e$  towards the  $i^{th}$  location  $l_i \in L$  and  
 372 applying the action  $a \in A$  is defined as  $l_{ip_i} = \{a, X_e, i\}$ .
- 373 – Output ports  $L_{OP} = \{l_{op1}, l_{op2}, \dots, l_{op_{n3}}\}$ , where the  $k^{th}$  output port  $l_{op_k} \in$   
 374  $L_{OP}$  sourcing from the  $i^{th}$  location  $l_i$  towards the  $j^{th}$  edge  $e_j$ , guarded by  
 375 the atomic proposition  $g \in G$ , triggered by the event trigger  $s_{et} \in S_{et}$ , and  
 376 having the probabilistic weight  $p_r \in P_r$  is defined as  $l_{op_k} = \{g, s_{et}, p_r, i, j\}$ .

377 PTAs traverse sequentially through output ports towards edges, followed  
 378 by input ports towards the next location, starting at an initial location denoted  
 379 by  $l_0$ . In the case of the PTA being at location  $l_i$ , the invariant  $inv_i$  must be  
 380 satisfied as long as the PTA is at location  $l_i$ . Similarly, an output port that has  
 381 a guard  $g$  with respect to its traversal can only be traversed if this guard  $g$  has

382 been satisfied. An output port with an event trigger  $s_{et}$  is synchronized with  
 383 another PTA, so that the output port is only traversed when it is activated by  
 384 the corresponding event trigger on the edge of the other PTA. Furthermore,  
 385 an output port can be traversed among other output ports in a probabilistic  
 386 manner by assigning a probability weight  $p_r$  to each of the possible candidates  
 387 for traversal of the output port.

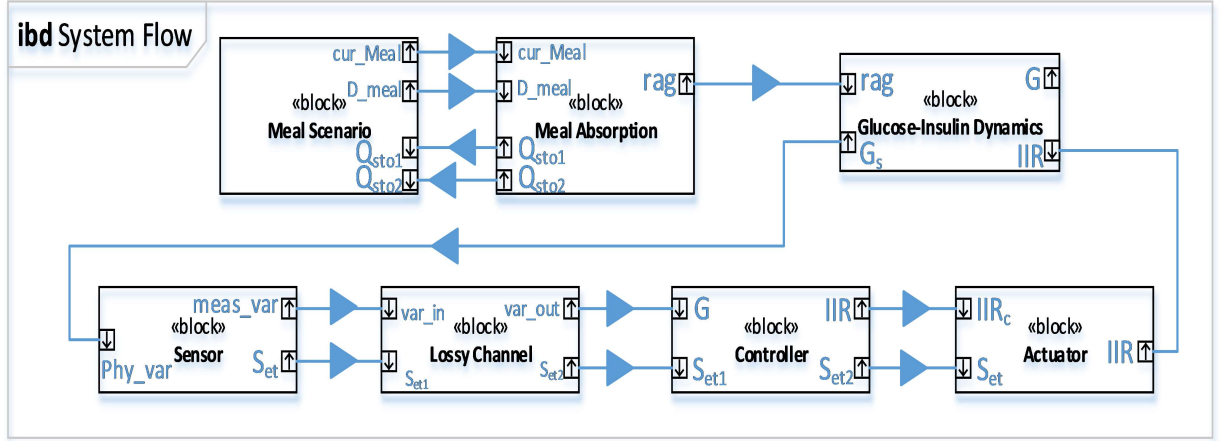


Fig. 10: SysML Flow Internal Block Diagram of the Closed-Loop Glucose Control System.

#### 388 4.1 Converting SysML into Equivalent PTA

389 In order to analyze the CPS described in SysML, it is necessary to model the  
 390 hybrid system in PTA. So, SysML blocks are translated into equivalent PTA  
 391 blocks which are parallel-composed to construct the hybrid system's global  
 392 behavior. The synchronization of actions and the transfer of values are specified  
 393 using shared variables.

394 The template of each PTA is instantiated with its input/output parameters  
 395 properly defined. The SysML flow internal block diagram (as in Fig. 10)  
 396 is consulted to define global variables for the parameters connecting the PTA  
 397 components of the system. When instantiating a PTA template, the parameters  
 398 are passed by-reference except for constant parameters that are passed  
 399 by-value. Instead, those constants can be defined as local variables in the PTA.  
 400 The following rules govern the definition of variables in PTA models.

- 401 – Continuous real-valued parameters are defined using clock variables.

- 402 – Discrete real-valued parameters are defined using floating point variables.  
 403 – An event trigger should be activated whenever a discrete variable is up-  
 404 dated, so that the other PTAs are notified about the new update.  
 405 – Discrete integer parameters are defined as integer variables and are passed  
 406 between PTAs similar to the floating point variables.  
 407 – When assigning or initializing a numerical variable, it can be evaluated to  
 408 a single value or to a range of values for a uniformly-distributed random  
 409 assignment.

410 ❖ **Converting SysML EAC into PTA**

411 This part presents the detailed procedure for constructing a PTA block that  
 412 represents a SysML EAC block. Alongside the description of the conversion  
 413 steps, an illustrative example is provided for converting the  $Act\_Channel_{lossy}$   
 414 block from EAC into PTA.

- 415 – The first step is to merge all the EAC nodes into the main EAC con-  
 416 struct. This is done by iterating through the auxiliary constructs ( $N_x$ ) and  
 substituting for them in the main construct as depicted in Fig. 11.

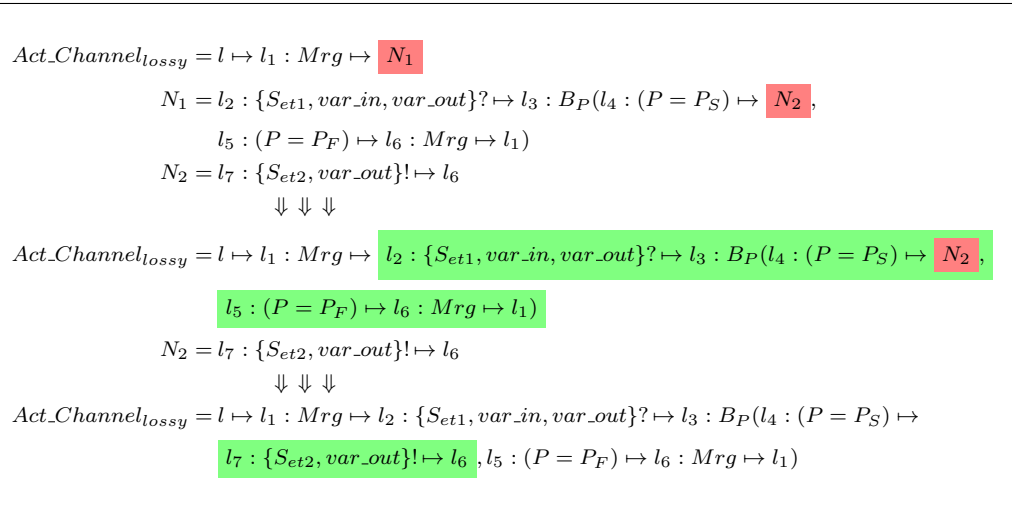


Fig. 11: EAC Lossy Channel Example - Merging Nodes

- 417 – Connecting the EAC terms so that each arrow is uniquely identified as  
 418 presented in Fig. 12.  
 419 – Handling branching terms ( $B_P$  or  $Comp\_Events$ ) and replicating the EAC  
 420 construct, so that each branching term has only one path at a time. This  
 421 is done by iterating through the branching terms and taking one branch  
 422 at a time as shown in Fig. 13.  
 423 – Building the PTA skeleton using the procedure described in Algorithm 1.  
 424 The resulting skeleton for  $Act\_Channel_{lossy}$  example is shown in Fig. 14.  
 425

$$\begin{array}{c}
\Downarrow \Downarrow \Downarrow \\
Act\_Channel_{lossy} = l \xrightarrow{1} l_1 : Mrg \xrightarrow{2} l_2 : \{S_{et1}, var\_in, var\_out\} ? \xrightarrow{3} l_3 : B_P(l_4 : (P = P_S) \xrightarrow{4} \\
l_7 : \{S_{et2}, var\_out\} ! \xrightarrow{5} l_6, l_5 : (P = P_F) \xrightarrow{6} l_6 : Mrg \xrightarrow{7} l_1)
\end{array}$$

Fig. 12: EAC Lossy Channel Example - Labeling Arrows

$$\begin{array}{c}
Act\_Channel_{lossy} = l \xrightarrow{1} l_1 : Mrg \xrightarrow{2} l_2 : \{S_{et1}, var\_in, var\_out\} ? \xrightarrow{3} l_3 : B_P( l_4 : (P = P_S) \xrightarrow{4} \\
l_7 : \{S_{et2}, var\_out\} ! \xrightarrow{5} l_6, l_5 : (P = P_F) \xrightarrow{6} l_6 : Mrg \xrightarrow{7} l_1 ) \\
Act\_Channel_{lossy} - Path(1) = l \xrightarrow{1} l_1 : Mrg \xrightarrow{2} l_2 : \{S_{et1}, var\_in, var\_out\} ? \xrightarrow{3} l_4 : B_P - Branch ( l_4 : \\
(P = P_S) \xrightarrow{4} l_7 : \{S_{et2}, var\_out\} ! \xrightarrow{5} l_6 ) \\
\Downarrow \Downarrow \Downarrow \\
Act\_Channel_{lossy} - Path(2) = l \xrightarrow{1} l_1 : Mrg \xrightarrow{2} l_2 : \{S_{et1}, var\_in, var\_out\} ? \xrightarrow{3} l_3 : B_P - Branch ( l_5 : \\
(P = P_F) \xrightarrow{6} l_6 : Mrg \xrightarrow{7} l_1 )
\end{array}$$

Fig. 13: EAC Lossy Channel Example - Branches Handling

- 426 – For each location node that has non-empty *prev* field, insert an input port.  
427 For locations with *next* field, insert an output port per edge node that is  
428 outgoing from the location. In the following steps, when an EAC term is  
429 linked to an output port, the one that is connected to the location where  
430 the EAC belongs is identified. In case the location is attached to two or  
431 more output ports, the sequence of EAC terms in the path construct is  
432 used to identify the corresponding output port. Moreover, an EAC node  
433 that shows up in more than one path is only converted once at its first  
434 appearance.
- 435 – Replacing the following EAC terms with their equivalent PTA terms.
- 436 – EAC term  $l$  signifies the location as an initial location.
  - 437 –  $D_{TB}(\tau_{min} : \tau_{max}, C)$ : Declare a clock variable  $t$ , Add a reset for the  
438 clock ( $t = 0$ ) to the input port action, Add the following constraint  
439 ( $t \leq \tau_{max} \ \&\& \ C$ ) to the invariants *inv* of the location, and add the  
440 following ( $t \geq \tau_{min}$ ) to the guard  $g$  of the output port.
  - 441 –  $\{S, X_{src}, X_{dst}\}?$ : Add the event trigger  $S?$  to the respective field  $s_{et}$   
442 of the output port, and add the assignment ( $X_{dst} = X_{src}$ ) to the action  
443 of the edge outgoing from the output port.

**Algorithm 1** Construction of PTA Skeleton.

---

```

for each: EAC_Path
1: prev_Node =  $\emptyset$  ▷ The first node of a path has no predecessor.
for each: EAC_Node  $\in$  EAC_Path

2: if EAC_Node  $\in$  {Mrg, Comp_Events, BP, D(*), {*,*}?, (P = *)} then
3:   EAC_Type = LOCATION
4: else if EAC_Node  $\in$  { $\mapsto$ , ACT, CALLP, {*,*}!} then
5:   EAC_Type = EDGE
6: end if

7: if EAC_Node processed before then
8:   cur_Node = PTA_Node[EAC_Node] ▷ Traverse through the node.
9:   cur_Node.prev.addMember(prev_Node) ▷ Create a new input port for the node.
10:  prev_Node.next.addMember(cur_Node) ▷ Create a new output port for the node.
11: else if EAC_Type == prev_Node.type then
12:  cur_Node.EAC.addMember(EAC_Node) ▷ A compliment for the previous node.
13: else ▷ A node not processed yet.
14:  cur_Node = create_Node(type = EAC_Type) ▷ Create the node.
15:  cur_Node.EAC.addMember(EAC_Node) ▷ Traverse through the node.
16:  cur_Node.prev.addMember(prev_Node) ▷ Create an input port.
17:  prev_Node.next.addMember(cur_Node) ▷ Create an output port.
18: end if

```

---

$Node_{ID}$	= [	<i>type</i> ,	<i>prev</i> ,	<i>next</i> ,	<i>EAC</i>	]
$Node_1$	= [	<i>LOCATION</i> ,	$\emptyset$ ,	2,	$l$	]
$Node_2$	= [	<i>EDGE</i> ,	1,	3,	$\xrightarrow{1}$	]
$Node_3$	= [	<i>LOCATION</i> ,	{2, 11},	4,	$l_1$	]
$Node_4$	= [	<i>EDGE</i> ,	3,	5,	$\xrightarrow{2}$	]
$Node_5$	= [	<i>LOCATION</i> ,	4,	6,	$l_2$	]
$Node_6$	= [	<i>EDGE</i> ,	5,	7,	$\xrightarrow{3}$	]
$Node_7$	= [	<i>LOCATION</i> ,	6,	{8, 10},	{ $l_3, l_4, l_5$ }	]
$Node_8$	= [	<i>EDGE</i> ,	7,	9,	{ $\xrightarrow{4}, l_7, \xrightarrow{5}$ }	]
$Node_9$	= [	<i>LOCATION</i> ,	{8, 10},	11,	$l_6$	]
$Node_{10}$	= [	<i>EDGE</i> ,	7,	9,	$\xrightarrow{6}$	]
$Node_{11}$	= [	<i>EDGE</i> ,	9,	3,	$\xrightarrow{7}$	]

Fig. 14: EAC Lossy Channel Example - Building Skeleton.

- 444 –  $\{S, X\}!$ : Add this event trigger  $S!$  to the respective event trigger field  
 445  $s_{et}$  of the containing edge.  
 446 –  $(P = p_x)$ : Add the following probabilistic weight to the corresponding  
 447 field  $p_r$  of the output port.  
 448 –  $ACT(A)$ : Add the action  $A$  to the corresponding field  $a$  of the edge.  
 449 –  $CALL_P(A)$ : Add the behavior call  $A()$  to the action field  $a$  of the edge.  
 450 The results shown in Fig. 15 are obtained when applying the above rules  
 on the  $Act\_Channel_{lossy}$  example:

$loc_1(label_1, \phi, \phi, op_1)$   
 $loc_2(label_2, \phi, ip_2, op_2)$   
 $loc_3(label_3, \phi, ip_3, op_3)$   
 $loc_4(label_4, \phi, ip_4, \{op_{(4,1)}, op_{(4,2)}\})$   
 $loc_5(label_5, \phi, ip_5, op_5)$   
 $ip_2(\phi, \{e_1, e_6\}, loc_2)$   
 $ip_3(\phi, e_2, loc_3)$   
 $ip_4(\phi, e_3, loc_4)$   
 $ip_5(\phi, \{e_4, e_5\}, loc_5)$   
 $op_1(\phi, \phi, \phi, loc_1, e_1)$   
 $op_2(\phi, \phi, \phi, loc_2, e_2)$   
 $op_3(\phi, Set_1?, \phi, loc_3, e_3)$   
 $op_{(4,1)}(\phi, \phi, P_S, loc_4, e_4)$   
 $op_{(4,2)}(\phi, \phi, P_F, loc_4, e_5)$   
 $op_5(\phi, \phi, \phi, loc_5, e_6)$   
 $e_1(\phi, \phi, op_1, ip_2)$   
 $e_2(\phi, \phi, op_2, ip_3)$   
 $e_3(var_{out} = var_{in}, \phi, op_3, ip_4)$   
 $e_4(\phi, Set_2!, op_{(4,1)}, ip_5)$   
 $e_5(\phi, \phi, op_{(4,2)}, ip_5)$   
 $e_6(\phi, \phi, op_5, ip_2)$

Fig. 15: EAC Lossy Channel Example - Replacing EAC with PTA Terms.

- 451 – After each EAC receive node, insert a new location between the event  
 452 trigger and the signal sampling. Also, a new location is added when an  
 453 output port with a probabilistic weight is directly followed by an edge  
 454 with an EAC send node. This is done so that the send node is separated  
 455 from the output port. When applying this on the  $Act\_Channel_{lossy}$ , the  
 456 results look like Fig. 16  
 457

```

loc1(label1, φ, φ, op1)
loc2(label2, φ, ip2, op2)
loc3(label3, φ, ip3, op3)
loc4(label4, φ, ip4, {op(4,1), op(4,2)})
loc5(label5, φ, ip5, op5)
loc6(label6, φ, ip6, op6)
loc7(label7, φ, ip7, op7)
ip2(φ, {e1, e6}, loc2)
ip3(φ, e2, loc3)
ip4(φ, e3, loc4)
ip5(φ, {e8, e5}, loc5)
ip6(φ, e7, loc6)
ip7(φ, e4, loc7)
op1(φ, φ, φ, loc1, e1)
op2(φ, φ, φ, loc2, e2)
op3(φ, Set1?, φ, loc3, e7)
op(4,1)(φ, φ, PS, loc4, e4)
op(4,2)(φ, φ, PF, loc4, e5)
op5(φ, φ, φ, loc5, e6)
op6(φ, φ, φ, loc6, e3)
op7(φ, φ, φ, loc7, e8)
e1(φ, φ, op1, ip2)
e2(φ, φ, op2, ip3)
e3(varout = varin, φ, op6, ip4)
e4(φ, φ, op(4,1), ip7)
e5(φ, φ, op(4,2), ip5)
e6(φ, φ, op5, ip2)
e7(φ, φ, op3, ip6)
e8(φ, Set2!, op7, ip5)

```

Fig. 16: EAC Lossy Channel Example - Inserting Locations

- 458 – Divide the locations into transient and regular (time-consuming) locations.  
459 A regular location is identified by having either a guard or an event trigger  
460 on the output port, or by having a non-empty invariant field. For the  
461  $Act\_Channel_{lossy}$  example, all the locations are transient except location  
462  $loc_3$  which has an event trigger on the output port.



- 463 – The rate of all local clocks should be identified on all regular locations.  
 464 Therefore, if a clock is not supposed to evolve in a specific regular location,  
 465 its evolution rate should be assigned to 0 in the invariants field of that  
 466 location.
- 467 – When exporting the PTAs into an XML file compatible with UPPAAL-  
 468 SMC analyzer, transient locations are specified as *urgent* locations except  
 469 for the following:
- 470 – A location which emits output ports with probabilistic weights (location  
 471  $loc_4$  in  $Act\_Channel_{lossy}$  example) is defined as an *anchor* point (for  
 472 syntax compatibility).
  - 473 – The first location following a receive node (location  $loc_6$  in  $Act\_Channel_{lossy}$   
 474 example) should be set to *committed* for synchronization correctness  
 475 (semantic compatibility).

476 The resulting PTA diagram for the above transformed lossy channel is  
 477 depicted in Fig. 17. This PTA initializes at the location  $loc_1$ . This location  
 478 is urgent which means that no time progress and hence the PTA will move  
 479 instantly through the output port  $op_1$ , the edge  $e_1$ , the input port  $ip_2$  to the  
 480 next location  $loc_2$ . This location is also an urgent location and hence the PTA  
 481 will move through the output port  $op_2$ , edge  $e_2$ , and the input port  $ip_2$  towards  
 482 the location  $loc_3$ . The output port  $op_3$  is activated by the event trigger  $S_{et1}$ ?  
 483 which is controlled by another PTA (the sensor in this case). **Then, this** sensor  
 484 activates the event trigger  $S_{et}$  to send a new measurement (the variable  $var_{in}$ )  
 485 through the wireless channel. When triggered by the event trigger  $S_{et1}$ , the  
 486 lossy channel PTA moves through the output port  $op_3$ , the edge  $e_7$ , **and** the  
 487 input port  $ip_6$  towards the committed location  $loc_6$ . Like the urgent location,  
 488 a committed location freezes time but also synchronizes the PTAs so that the  
 489 correct sequence of actions takes place. In this PTA, it is required so that the  
 490 up-to-date version of the measurement value  $var_{in}$  is read.

491 The PTA moves through  $op_6$  towards the edge  $e_3$  where the measurement  
 492 is sampled, and then through the input port  $ip_4$  to the location  $loc_4$  which is a  
 493 probabilistic branching point. **Then**, the PTA will take a branch depending on  
 494 probability weights. At one branch, the message will get lost and so the PTA  
 495 takes the output port  $op_{(4,2)}$  towards the edge  $e_5$  and the input port  $ip_5$  to  
 496 reach the location  $loc_5$ . In the other branch, the measurement is successfully  
 497 relayed so the other PTA (the controller in this case) is notified with the event  
 498 trigger  $S_{et2}!$ , so the PTA moves through  $op_{(4,1)}$ ,  $e_4$ ,  $ip_7$  to the transient location  
 499  $loc_7$  towards the output port  $op_7$  and the edge  $e_8$  (where  $S_{et2}!$  is activated) to  
 500 the input port  $ip_5$  **while merging** with the other branch in the location  $loc_5$ .  
 501 **Finally**, the PTA moves via the output port  $op_5$  and the edge  $e_6$  through the  
 502 input port  $ip_2$  to merge in the location  $loc_2$ .

#### 503 ❖ modeling ODESCD using PTA

504 The same rules apply to convert ODESCD into PTA where the ODE variables  
 505  $X$  are defined as clock variables. The PTA is composed of one location where  
 506 the rates of the ODE variables  $X$  are assigned using equality constraints in  
 507 the invariant field of the main location. If some variables or parameters are

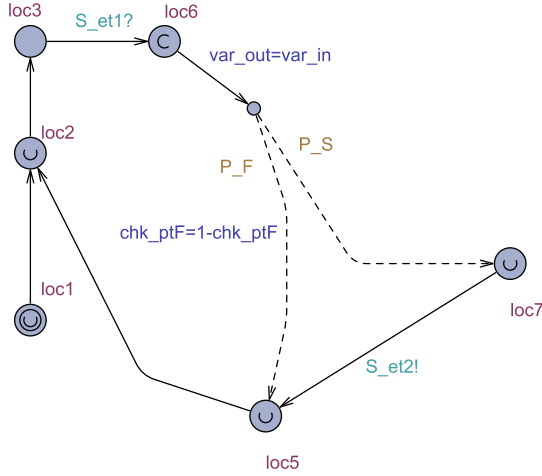


Fig. 17: The Resulting PTA Diagram for the Lossy Channel

508 initialized with random values, an additional transient initial location is added  
 509 with the variables assigned in the edge connecting the initial location to the  
 510 main operational location.

#### 511 4.2 Soundness

512 After presenting the semantics of CPS and PTA, we prove the soundness of the  
 513 developed framework. First, let  $\Gamma$  to be a function denoting Algorithm 1. Now,  
 514 we prove the soundness of the transformation by showing that  $\Gamma$  guarantees  
 515 the integrity of the CPS design, i.e. no added, modified, or excluded behavior.  
 516 Thus, an equivalent PTA behavioral model is produced. Then, we show that  
 517 the soundness proves the satisfiability preservation of MILT expressions when  
 518 applying  $\Gamma$ .

519 As depicted in Fig. 18, we have to show the nature of the relation  $\mathcal{R}$ ,  
 520 that compares both  $PTA^{cps}$  and  $PTA^f$  constructed through EAC and PTA  
 521 semantics rules respectively, while preserving both behaviours. Indeed, the  
 522 relation  $\mathcal{R}$  could be determined by comparing the semantics of each term in  
 523 EAC and the semantics of its image obtained by the function  $\Gamma$ . Since the goal  
 524 is to guarantee the behaviour integrity of  $PTA^{cps}$  and the resulting  $PTA^f$   
 525 should not differ from  $PTA^{cps}$ , Lemma 1 proves that  $\mathcal{R}$  is a bisimulation  
 526 relation.

527 **Lemma 1** *The binary relation  $\mathcal{R}$ , is a bisimulation, whenever  $S\hat{\mathcal{R}}\hat{S}$ , satisfies*  
 528 *the following.*

- 529 1. If  $S \xrightarrow{\alpha} S'$  then  $\exists \hat{S}'$  such that  $\hat{S} \xrightarrow{\alpha} \hat{S}'$  and  $S' \equiv_{\mathcal{R}} \hat{S}'$ .

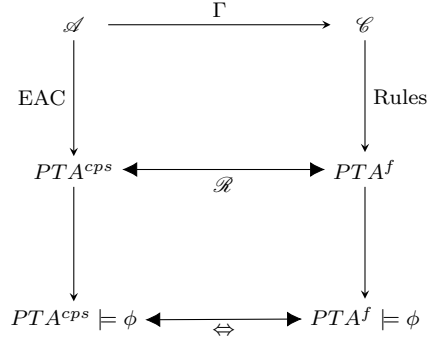


Fig. 18: The Transformation Soundness Schema.

530 2. If  $\hat{S} \xrightarrow{\alpha} \hat{S}'$  then  $\exists S$  such that  $S' \xrightarrow{\alpha} S'$  and  $\hat{S}' \equiv_{\mathcal{R}} S'$ .

531 *Proof* Let's consider  $A \in PTA^{cps}$  and  $B \in PTA^f$  where  $\Gamma(A) = B$ . So, by  
 532 induction on EAC terms, we prove that  $\mathcal{R}$  is a bisimulation binary relation as  
 533 follows.

- 534 – When  $A = i \rightarrow \mathcal{N}$ , then based on the rule  $\exists S \xrightarrow{\alpha} S'$  such that  $S = i \rightarrow \mathcal{N}$   
 535 and  $S' = i \rightarrow \mathcal{N}'$ , we will have,  $\Gamma(A) = \Gamma(i \rightarrow \mathcal{N}) = \text{initial to } i$ . Thus,  
 536  $\text{initial} \wedge \neg i \xrightarrow{\alpha} \neg \text{initial} \wedge i \in B^s$ . Then,  $PTA^{cps} \mathcal{R} PTA^f$  when  $A = i \rightarrow \mathcal{N}$ .
- 537 – For  $\{S, X\}! \rightarrow \mathcal{N}$ , then  $\overline{X} \rightarrow \mathcal{N} \rightarrow \{S, X, X'\}! \rightarrow \mathcal{N} \in PTA^{cps}$ . Also,  
 538  $\Gamma(A) = \text{resource} \langle v \rangle \rightarrow \mathcal{N}$  which means  $\text{resource}_v \wedge \neg \mathcal{N} \xrightarrow{\text{prt}} \neg \text{resource}_v \wedge$   
 539  $\mathcal{N} \in B^s$ . So,  $PTA^{cps} \mathcal{R} PTA^f$ .
- 540 – In the case of  $A = \text{resource?} v \rightarrow \mathcal{N}$ , we have  $\overline{\text{resource?} v \rightarrow \mathcal{N}} \rightarrow$   
 541  $\text{resource?} v \rightarrow \mathcal{N} \in B^{sn}$ . Thus,  $PTA^{cps} \mathcal{R} PTA^f$ .
- 542 – If  $A = \text{resource!} v \rightarrow \mathcal{N}$ , we have  $\overline{\text{resource!} v \rightarrow \mathcal{N}} \rightarrow \text{resource!} v \rightarrow$   
 543  $\mathcal{N} \in B^{sn}$   $R$   $\text{resourceout}_v \wedge \exists v \wedge \neg \mathcal{N} \xrightarrow{\text{prt}} \neg \text{resourceout}_v \wedge \mathcal{N} \in B^s$ .
- 544 – By considering  $A = \text{resource} \uparrow \text{expression} \rightarrow \mathcal{N}$ , then  $\overline{\text{resource} \uparrow \text{expression} \rightarrow \mathcal{N}} \rightarrow$   
 545  $\text{resource} \uparrow \text{expression} \rightarrow \mathcal{N} \in B^{sn}$ . As a result, we have  $\text{resource}_v \wedge$   
 546  $\neg \mathcal{N} \xrightarrow{\text{prt}} \neg \text{resource}_v \wedge v = \text{newvalue} \wedge \mathcal{N} \in B^s$ , which means  $PTA^{cps} \mathcal{R}$   
 547  $PTA^f$ .
- 548 – For the decision term  $A = D(g_{v1}, \mathcal{N}_1, \mathcal{N}_2)$ , we differentiate two cases:  
 549 1. When  $\neg g_{v1} \models \top$ , we have  $\overline{D(g_{v1}, \mathcal{N}_1, \mathcal{N}_2)} \xrightarrow{\neg g_{v1}} D(g_{v1}, \mathcal{N}_1, \mathcal{N}_2) \in B^{sn}$   
 550 by relying on the decision rule. Also, we have:  $\Gamma(A) = \{\text{on prt}_i \text{ from}$   
 551  $\text{source to } \mathcal{N} \text{ provided } g_{v_i} = \text{eval}(v_i) : i \in \{1, 2\}\}$ . Also, since  $\neg g_{v1} \models \top$ ,  
 552 we have:  $\overline{D(g_{v1}, \mathcal{N}_1, \mathcal{N}_2)} \xrightarrow{\neg g_{v1}} D(g_{v1}, \mathcal{N}_1, \mathcal{N}_2) \in B^s$ .
- 553 2. For the other case, when  $g_{v1} \models \top$ , we have shown that  $D(g_{v1}, \mathcal{N}_1, \mathcal{N}_2) \equiv$   
 554  $D(\neg g_{v1}, \mathcal{N}_2, \mathcal{N}_1)$ . Thus,  $PTA^{cps} \mathcal{R} PTA^f$ .

555 We have shown that for each EAC term, we have  $PTA^{cps} \mathcal{R} PTA^f$  in which  
 556 result that  $\mathcal{R}$ , is a bisimulation relation and it is symmetric.

557 Based on the illustration presented in Fig 18, the transformation's objec-  
 558 tive is to verify functional properties of the generated PTA model and then

559 infer satisfiability results for the CPS design. Using Lemma 1, Proposition 1  
560 demonstrates how the properties expressed in MITL logic can be satisfied.

561 **Proposition 1**  $\forall A \in PTA^{cps}, B \in PTA^f$  s.t.  $\Gamma(A) = B$ , we have:  $\forall \phi \in$   
562  $MITL : PTA^f \models \phi \implies PTA^{cps} \models \phi$ .

563 *Proof* By induction on MITL terms, we prove that  $B \models \phi \implies A \models \phi$ .

564 1. First, let's consider the state formulae  $\phi = \varphi_1 \wedge \varphi_2$  where  $B \models \phi$ . Now, we  
565 show the satisfiability of  $\phi$  on  $A$  for the following EAC terms.

566 – For  $A = i \rightarrow \mathcal{N}$ , we have  $i \rightarrow \mathcal{N} \xrightarrow{\alpha} \overline{i \rightarrow \mathcal{N}} R \text{ initial} \wedge \neg i \xrightarrow{\alpha}$   
567  $\neg \text{initial} \wedge i$ . If  $\text{initial} \wedge \neg i \models \varphi_1 \wedge \varphi_2$  means  $\text{initial} \wedge \neg i = \varphi_1 \wedge \varphi_2$ .  
568 Thus,  $i \rightarrow \mathcal{N} \models \phi$ , and,  $B \models \phi$

569 – For  $A = \text{resource} < v > \rightarrow \mathcal{N}$  when  $\neg \text{resource}_v \wedge \mathcal{N} \models \varphi_1 \wedge \varphi_2$ , we  
570 have  $\text{resource} < v > \rightarrow \overline{\mathcal{N}} \models \varphi_1 \wedge \varphi_2$ . Then,  $B \models \phi$ .

571 – For  $A = \text{resource}?v \rightarrow \mathcal{N}$ , then  $B \models \phi$   $\text{resourcein}_v \wedge v = \text{newvalue} \wedge$   
572  $\neg \mathcal{N} \xrightarrow{prt} \neg \text{resourcein}_v \wedge \mathcal{N} \models \phi$ . Thus, we have  $\overline{\text{resource}?v \rightarrow \mathcal{N}} \rightarrow$   
573  $\text{resource}?v \rightarrow \overline{\mathcal{N}} \models \phi$ . Consequently,  $B \models \phi$ .

574 2. Now, we consider the path formulae  $P_{\neq p}[\psi]$ . So, since EAC does not sup-  
575 port probabilistic decisions and has only deterministic ones,  $P_{\geq 1}[\psi]$  means  
576  $\psi$  else we consider the case of  $P_{\leq 0}[\psi]$ . Then, we prove by induction on the  
577 path operators that  $PTA^{cps} \models \phi$  when  $PTA^f \models \phi$  as follows.

578 – For  $\phi = N\varphi$ ,  $B \models \phi$  means  $\exists \hat{S} \xrightarrow{\alpha} \hat{S}' \in B^n$  such that  $\hat{S}' \models \varphi$ . In  
579 addition, since  $\mathcal{R}$  is symmetric, then  $\exists S \xrightarrow{\alpha} S' \in B$  such that:  $S' \models \varphi$ .

580 – For  $\phi = \varphi_1 \cup^t \varphi_2$ , we have  $\exists \hat{S}_1 \xrightarrow{\alpha} \dots \rightarrow \hat{S}'_t \subseteq B^n$  such that  $\hat{S}'_{i:i < t'} \models \varphi_1$   
581 and  $\hat{S}'_t \models \varphi_2$ . Also,  $\mathcal{R}$  is symmetric and  $\exists S_1 \xrightarrow{\alpha} \dots \rightarrow S'_t \subseteq B^{sn}$  where  
582  $S_i \mathcal{R} \hat{S}'_i : 0 < i \leq t$ . Thus,  $B \models \phi$ .

583 Based on the previous proof, we have shown that for each EAC and MITL  
584 term,  $\mathcal{R}$  always preserves the satisfiability of MITL formulae. Consequently,  
585  $B \models \phi \implies A \models \phi$  for all  $\phi$  expressed in MITL when  $PTA^{cps} \mathcal{R} PTA^f$ .

## 586 5 Experimentation

587 This section shows the effectiveness of the proposed framework by first vali-  
588 dating the transformation algorithm. Then, the proposed approach is used to  
589 demonstrate how the safety of the obtained model can be examined by statisti-  
590 cal model checking over a list of selected functional and safety requirements.

### 591 5.1 Validation of the Conversion Procedure

592 In order to demonstrate the correctness of the proposed approach, PTA models  
593 are validated. Properties are specified for each component of the system that  
594 constrain its functional behavior. To evaluate whether the resulting PTA model  
595 meets the behavioral properties, random simulations are conducted and trace

596 log analysis is applied to the results. The resulting PTA models are more likely  
 597 to be valid representations of the CPS components when all the properties are  
 598 satisfied.

599 By comparing the values of the ODE variables with a mathematical ODE  
 600 solver, PTAs representing ODESCD are validated. In the case of the ODE-  
 601 SCDs describing meal absorption and glucose-insulin dynamics, multiple sim-  
 602 ulations are conducted on 10 virtual patients for 24 hours under various meal  
 603 scenarios. The PTAs for these ODESCDs that are constructed using the above  
 604 automatic procedure are simulated.

605 The trace logs of the physical variables are compared against our ODE  
 606 solver developed in Matlab and errors are recorded. The absolute errors of  
 607 variable samples are divided by the variable root mean square to get the re-  
 608 lative absolute errors. The percentage mean and standard deviation (std) of  
 609 these relative absolute errors are depicted in Table 2. It can be noted that  
 610 the relative errors are negligible and hence demonstrate the correctness of the  
 611 proposed procedure.

Table 2: Meal and Glucose-Insulin Dynamics ODESCD Variables (Results Against a Mathematical Solver).

Variable Identifier	Relative Absolute Error {mean+std}
$Q_{sto1}$	0.018% $\pm$ 0.007%
$Q_{sto2}$	0.027% $\pm$ 0.012%
$rag$	0.028% $\pm$ 0.012%
$I_{sc1}$	0.166% $\pm$ 0.049%
$I_{sc2}$	0.117% $\pm$ 0.039%
$X_1$	0.219% $\pm$ 0.029%
$G_s$	0.164% $\pm$ 0.203%
$I_1$	0.071% $\pm$ 0.030%
$I_d$	0.047% $\pm$ 0.027%
$I_l$	0.118% $\pm$ 0.040%
$I_p$	0.118% $\pm$ 0.040%
$G$	0.165% $\pm$ 0.200%
$G_s$	0.180% $\pm$ 0.209%

612 For the case of cyber components which are specified by EAC, the following  
 613 steps demonstrate the model validation for this type of PTAs.

- 614 – Sensor: The sensor PTA shown in Fig. 19-a has three locations. It periodically  
 615 waits in  $loc_3$  before sampling the subcutaneous glucose measurement  
 616  $phy\_var$  into the variable  $meas\_var$ . The edge originating from  $loc_3$  to  $loc_2$   
 617 synchronizes the sensor with the lossy channel by means of the event trigger  
 618  $S_{et}$ .
- 619 – A new measurement is sent periodically every  $T_p$  minutes: to check on  
 620 this property, a new binary flag variable is added to the PTA ( $chk\_pt_1$  in  
 621 the sensor PTA shown in the graph of Fig. 19-a). The variable is marked

622 whenever a measurement is sent. This can be achieved by flipping the  
 623 value of the variable in an ACT term **at** the same edge as the send  
 624 term (the edge goes from *loc3* to *loc2*). The variable is monitored on  
 625 random simulations and its value should be flipped periodically every  
 626  $T_p$  minute.

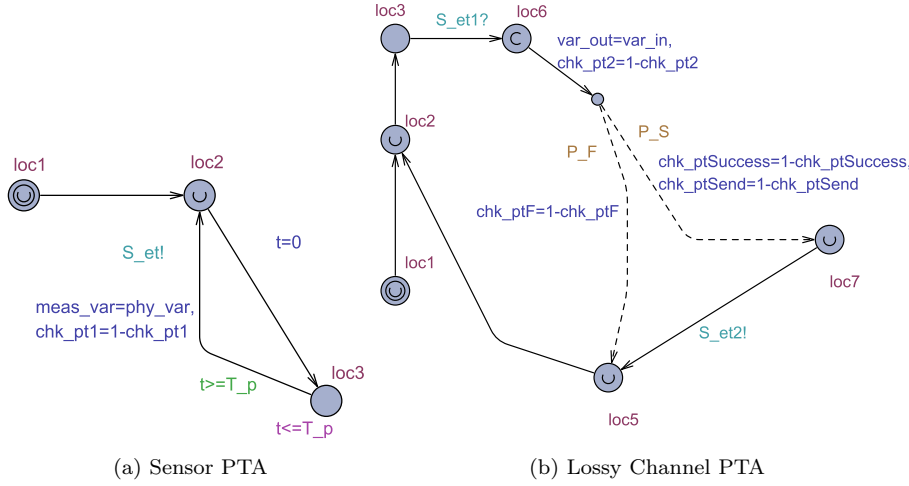


Fig. 19: A Part of the Sensor's PTA Communication Network.

- 627 – Whenever a measurement is sent, its value should be equal to the most  
 628 recent sample of the physical variable monitored. **Then**, the value of  
 629 the measurement is examined in particular whenever the binary flag,  
 630 defined above, is flipped.
- 631 – The mapping of all the variables that are shared with other PTAs  
 632 should be validated as well. In particular, the variables (*phy\_var*, *S<sub>et</sub>*,  
 633 *meas\_var*) in the Sensor PTA are examined against  $G_s$  in the glucose-  
 634 insulin dynamics PTA and (*S<sub>et1</sub>*, *var\_in*) in the *Act\_Channel<sub>lossy</sub>* PTA,  
 635 respectively. For a properly mapped system, the values of the variables  
 636 in a PTA should be matched to their corresponding ones in all other  
 637 PTAs at any time.
- 638 – *Channel<sub>lossy</sub>*: The PTA shown in Fig. 19-b has seven locations where the  
 639 edge from *loc3* towards *loc6* synchronizes with the sensor PTA to receive  
 640 the measurement value as an input variable *var\_in*. Similarly, the edge from  
 641 *loc7* to *loc5* synchronizes with the controller PTA to send the measurement  
 642 value as an output variable *var\_out*.
- 643 – For every received measurement, the PTA will either successfully relay  
 644 the measurement to the controller with probability  $P_S$  or fail with prob-

- 645 ability  $P_F$ . To check on this, binary flags are marked (flipped) on the  
 646 corresponding edges for success and failure ( $chk\_pt_{Success}$  and  $chk\_pt_F$   
 647 in the graph of Fig. 19-b). These binary flags are monitored for random  
 648 simulations over various probabilistic weights.
- 649 – A measurement is sent to the controller if and only if the edge with  $P_S$   
 650 probabilistic weight is traversed. This can be checked by examining the  
 651 corresponding binary flags.
  - 652 – Whenever a measurement is sent to the controller ( $S_{et2}$  is activated),  
 653 the value of the measurement ( $var\_out$ ) should be equal to the value of  
 654 the sample received from the sensor ( $var\_in$ ).
  - 655 – To validate the mapping of variables, the values of the variables ( $S_{et2}$ ,  
 656  $var\_out$ ) should be equal to the values of the corresponding variables  
 657 in the controller PTA ( $S_{et1}$ ,  $G$ ), respectively.
- 658 – Controller: The PTA shown in Fig. 20-a has five locations where the edge  
 659 from  $loc_3$  towards  $loc_5$  synchronizes with the lossy channel PTA to receive  
 660 the glucose measurement value as an input variable  $G$ . Similarly, the edge  
 661 from  $loc_4$  to  $loc_2$  synchronizes with the actuator PTA to send the control  
 662 value as an output variable  $IIR$ .

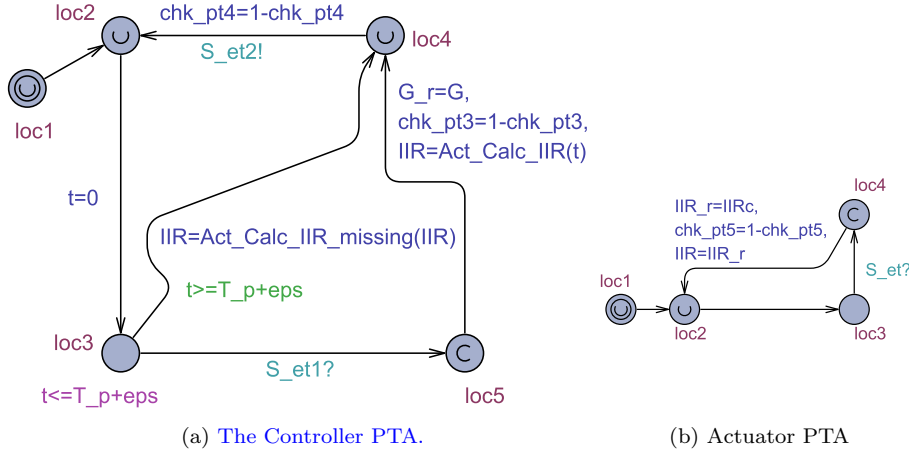


Fig. 20: A Part of the Artificial Pancreas Control Network.

- 663 – For each measurement delivered ( $S_{et1}$  activated), the PTA will read  
 664 the measurement value  $G$  and use it to calculate a new Insulin Infusion  
 665 Rate (IIR) using the standard **Proportional-Integral-Derivative (PID)**  
 666 control [4, 34]. This new calculated value of  $IIR$  should be sent to the  
 667 actuator by activating the event trigger  $S_{et2}$ .
- 668 – If the time since the last delivered measurement exceeds the control  
 669 period  $T_p$ , the value of the variable  $IIR$  is zeroed and the event trigger  
 670  $S_{et2}$  is activated to command insulin delivery suspension.

- 671 – To validate the mapping of variables, the values of the variables ( $S_{et2}$ ,  
672  $IIR$ ) should be equal to the values of the corresponding variables in  
673 the actuator PTA ( $S_{et}$ ,  $IIR_c$ ), respectively.
- 674 – Actuator: The PTA shown in Fig. 20-b has four locations where the edge  
675 from  $loc_3$  towards  $loc_4$  synchronizes with the controller PTA to receive the  
676 control value as an input variable  $IIR_c$ . The actuator then modifies the  
677 corresponding physical values in the glucose-insulin dynamics PTA through  
678 the output variable  $IIR$ .
  - 679 – Whenever a new infusion rate value  $IIR_c$  control command from the  
680 controller PTA is received ( $S_{et}$  activation), the actuator should update  
681 the value of the physical real-time variable  $IIR$ .
  - 682 – To verify the mapping of variables, the values for the variables  $IIR$  in  
683 both PTAs, actuator and glucose-insulin dynamics, should be equal at  
684 all times.
- 685 – Meal Scenario: This PTA is used to assign the input variables of the meal  
686 absorption model such as the carbohydrate amounts and the inter-meal  
687 times.
  - 688 – Each of the variables ( $meal\_carbs$ ,  $meal\_dur$ ,  $inter\_meal\_time$ ) takes  
689 a value ranging between the configured minimum and maximum with  
690 uniform distribution. Based on the histogram of the variables, this can  
691 be validated.
  - 692 – The PTA should generate the values of the real-time variables ( $cur\_meal$ ,  
693  $D_{meal}$ ) complying with the right amounts of insulin-carbs, meal dura-  
694 tions, and inter-meal times.
  - 695 – Validation for the mapping of the variables ( $cur\_meal$ ,  $D_{meal}$ ,  $Q_{sto1}$ ,  
696  $Q_{sto2}$ ) with their corresponding variables in the meal absorption PTA.
- 697 – Meal Absorption & Glucose-Insulin Dynamics:
  - 698 – The variables of the ODEs for both PTAs are observed and compared  
699 using our ODE simulator. The values for all variables should be identi-  
700 cal to the ones calculated by the mathematical ODE solver developed  
701 in Matlab except for marginal numerical computational errors, e.g. pre-  
702 cision.

## 703 5.2 Model Verification

704 PTAs are constructed for all the CPS components and are exported to a file for  
705 verification and analysis. This file is loaded into UPPAAL-SMC. A network of  
706 PTAs is created by instantiating and parallel-composing the PTA blocks using  
707 the UPPAAL-SMC. The tool performs hypothesis testing on queries specified  
708 by Metric Interval Temporal Logic (MITL). Also, monitor-based verification  
709 [8] could be used to specify more complicated queries using simpler expressions  
710 or for queries that are beyond the expressive power of MITL query language.

711 To demonstrate the use of the proposed framework to analyze real-life  
712 systems, UPPAAL-SMC is utilized to investigate safety properties of the arti-  
713 ficial pancreas CPS that is supposed to regulate the blood glucose levels using



714 a pre-configured closed-loop control strategy. A good control strategy would be  
 715 able to satisfy safety properties under normal conditions. Moreover, it would  
 716 accommodate disturbances and minimize the side effects of faults.

717 Using this system, the sensor periodically transmits measurements to the  
 718 controller over a wireless channel, but wireless packet transmission failure can  
 719 cause measurements to be missing. Missing measurements can be handled  
 720 using different control approaches. With the proposed SMC modeling and  
 721 analysis, it is possible to evaluate whether each control approach can preserve  
 722 safety properties at various error rates.

723 Whenever the controller receives a measurement, it calculates the required  
 724 insulin rate using the standard PID. For a missing measurement, the controller  
 725 will behave in one of three ways.

- 726 – Sustain: The controller will keep configuring the last valid calculated insulin  
 727 rate until a new valid measurement is received.
- 728 – Suspend: The controller will stop insulin delivery until a new valid mea-  
 729 surement is received.
- 730 – Revert: The controller will revert to a low value which is equal to the PID  
 731 controller basal insulin rate until a new valid measurement is received.

732 The analysis is conducted on a database of 10 adult patients publicly acces-  
 733 sible [38]. Each patient receives random meals of (20–50) *grams* carbohydrates  
 734 each. Per patient, the analysis evaluates whether or not the controller satisfies  
 735 safety properties for each of the three control configurations: sustain, suspend  
 736 or revert. The following two safety properties are defined for analysis.

- 737 –  $S_A$ : At all times, the blood glucose levels should not cross the boundaries  
 738 of severe minimum and maximum values of 50 *mg/dL* and 300 *mg/dL*,  
 739 respectively.
- 740 –  $S_B$ : Whenever the glucose elevates to values higher than the threshold of  
 741 180 *mg/dL*, it should restore its value to normal range below this threshold  
 742 within a maximum of two and a half hours.

743 The first safety property  $S_A$  is straightforward and can be described using  
 744 the following MITL query:

$$745 \quad \mathbf{Pr}[t \leq 1440] ( \square G \geq 50 \ \&\& \ G \leq 300 ) \geq 0.99$$

746 This property specifies that throughout the test duration of one day (1440  
 747 minutes) the blood glucose levels should be limited between 50 *mg/dL* and  
 748 300 *mg/dL* with a probability above or equal 99%. On the other side, the  
 749 second safety property  $S_B$  is too elaborate to describe in a query using MITL.  
 750 Instead, a monitor PTA is designed to observe the time duration for each  
 751 time the glucose level elevates above 180 *mg/dL* as shown in Fig. 21. Having  
 752 this variable ( $tg_{180}$ ) assigned, the safety property  $S_B$  is described using the  
 753 following MITL property.

$$754 \quad \mathbf{Pr}[t \leq 1440] ( \square tg_{180} \leq 150 ) \geq 0.99$$

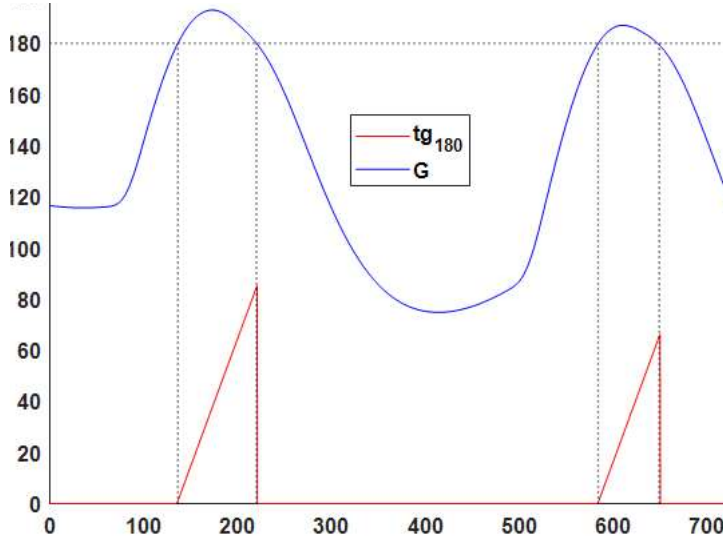


Fig. 21: The Duration of Time Where Glucose Exceeds 180 (mg/dL)  $\{tg_{180}\}$

755 This property is satisfied if and only if a high glucose incidence would  
 756 recover to normal range within two and a half hours maximum with at least  
 757 99% probability. It should be noted that the monitor PTA is constructed by  
 758 creating a SysML activity diagram characterizing its behavior as shown in Fig.  
 759 22 and applying the new proposed automatic procedure to convert the EAC  
 760 description into a PTA component that is parallel-composed with the other  
 761 PTAs in UPPAAL-SMC tool.

$$\begin{aligned}
 Act\_Monitor &= l \mapsto l_1 : B_C(l_2 : (C = G > 180) \mapsto N_1, l_3 : (C = G \leq 180) \mapsto N_2) \\
 N_1 &= l_4 : D_{CB}(G < 180, G \geq 180 - 1 \&\& tg'_{180} == 1) \\
 &\mapsto l_5 : Act(tg_{180} = 0) \mapsto l_6 : D_{CB}(G > 180 - 1, G \leq 180) \mapsto l_4 \\
 N_2 &= l_6
 \end{aligned}$$

762

763 The percentage of the patients with violations for each safety property  
 764 is shown in Fig. 23. No violations exist in the absence of message errors.  
 765 When message errors are introduced, the three control configurations result in  
 766 varying behaviors. For safety property  $S_A$ , message errors result in a gradual  
 767 increase of violations on *sustain* and *suspend* approaches. However, the *revert*  
 768 approach preserves the safety property  $S_A$  on all patients with message errors  
 769 up to 50%. For safety property  $S_B$ , the *suspend* approach fails on timely  
 770 recovery of normal glucose levels in the existence of message errors. The other  
 771 configurations, *sustain* and *revert*, avoid  $S_B$  violations with message errors as  
 772 high as 30%. When the error rate exceeds that level, violations start to occur  
 773 with the *revert* approach suffering more violations.

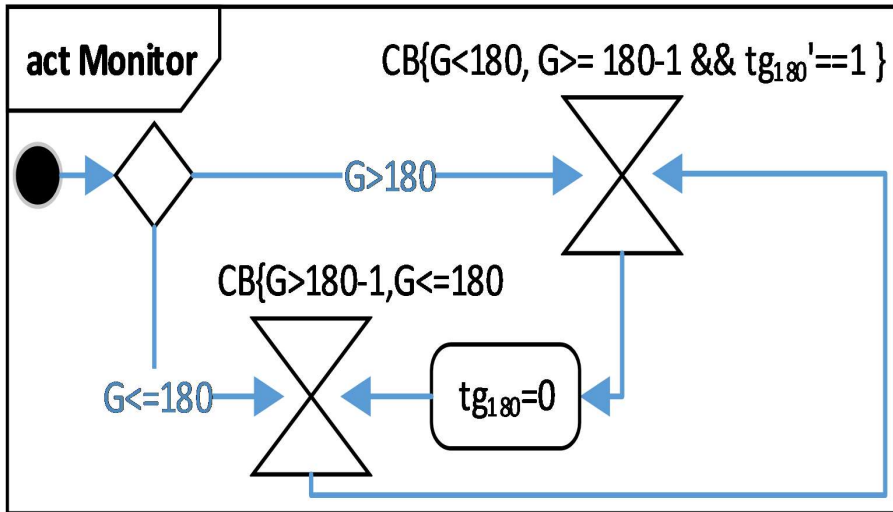


Fig. 22: SysML Activity Diagram of the Monitor

### 774 5.3 Discussion

775 To understand the experimental results, the following facts should be noted.

- 776 – In the absence of message errors, the three control configurations fall back
- 777 to being the same standard PID controller.
- 778 – The analyzed artificial pancreas is a single hormone unidirectional controller
- 779 (as opposed to dual-hormone systems [22]). This implies that it can
- 780 deliver more insulin to counteract the excessive glucose levels, but it can
- 781 only counteract low glucose levels by suspending the insulin delivery and
- 782 waiting for the pre-delivered insulin to get consumed by the physiological
- 783 processes inside the body.

784 Putting this in mind can explain the results on safety property  $S_A$  (left graph

785 in Fig. 23), where the *sustain* approach accidentally delivers excessive insulin

786 amounts that can cause glucose drops below  $50 \text{ mg/dL}$  even at low message

787 error rates. On the contrary, the *suspend* approach stops insulin delivery and

788 can make it up by restarting insulin delivery when valid messages are received

789 again. However, when the message error rate increases, there is a chance that

790 the *suspend* approach might fail to prevent large glucose levels above  $300$

791  $\text{mg/dL}$ . Instead of completely halting the insulin delivery, the *revert* continues

792 delivering small amounts of insulin to make a balance between the two other

793 approaches and avoid extreme highs and lows of glucose. The same concept

794 explains the results in the right graph of Fig. 23 where the *sustain* approach

795 provides better performance in avoiding long times with glucose levels above

796  $180 \text{ mg/dL}$  as opposed to the *suspend* approach which fails to avoid that.

797 The *revert* approach provides performance similar to the *sustain* approach

798 except for high message error rates where the violations start to increase when  
 799 utilizing the *revert* approach.



Fig. 23: Results for Safety Properties Violations:  $S_A$  (left) and  $S_B$  (right)

## 800 6 Conclusion

801 In this work, a framework is proposed to formally model and automatically an-  
 802alyze cyber-physical systems using statistical model checking. The framework  
 803takes models specified using SysML modeling language as SysML diagrams.  
 804The latters are then represented in textual format using the proposed enhanced  
 805activity calculus and ordinary-differential equations of SysML constraint dia-  
 806grams. Then, these textual representations of the model components are fed  
 807into a new proposed conversion algorithm that automatically transforms them  
 808into equivalent priced timed automata. Thus, the resulting model is fed into  
 809UPPAAL-SMC statistical model checking tool which parallel-composes all the  
 810system components and verifies the system behaviors. The use of the proposed  
 811framework to verify safety properties is demonstrated on an artificial pancreas  
 812case study.

813 The proposed framework can be used to verify the safety of cyber-physical  
 814systems and gain insight into their most critical behaviors at an early stage  
 815of the design process, thus saving valuable time and money. Ultimately, it  
 816promotes the integration of real-life problems into model-based analysis and  
 817allows experimenting a variety of scenarios without compromising participant  
 818safety. This is especially crucial when dealing with systems that involve human  
 819life, whether directly as in biomedical systems or indirectly as in automotive  
 820systems. In the near future, we target to improve the framework to cover more  
 821issues, mainly:

- 822 – Develop a library of different CPS components and applications.
- 823 – Model more cyber-physical systems with a focus on faults and security  
 824 threats.
- 825 – Before the CPS deployment, we target also to automatically generate the  
 826 source code related to the modeled and analyzed CPS.

- 827 – Provide guidance to correct the CPS whenever a property has not been  
828 satisfied.
- 829 – Establish a mechanism for defining CPS complex requirements automati-  
830 cally and easily.

## 831 References

- 832 1. Abdel-Latif Alshalalfah and Otmane Ait Mohamed. System-level mod-  
833 eling and safety analysis of vehicular coordinated emergency braking under  
834 degraded wireless connectivity using priced timed automata. In *2020*  
835 *27th IEEE International Conference on Electronics, Circuits and Systems*  
836 *(ICECS)*, pages 1–4. IEEE, 2020.
- 837 2. Abdel-Latif Alshalalfah, Ghaith Bany Hamad, and Otmane Ait Mohamed.  
838 Towards system level security analysis of artificial pancreas via uppaal-  
839 smc. In *2019 IEEE International Symposium on Circuits and Systems*  
840 *(ISCAS)*, pages 1–5. IEEE, 2019.
- 841 3. Abdel-Latif Alshalalfah, Ghaith Bany Hamad, and Otmane Ait Mohamed.  
842 System-level analysis of closed-loop anesthesia control under temporal sensor  
843 faults via uppaal-smc. In *2020 42nd Annual International Conference*  
844 *of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages  
845 2508–2511. IEEE, 2020.
- 846 4. Abdel-Latif Alshalalfah, Ghaith Bany Hamad, and Otmane Ait Mohamed.  
847 Towards safe and robust closed-loop artificial pancreas using improved pid-  
848 based control strategies. *IEEE Transactions on Circuits and Systems I:*  
849 *Regular Papers*, 68(8):3147–3157, 2021.
- 850 5. Georges M Arnaout and Jean-Paul Arnaout. Exploring the effects of coop-  
851 erative adaptive cruise control on highway traffic flow using microscopic  
852 traffic simulation. *Transportation Planning and Technology*, 37(2):186–  
853 199, 2014.
- 854 6. Ananda Basu, Saddek Bensalem, Marius Bozga, Benoît Delahaye, and  
855 Axel Legay. Statistical abstraction and model-checking of large hetero-  
856 geneous systems. *International Journal on Software Tools for Technology*  
857 *Transfer*, 14(1):53–72, 2012.
- 858 7. Gerd Behrmann, Alexandre David, and Kim G Larsen. A tutorial on  
859 uppaal. *Formal methods for the design of real-time systems*, pages 200–  
860 236, 2004.
- 861 8. Peter Bulychev, Alexandre David, Kim Guldstrand Larsen, Axel Legay,  
862 Guangyuan Li, Danny Bøgsted Poulsen, and Amélie Stainer. Monitor-  
863 based statistical model checking for weighted metric temporal logic. In  
864 *International Conference on Logic for Programming Artificial Intelligence*  
865 *and Reasoning*, pages 168–182. Springer, 2012.
- 866 9. Fraser Cameron, Georgios Fainekos, David M Maahs, and Sriram Sankara-  
867 narayanan. Towards a verified artificial pancreas: Challenges and solutions  
868 for runtime verification. In *Runtime Verification*, pages 3–17. Springer,  
869 2015.

- 870 10. Edmund M Clarke Jr, Orna Grumberg, Daniel Kroening, Doron Peled,  
871 and Helmut Veith. *Model checking*. MIT press, 2018.
- 872 11. ML Cummings and David Britton. Regulating safety-critical autonomous  
873 systems: past, present, and future perspectives. In *Living with robots*,  
874 pages 119–140. Elsevier, 2020.
- 875 12. Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis, and  
876 Zheng Wang. Time for statistical model checking of real-time systems. In  
877 *International Conference on Computer Aided Verification*, pages 349–355.  
878 Springer, 2011.
- 879 13. Alexandre David, DeHui Du, Kim G Larsen, Marius Mikučionis, and Arne  
880 Skou. An evaluation framework for energy aware buildings using statistical  
881 model checking. *Science China information sciences*, 55(12):2694–2707,  
882 2012.
- 883 14. Alexandre David, Kim Guldstrand Larsen, Axel Legay, Marius Mikučionis,  
884 Danny Bøgsted Poulsen, and Sean Sedwards. Runtime verification of bio-  
885 logical systems. In *International Symposium On Leveraging Applications  
886 of Formal Methods, Verification and Validation*, pages 388–404. Springer,  
887 2012.
- 888 15. Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis, and  
889 Danny Bøgsted Poulsen. Uppaal smc tutorial. *International journal on  
890 software tools for technology transfer*, 17(4):397–415, 2015.
- 891 16. Yair Bar David, Tal Geller, Ilai Bistriz, Irad Ben-Gal, Nicholas Bambos,  
892 and Evgeni Khmelnsky. Wireless body area network control policies for  
893 energy-efficient health monitoring. *Sensors*, 21(12):4245, 2021.
- 894 17. Mourad Debbabi, Fawzi Hassaine, Yosr Jarraya, Andrei Soeanu, and Luay  
895 Alawneh. *Verification and validation in systems engineering: assessing  
896 UML/SysML design models*. Springer Science & Business Media, 2010.
- 897 18. Predrag Filipovikj, Nesredin Mahmud, Raluca Marinescu, Cristina Sece-  
898 leanu, Oscar Ljungkrantz, and Henrik Lönn. Simulink to uppaal statistical  
899 model checker: Analyzing automotive industrial systems. In *FM 2016: For-  
900 mal Methods: 21st International Symposium, Limassol, Cyprus, November  
901 9-11, 2016, Proceedings 21*, pages 748–756. Springer, 2016.
- 902 19. Goran Frehse. An introduction to hybrid automata, numerical simulation  
903 and reachability analysis. In *Formal Modeling and Verification of Cyber-  
904 Physical Systems: 1st International Summer School on Methods and Tools  
905 for the Design of Digital Systems, Bremen, Germany, September 2015*,  
906 pages 50–81. Springer, 2015.
- 907 20. Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi  
908 Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and  
909 Oded Maler. Spaceex: Scalable verification of hybrid systems. In *In-  
910 ternational Conference on Computer Aided Verification*, pages 379–395.  
911 Springer, 2011.
- 912 21. Patrice Godefroid. *Partial-order methods for the verification of concurrent  
913 systems: an approach to the state-explosion problem*. Springer, 1996.
- 914 22. Ahmad Haidar, Laurent Legault, Virginie Messier, Tina Maria Mitre,  
915 Catherine Leroux, and Rémi Rabasa-Lhoret. Comparison of dual-hormone

- artificial pancreas, single-hormone artificial pancreas, and conventional insulin pump therapy for glycaemic control in patients with type 1 diabetes: an open-label randomised controlled crossover trial. *The lancet Diabetes & endocrinology*, 3(1):17–26, 2015.
23. Jinpei Han, Joseph Davids, Hutan Ashrafian, Ara Darzi, Daniel S Elson, and Mikael Sodergren. A systematic review of robotic surgery: From supervised paradigms to fully autonomous robotic approaches. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 18(2):e2358, 2022.
24. Thomas Hérault, Richard Lassaigne, Frédéric Magniette, and Sylvain Peyronnet. Approximate probabilistic model checking. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 73–84. Springer, 2004.
25. Jon Holt and Simon Perry. *SysML for systems engineering*, volume 7. IET, 2008.
26. Zhihao Jiang, Miroslav Pajic, Salar Moarref, Rajeev Alur, and Rahul Mangharam. Modeling and verification of a dual chamber implantable pacemaker. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 188–203. Springer, 2012.
27. Nikolaos Kekatos, Marcelo Forets, and Goran Frehse. Constructing verification models of nonlinear simulink systems via syntactic hybridization. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1788–1795. IEEE, 2017.
28. Nikolaos Kekatos, Marcelo Forets, and Goran Frehse. Modeling the wind turbine benchmark with pwa hybrid automata. *EPiC Series in Computing*, 48:100–113, 2017.
29. Jun Kit Koong, Gaik Huey Ng, Kamarajan Ramayah, Peng Soon Koh, and Boon Koon Yoong. Early identification of the critical view of safety in laparoscopic cholecystectomy using indocyanine green fluorescence cholangiography: A randomised controlled study. *Asian Journal of Surgery*, 44(3):537–543, 2021.
30. Ray Kurzweil. The law of accelerating returns. In *Alan Turing: Life and legacy of a great thinker*, pages 381–416. Springer, 2004.
31. M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV’11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.
32. Sridhar Lakshmanan, Yuedong Yan, Stan Baek, and Hesham Alghodhaifi. Modeling and simulation of leader-follower autonomous vehicles: environment effects. In *Unmanned systems technology XXI*, volume 11021, page 110210J. International Society for Optics and Photonics, 2019.
33. Kim G Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and computation*, 94(1):1–28, 1991.
34. Srinivas Laxminarayan, Jaques Reifman, and Garry M Steil. Use of a food and drug administration-approved type 1 diabetes mellitus simulator to evaluate and optimize a proportional-integral-derivative controller.

- 962 *Journal of diabetes science and technology*, 6(6):1401–1412, 2012.
- 963 35. Axel Legay, Anna Lukina, Louis Marie Traonouez, Junxing Yang, Scott A  
964 Smolka, and Radu Grosu. Statistical model checking. In *Computing and  
965 Software Science*, pages 478–504. Springer, 2019.
- 966 36. Alexios Lekidis, Paraskevas Bourgos, Simplicio Djoko-Djoko, Marius  
967 Bozga, and Saddek Bensalem. Building distributed sensor network appli-  
968 cations using bip. In *2015 IEEE Sensors Applications Symposium (SAS)*,  
969 pages 1–6. IEEE, 2015.
- 970 37. Jun Liu, Kara M Kockelman, Patrick M Boesch, and Francesco Ciari.  
971 Tracking a system of shared autonomous vehicles across the austin, texas  
972 network using agent-based simulation. *Transportation*, 44(6):1261–1278,  
973 2017.
- 974 38. Chiara Dalla Man, Francesco Micheletto, Dayu Lv, Marc Breton, Boris  
975 Kovatchev, and Claudio Cobelli. The uva/padova type 1 diabetes simula-  
976 tor: new features. *Journal of diabetes science and technology*, 8(1):26–34,  
977 2014.
- 978 39. Braham Lotfi Mediouni, Ayoub Nouri, Marius Bozga, Mahieddine Della-  
979 bani, Axel Legay, and Saddek Bensalem. SBIP 2.0: Statistical model  
980 checking stochastic real-time systems. In *International Symposium on  
981 Automated Technology for Verification and Analysis*, pages 536–542.  
982 Springer, 2018.
- 983 40. Stefano Minopoli and Goran Frehse. Sl2sx translator: from simulink to  
984 spacex models. In *Proceedings of the 19th International Conference on  
985 Hybrid Systems: Computation and Control*, pages 93–98, 2016.
- 986 41. Umberto Montanaro, Shilp Dixit, Saber Fallah, Mehrdad Dianati, Alan  
987 Stevens, David Oxtoby, and Alexandros Mouzakitis. Towards connected  
988 autonomous driving: review of use-cases. *Vehicle system dynamics*, 57(6):  
989 779–814, 2019.
- 990 42. Katherine Ogurtsova, JD da Rocha Fernandes, Y Huang, Ute Linnenkamp,  
991 L Guariguata, Nam H Cho, David Cavan, JE Shaw, and LE Makaroff. Idf  
992 diabetes atlas: Global estimates for the prevalence of diabetes for 2015  
993 and 2040. *Diabetes research and clinical practice*, 128:40–50, 2017.
- 994 43. Masashi Okamoto. Some inequalities relating to the partial sum of bino-  
995 mial probabilities. *Annals of the institute of Statistical Mathematics*, 10  
996 (1):29–35, 1959.
- 997 44. Samir Ouchani, Yosr Jarraya, Otmane Ait Mohamed, and Mourad Deb-  
998 babi. Probabilistic attack scenarios to evaluate policies over communica-  
999 tion protocols. *J. Softw.*, 7(7):1488–1495, 2012.
- 1000 45. Samir Ouchani, Otmane Ait Mohamed, and Mourad Debbabi. A formal  
1001 verification framework for sysml activity diagrams. *Expert Systems with  
1002 Applications*, 41(6):2713–2728, 2014.
- 1003 46. Miroslav Pajic, Rahul Mangharam, Oleg Sokolsky, David Arney, Julian  
1004 Goldman, and Insup Lee. Model-driven safety analysis of closed-loop  
1005 medical systems. *IEEE Transactions on Industrial Informatics*, 10(1):  
1006 3–16, 2012.



- 1007 47. Alkis Papadoulis, Mohammed Quddus, and Marianna Imprialou. Evaluat-  
1008 ing the safety impact of connected and autonomous vehicles on motorways.  
1009 *Accident Analysis & Prevention*, 124:12–22, 2019.
- 1010 48. Sriram Sankaranarayanan, Suhas Akshar Kumar, Faye Cameron, B Wayne  
1011 Bequette, Georgios Fainekos, and David M Maahs. Model-based falsifica-  
1012 tion of an artificial pancreas control system. *ACM SIGBED Review*, 14  
1013 (2):24–33, 2017.
- 1014 49. Stefan Schupp, Erika Ábrahám, Xin Chen, Ibtissem Ben Makhlof, Goran  
1015 Frehse, Sriram Sankaranarayanan, and Stefan Kowalewski. Current chal-  
1016 lenges in the verification of hybrid systems. In *International Workshop on*  
1017 *Design, Modeling, and Evaluation of Cyber Physical Systems*, pages 8–24.  
1018 Springer, 2015.
- 1019 50. Stefan Schupp, Francesco Leofante, Leander Behr, Erika Ábrahám, and  
1020 Armando Tacella. Robot swarms as hybrid systems: Modelling and veri-  
1021 fication. *arXiv preprint arXiv:2207.06758*, 2022.
- 1022 51. Koushik Sen, Mahesh Viswanathan, and Gul Agha. Statistical model  
1023 checking of black-box probabilistic systems. In *International Conference*  
1024 *on Computer Aided Verification*, pages 202–215. Springer, 2004.
- 1025 52. Robert E Shannon. Systems simulation; the art and science. Technical  
1026 report, 1975.
- 1027 53. Alyona Skorobogatjko, Andrejs Romanovs, and Nadezhda Kunicina. State  
1028 of the art in the healthcare cyber-physical systems. *Information Technol-*  
1029 *ogy and Management Science*, 17(1):126–131, 2014.
- 1030 54. OMG Available Specification. Omg systems modeling language (omg  
1031 sysml™), v1. 0, 2007.
- 1032 55. Kay W Axhausen, Andreas Horni, and Kai Nagel. *The multi-agent trans-*  
1033 *port simulation MATSim*. Ubiquity Press, 2016.
- 1034 56. Abraham Wald. *Sequential analysis*. Courier Corporation, 2004.
- 1035 57. Hakan Lorens Samir Younes. *Verification and planning for stochastic pro-*  
1036 *cesses with asynchronous events*. Carnegie Mellon University, 2004.
- 1037 58. Håkan LS Younes and Reid G Simmons. Probabilistic verification of dis-  
1038 crete event systems using acceptance sampling. In *International Confer-*  
1039 *ence on Computer Aided Verification*, pages 223–235. Springer, 2002.