Formal algebraic modelling of a city-wide smart parking system

Conference Paper · June 2020

DOI: 10.1109/ICECCE49384.2020.9179447

5 authors, including:

Pedro Juan Roig
Universidad Miguel Hernández de Elche
35 PUBLICATIONS **65** CITATIONS
SEE PROFILE

Salvador Alcaraz
Universidad Miguel Hernández de Elche
53 PUBLICATIONS **137** CITATIONS
SEE PROFILE

Katja Gilly
Universidad Miguel Hernández de Elche
66 PUBLICATIONS **273** CITATIONS
SEE PROFILE

Sonja Filiposka
Ss. Cyril and Methodius University in Skopje
140 PUBLICATIONS **840** CITATIONS
SEE PROFILE

Formal algebraic modelling of a city-wide smart parking system

Pedro Juan Roig
Department of Computer Engineering
Miguel Hernández University
Elche (Alicante), Spain
pedro.roig@graduado.umh.es

Salvador Alcaraz
Department of Computer Engineering
Miguel Hernández University
Elche (Alicante), Spain
salcaraz@umh.es

Katja Gilly
Department of Computer Engineering
Miguel Hernández University
Elche (Alicante), Spain
katya@umh.es

Sonja Filiposka
Faculty of Computer Science and Eng.
Ss. Cyril and Methodius University
Skopje, North Macedonia
sonja.filiposka@finki.ukim.mk

Noura Aknin
Faculty of Science
Abdelmalek Essaadi University
Tétouan, Morocco
noura.aknin@uae.ac.ma

***Abstract*—One of the main applications of smart cities are parking systems based on IoT/Fog technology. In this paper, we carry out the study of formal algebraic models for such systems. In that context, first of all we undertake the study of a model for a street with a single parking space, extending it with a string of parking spaces. Additionally, we add up some parking control at fog level in order to apply parking restrictions in case of local environment issues, such as air pollution or poor weather conditions. Eventually, we also add up an orchestration level so as to prevent parking related to intelligence information, such as traffic congestion down the road, an accident nearby, or adverse weather forecast to come.**

***Keywords*—ACP, fog computing, IoT, formal protocol specification, networking.**

I. INTRODUCTION

Parking issues are a common nuisance in every city all over the world [1], but the use of new technologies applied in the context of smart cities may help mitigate its effects [2]. Mainly, the rise of IoT/fog deployments around a city may improve how the drivers find a proper parking place by using different approaches, as described in [3], [4] and [5].

Focusing on the synergy between fog computing and IoT [6], it is considered as one of the key vectors in research and development within the new technologies field, along with quantum computing [7] and artificial intelligence [8].

The growth and evolution of such IoT/fog facilities will provide the necessary infrastructure to implement smart parking policies, by means of a wide range of technologies, such as 5G, ZigBee, Bluetooth or new WiFi standards [9].

IoT devices usually have restricted capabilities as per computing power, memory, storage, bandwidth and batteries, hence, they generally have associated virtual machines (VMs) in a remote location in order to undertake their computing needs [10].

Those remote locations are usually the cloud or the fog, although the latter seems to be the best solution, as latency and bandwidth get improved. Also, VM migration in a fog environment provides the capability to VMs to get closer to their associated IoT devices as they move around [11].

In this paper, the target is to achieve a formal algebraic model of a smart parking system based on IoT/fog technology, which may control the availability of a given parking space in order to permit or deny the driver to pull over, according to some information obtained by external sources, those being environmental sensors or intelligence.

For this purpose, first of all, a formal algebraic model is going to be derived for a street where no parking restriction controls are put in place. Then, a fog level entity is attached so as to control each parking place based on sensor information obtained from that area, regarding air pollution or weather conditions. Finally, an orchestration level is attached as a higher entity to control all fog facilities to implement joint operations in all of them due to exceptional conditions, or otherwise, to load balance flows at fog level.

In order to keep things simple, a linear setup is going to be considered at a fog level, thus considering it as a long straight avenue with multiple parking places, and as a consequence, orchestration level is going to be considered as the central entity joining together all of those linear avenues. However, any other kind of topology at a fog level might be considered, just by changing the available paths among blocks and carrying out the proper derivations. Nevertheless, if that work is undertaken, the overall results obtained may not differ much in essence from those reached herein.

The organisation of this paper is as follows: first of all, Section 2 presents ACP basics, then, Section 3 works out the specification of a single block, afterwards, Section 4 extends it to a string of blocks, next, Section 5 adds up control at a fog level, after that, Section 6 includes control at an orchestration level, later on, Section 7 shows a case study with a dataset, and finally, Section 8 draws final conclusions.

II. ACP BASICS

The formal algebraic model is going to be built by means of a process algebra called Algebra of Communicating Processes (ACP), being an abstract algebra aimed at reasoning about the properties and relationships of concurrent systems [12].

The algebraic model has to be formed by two sorts of atomic actions, namely, send through a channel and read out of it, making process terms. Besides, these actions are being applied to entities, those being considered as processes. Some operators are allowed to establish relationships among process terms [13], such as the *sequential* one (defined by the multiplication sign) and the *alternative* one (given by the addition sign). On the other hand, the fact that some entities are working in a concurrent manner is expressed by the *concurrent* operator (stated by the ‖ sign).

Regarding the latter, also known as merge operator, it is necessary to cite the Expansion Theorem by Bergstra and Klop [14], stating the equivalent operations to be carried out when dealing with concurrent entities. Those are *left merge* (quoted by the ⊩ sign), meaning that the initial transition of the entity called is executed first and then it behaves as a merge, and *communication merge* (named by the | sign), meaning that a communication between the initial transitions of the entities called are executed first and then it behaves as an ordinary merge [15].

$$\left(Y_1 \parallel \ldots \parallel Y_n\right) = \sum Y_i \Vdash Y^i + \sum \left(Y_i | Y_j\right) \Vdash Y^{i,j} \tag{1}$$

In that expression, the term Y^i implies all entities involved but the i-th one, this is, $Y^i = \{Y_1 \parallel \ldots \parallel Y_n\} - \{Y_i\}$, and the term $Y^{i,j}$ implies all entities involved but the i-th and the j-th ones, this is, $Y^{i,j} = \{Y_1 \parallel \ldots \parallel Y_n\} - \{Y_i, Y_j\}$.

At this point, the *encapsulation* operator is employed to transform atomic actions within the system into communications, leading to deadlock the rest of them. This is expressed by using the symbol ∂_H, where set H includes the aforesaid internal atomic actions. Afterwards, the *abstraction* operator is used to hide all internal communications, hence, permitting only external communications to show, thus unveiling the external behaviour of the model. This is expressed by means of the symbol τ_I, where set I includes all internal communications.

Eventually, the external behaviour of the real system is expressed in ACP notation and it is then compared with that of the model [16], and if both are rooted branching bisimilar, meaning that they share the same string of actions and the same branching structure, then it may be said that the model gets verified [17].

III. Specification of a single Building Block

To start with, let us suppose a piece of a one-way street in which there is just one parking place. Such a simple scenario may be modelled by a building block like the one shown in Fig. 1.

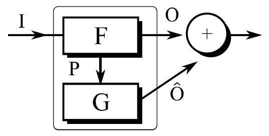


Fig. 1. Diagram with just one building block

Two different areas may be spotted therein, which may well be considered as different entities, those being *F* and *G*, where the former represents the piece of a one-way street and the latter, a parking space which may be available or not.

Also, four channels may be distinguished coming into a state or going out of an entity, those being channel *I* in order to get into the one-way street, channel *P* to use the parking place if it is available and the user wants to, channel *O* to leave the street without having parked, and channel *Ô* to get out of the street after having parked. All of them may well be considered as transitions among the states called previously.

Furthermore, two atomic actions may be defined, such as read (*r*) and send (*s*), the former to get into one of both states and the latter to get off a state, whereas the messages passing through those atomic actions are being denoted by the variable *d*, whose values are taken from dataset Δ.

Taking all that into account, it is possible to define the recursive variables explaining the model description:

$$F = \sum_{d\epsilon\Delta} r_I(d) \cdot (s_P(d) + s_O(d)) \cdot F \tag{2}$$

$$G = \sum_{d\epsilon\Delta} r_P(d) \cdot s_{\hat{O}}(d) \cdot G$$

It is to be noted that the states within the building block work in a concurrent manner, hence, the Expansion Theorem of Bergstra and Klop customised for two entities may be applied in order to get the behaviour of the overall system.

$$Y_1 \parallel Y_2 = Y_1 \Vdash Y_2 + Y_2 \Vdash Y_1 + (Y_1 | Y_2) \tag{3}$$

Let us suppose that initially there are no vehicles into the system, this is, states F and G are empty. In such a case, the application of the *encapsulation* operator yields the following outcome, where data related to atomic actions are omitted to keep expressions as simple as possible.

$$\partial_H(F \parallel G) = r_I \cdot \partial_H(\{(s_P + s_O) \cdot F\} \parallel G) \tag{4}$$

$$\begin{aligned}\partial_H(\{(s_P + s_O) \cdot F\} \parallel G) \\ = s_O \cdot \partial_H(F \parallel G) + c_P \cdot \partial_H(F \\ \parallel \{s_{\hat{O}} \cdot G\})\end{aligned}$$

$$\begin{aligned}\partial_H(F \parallel \{s_{\hat{O}} \cdot G\}) = r_I \cdot \partial_H(\{(s_P + s_O) \cdot F\} \\ \parallel \{s_{\hat{O}} \cdot G\}) + s_{\hat{O}} \cdot \partial_H(F \parallel G)\end{aligned}$$

$$\begin{aligned}\partial_H(\{(s_P + s_O) \cdot F\} \parallel \{s_{\hat{O}} \cdot G\}) \\ = s_O \cdot \partial_H(F \parallel \{s_{\hat{O}} \cdot G\}) + s_{\hat{O}} \\ \cdot \partial_H(\{(s_P + s_O) \cdot F\} \parallel G)\end{aligned}$$

The next step is to apply the *abstraction* operator, yielding the following results:

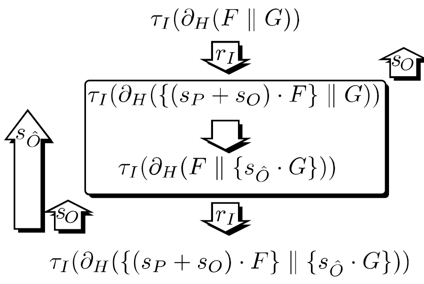
$$\tau_I(\partial_H(F \parallel G)) = r_I \cdot \tau_I(\partial_H(\{(s_P + s_O) \cdot F\} \parallel G)) \tag{5}$$

$$\begin{aligned}\tau_I(\partial_H(\{(s_P + s_O) \cdot F\} \parallel G)) \\ = s_O \cdot \tau_I(\partial_H(F \parallel G)) \\ + \tau_I(\partial_H(F \parallel \{s_{\hat{O}} \cdot G\}))\end{aligned}$$

$$\begin{aligned}\tau_I(\partial_H(F \parallel \{s_{\hat{O}} \cdot G\})) \\ = r_I \cdot \tau_I(\partial_H(\{(s_P + s_O) \cdot F\} \\ \parallel \{s_{\hat{O}} \cdot G\})) \\ + s_{\hat{O}} \cdot \tau_I(\partial_H(F \parallel G))\end{aligned}$$

$$\begin{aligned}\tau_I(\partial_H(\{(s_P + s_O) \cdot F\} \parallel \{s_{\hat{O}} \cdot G\})) \\ = s_O \cdot \tau_I(\partial_H(F \parallel \{s_{\hat{O}} \cdot G\})) + s_{\hat{O}} \\ \cdot \tau_I(\partial_H(\{(s_P + s_O) \cdot F\} \parallel G))\end{aligned}$$

With these results, it is possible to build up a state diagram such as Fig. 2, noting that the first expression corresponds to the case where there is no vehicle into the building block, the second one shows the case where there is a vehicle inside the piece of one-way street but the parking place is empty, the third one means the case where the piece of one-way street is empty but the parking place is full, and finally, the last one states the case where both the piece of one-way street and the parking are full.



$\tau_I(\partial_H(F \parallel G))$
r_I
s_O
$s_{\hat{O}}$
$\tau_I(\partial_H(\{(s_P + s_O)\cdot F\} \parallel G))$
$\tau_I(\partial_H(F \parallel \{s_{\hat{O}}\cdot G\}))$
$s_{\hat{O}}$
s_O
r_I
$\tau_I(\partial_H(\{(s_P + s_O)\cdot F\} \parallel \{s_{\hat{O}}\cdot G\}))$

Fig. 2. State diagram for just one building block

It is to be remarked that the state diagram shows four internal states, but from an external point of view, cases second and third are the same, as they both represent the situation where there is just one vehicle into the system, and that is why the transition between them is internal, hence, they are wrapped together in the state diagram. As per those two equivalent states externally, the lower one is going to be selected as the canonical representative of that state, as it sets free the entity F, which allows entrance of a new item into the system.

As a general rule, the numbers of states from the internal and external point of view are given by the number of entities into the system considered, in this case, the building block. If such a number is supposed to be *n*, the number of states viewed from outside the system ranges from *0* to *n*, this is, *n+1* possible states. Furthermore, for each of those *k* states, there may be some equivalent expressions, associated with the number of internal states available for each *k* value, whose numbers are obtained either by the *n-th* row and the *k-th* column of the Pascal Triangle, as depicted in Fig. 3, or by the number of combinations of k items taken from n items, as per the mathematical expression: $C(n,k) = \binom{n}{k} = \frac{n!}{k!\cdot(n-k)!}$

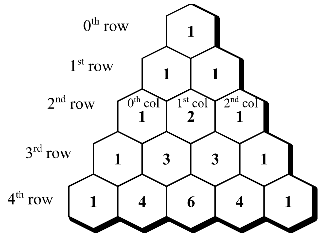


Fig. 3. Pascal Triangle for the first rows, showing the columns for n=2

Therefore, the state diagram herein, where n=2 as there are just 2 entities within a building block, shows only three states externalwise, whilst it shows four states internalwise, and that determines the external behaviour of the model proposed:

$$\tau_I(\partial_H(F \parallel G)) = r_I \cdot \tau_I(\partial_H(\{(s_P + s_O)\cdot F\} \parallel G)) \quad (6)$$

$$\tau_I(\partial_H(F \parallel \{s_{\hat{O}}\cdot G\})) = r_I \cdot \tau_I(\partial_H(\{(s_P + s_O)\cdot F\} \parallel \{s_{\hat{O}}\cdot G\}) + s_{\hat{O}} \cdot \partial_H(F \parallel G)$$

$$\tau_I(\partial_H(\{(s_P + s_O)\cdot F\} \parallel \{s_{\hat{O}}\cdot G\})) = s_O \cdot \tau_I(\partial_H(F \parallel \{s_{\hat{O}}\cdot G\})) + s_{\hat{O}} \cdot \tau_I(\partial_H(\{(s_P + s_O)\cdot F\} \parallel G))$$

At this point, it is to be pointed out the external behaviour of the real system. In order to undertake it, let us first consider the set of variables X_i, where *i* states for the number of vehicles inside the system. With that in mind, the external behaviour of the real system may be exposed by these recursive expressions:

$$X_0 = r_I \cdot X_1 \quad (7)$$
$$X_1 = r_I \cdot X_2 + (s_O + s_{\hat{O}}) \cdot X_0$$
$$X_2 = (s_O + s_{\hat{O}}) \cdot X_1$$

Finally, by comparing the external behaviour of the real system with that of the model proposed, it may be clearly seen that the following state matches apply, as each pair of recursive variables have analogous expressions:

$$X_0 \leftrightarrow \tau_I(\partial_H(F \parallel G)) \quad (8)$$
$$X_1 \leftrightarrow \tau_I(\partial_H(F \parallel \{s_{\hat{O}}\cdot G\}))$$
$$X_2 \leftrightarrow \tau_I(\partial_H(\{(s_P + s_O)\cdot F\} \parallel \{s_{\hat{O}}\cdot G\}))$$

Therefore, the model proposed exhibits the same external behaviour as the real system, meaning that they both are rooted branching bisimilar. And this fact leads to the conclusion that the model proposed gets verified.

The model proposed may be considered as a 2-item queue with priority, as the way they both work is analogous. This may be seen if a triage operation is considered in order to select whether an item is put in the slower queue (going to parking, from state F to state G), or otherwise, in the faster queue (getting out of the system from state F).

IV. Specification of more that one Building Block

The model proposed for one single building block may be easily extended, as the fact that there might be more than one parking place in a street may be seen as a string of *n* sequential building blocks, such as in Fig. 4.

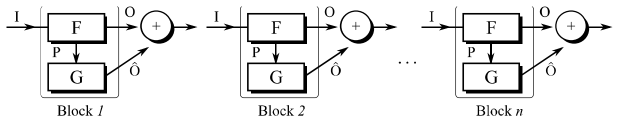


Fig. 4. State diagram for more than one building block

As those building blocks only have one entry point and two exit points, but being merged into just one, and being a one-way street means that the blocks will grow from left to right, it may appear that a string of such blocks behave like a string of natural numbers (ℕ) growing sequentially, starting by block 1 (even though starting by block 0 would also apply).

In order to check this out, it is to be verified that a string of blocks being connected in a linear daisy chain way is a totally ordered set, meaning that each block has a unique predecessor and a unique successor.

The properties to be checked out for three blocks whatsoever such as *a, b* and *c* are the following: reflexivity, such as $a \le a$ for all blocks, antisymmetry, such as if $a \le b$ and $b \le a$, then $a = b$, transitivity, such as if $a \le b$ and $b \le c$, then $a \le c$, and comparability, such as $a \le b$ or $b \le a$. It is clear that all of them apply due to the characteristics of a linear daisy chain topology, thus, total order applies in this case.

Therefore, the string of building blocks proposed in the model may be considered as a subset of natural numbers, even though such a subset does not contain any neutral element, thus, it is not isomorphic to the set of all natural numbers, but it does not really matter in this case. Hence, by applying structural induction, the model gets verified for *n* blocks.

At this stage, it is to be taken into account that a two-way street fits the model as well, as the cars coming in one direction may not change it in the middle of the street, thus they follow that direction all the way. This implies that rightwards and leftwards traffic do not mix to each other, allowing them to be treated as independent flows. Hence, the model may hold for two-way streets as well.

V. Adding up Control at Fog Level

At this point, some degree of control at fog level is going to be introduced, where a control channel (*ctl*) comes in to each block so as to permit the parking in such a block, or otherwise, depending on the information that the fog level obtains from the environment channel (*env*), regarding air pollution and weather conditions for each area. This is exhibited in Fig. 5.

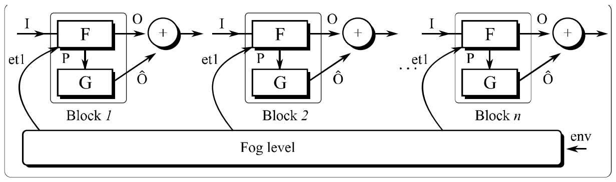


Fig. 5. State diagram adding up control at fog level

In this scenario, there are three entities, such as F and G defined previously, and the fog level, which may be called H and is common to all blocks, meaning that it may influence all parking places to be free or otherwise by means of signaling through control channel, depending on the environment information it gets.

It is going to be supposed that parking in a given block is only allowed when the proper communication from the *ctl* control channel is received on that block, namely at entity F. In other words, *ctl* signaling may enable or disable the access to G entity through channel P, in a way that such channel becomes unreachable. However, it does nothing to do with channel Ô, so it still remains clear for an item in G to leave.

Putting this all together, it is possible to define the recursive variables explaining the model description for a single block, where d accounts for data messages taken from dataset Δ, whereas d' does it for control messages taken from dataset Δ', and e does it for environment messages ∇:

$$F = \sum_{d\epsilon\Delta, d'\epsilon\Delta'} r_I(d) \cdot \left(r_{ctl}(d') \cdot s_P(d) + s_O(d)\right) \cdot F \qquad (9)$$

$$G = \sum_{d\epsilon\Delta} r_P(d) \cdot s_{\hat{O}}(d) \cdot G$$

$$H = \sum_{d'\epsilon\Delta', e\epsilon\nabla} r_{env}(e) \cdot s_{ctl}(d') \cdot H$$

In this context, the Expansion Theorem of Bergstra and Klop has to be customised for three entities, and it may be applied in order to get the behaviour of the overall system.

$$\begin{aligned} Y_1 \parallel Y_2 \parallel Y_3 = {} & Y_1 ⫦ (Y_2 \parallel Y_3) + Y_2 ⫦ (Y_1 \parallel Y_3) \\ & + Y_3 ⫦ (Y_1 \parallel Y_2) + (Y_1 | Y_2) ⫦ Y_3 \\ & + (Y_1 | Y_3) ⫦ Y_2 + (Y_2 | Y_3) ⫦ Y_1 \end{aligned} \qquad (10)$$

The necessary condition for parking to be allowed is that an *env* measurement received at the fog level indicates that there is no issue regarding the environment. If this is the case, the H entity is ready to send positive control messages to all blocks (through channel *ctl_i*) in order to allow parking in them, otherwise, no parking will be permitted (by means of channel *ctl_i*) until environmental conditions get better.

Hence, a positive *env* measurement makes possible a communication in channel ctl_i and opens the way for a new parking to be obtained. But that is the only moment where this affects the system behaviour, as it is irrelevant where no parking is sought after nor a parking place is already full. Therefore, the *env* signal may be seen as a filter to access the parking state, in a way that if the filter is switched on, the system behaviour will be the same as in Section III, whereas if it is switched off, the system will only allow one item in.

Let us suppose that initially there are no vehicles into a block. Then, the resulting calculations are the following, leaving out the transferred data in the expressions for simplicity purposes, and considering that control signals will be masked by applying the abstraction operator:

$$\begin{aligned} & \tau_I(\partial_H(F \parallel G \parallel H)) \\ & \quad = r_I \cdot \tau_I(\partial_H(\{(r_{ctl} \cdot s_P + s_O) \cdot F\} \parallel G \parallel H)) \end{aligned} \qquad (11)$$

$$\begin{aligned} & \tau_I(\partial_H(\{(r_{ctl} \cdot s_P + s_O) \cdot F\} \parallel G \parallel H) \\ & \quad = s_O \cdot \tau_I(\partial_H(F \parallel G \parallel H)) \\ & \quad + r_{env} \cdot \tau_I(\partial_H(F \parallel \{s_{\hat{O}} \cdot G\} \parallel H)) \end{aligned}$$

$$\begin{aligned} & \tau_I(\partial_H(F \parallel \{s_{\hat{O}} \cdot G\} \parallel H)) \\ & \quad = r_I \cdot \tau_I(\partial_H(\{(r_{ctl} \cdot s_P + s_O) \cdot F\} \parallel \{s_{\hat{O}} \cdot G\} \parallel H) \\ & \quad + s_{\hat{O}} \cdot \partial_H(F \parallel G \parallel H) \end{aligned}$$

$$\begin{aligned} & \tau_I(\partial_H(\{(r_{ctl} \cdot s_P + s_O) \cdot F\} \parallel \{s_{\hat{O}} \cdot G\} \parallel H)) \\ & \quad = s_O \cdot \tau_I(\partial_H(F \parallel \{s_{\hat{O}} \cdot G\} \parallel H)) \\ & \quad + s_{\hat{O}} \cdot \tau_I(\partial_H(\{(r_{ctl} \cdot s_P + s_O) \cdot F\} \parallel G \parallel H)) \end{aligned}$$

Regarding the external behaviour of the real system, the expressions are the same as the ones presented for a single building block in (7), whereas the external behaviour of the model is given by the first, third and fourth expressions in (11), as explained in Section III, taking into account that r_{env} only affects an internal transition. Furthermore, the reasoning exposed in Section IV still applies, so it may be extended to a string of blocks for both one-way and two-way streets.

Hence, by comparing the external behaviour of both the real system and the model, it may be clearly spotted that the following state matches apply, as each pair of recursive variables have analogous expressions, in a way that the model presents a restricted version of the real system due to the control signals included, which depend on the environment signaling received at any given time:

$$X_0 \leftrightarrow \tau_I(\partial_H(F \parallel G \parallel H)) \tag{12}$$
$$X_1 \leftrightarrow \tau_I(\partial_H(F \parallel \{s_{\hat{O}} \cdot G\} \parallel H))$$
$$X_2 \leftrightarrow \tau_I(\partial_H(\{(r_{ctl} \cdot s_P + s_O) \cdot F\} \parallel \{s_{\hat{O}} \cdot G\} \parallel H))$$

It is to be noted that the factor r_{ctl} is related to control flows and not to data streams, so it may well be masked when dealing with data transfers. Therefore, it is clear that the model proposed shows the same external behaviour as the real system, meaning that they both are rooted branching bisimilar. And this fact leads to the conclusion that the model proposed gets verified.

VI. Adding up Control at Orchestration Level

Additionally, some higher level information may be brought in about by an orchestration level, which may get information from each fog level facilities by means of a sensor channel (*sen*), which allows the monitoring and responding to some anomalous conditions read through an actuator channel (*act*) to a particular fog level facility.

Furthermore, an extra intelligence channel (*int*) may be put in place in order to get some general information such as traffic congestion nearby, road accidents, unexpected events or adverse weather forecasts. This is shown in Fig. 6, where it may be noted that *int* signal is unique and affects to all fog facilities, whereas each *act* signal affects to just one of them, thus, *int* signal is hierarchically superior to each *act* signal.

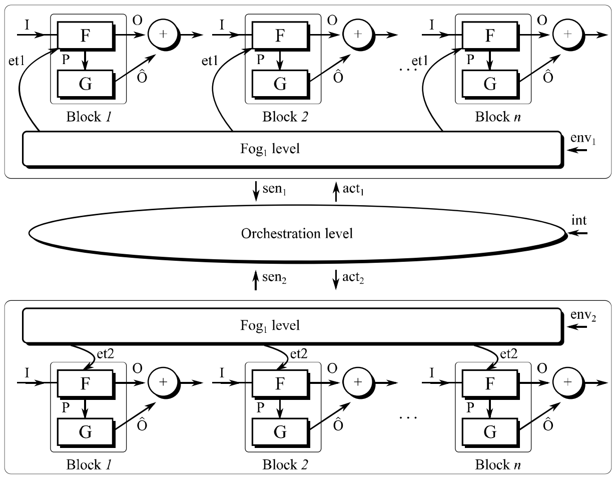


Fig. 6. State diagram adding up control at orchestration level

In this scenario, there are four entities, such as the three ones present in the previous scenario, and the orchestration level, which may be called Z and is common to all fog levels. The task of Z is getting external information through channel *int* and internal information from each fog facility *j* through channels sen_j, and after applying the proper processing, Z may send control signals through channels act_j to all *j* fog facilities, and in turn, those will forward them over to all *i* blocks through the same channels ctl_i described in the previous scenario.

Wrapping it all together, the recursive variables describing the model, using the convention for the previous scenarios, and adding up variable e' to account for intelligence messages ∇', are the following:

$$F = \sum_{d\epsilon\Delta, d'\epsilon\Delta'} r_I(d) \cdot (r_{ctl}(d') \cdot s_P(d) + s_O(d)) \cdot F \tag{13}$$
$$G = \sum_{d\epsilon\Delta} r_P(d) \cdot s_{\hat{O}}(d) \cdot G$$
$$H = \sum_{d'\epsilon\Delta', e\epsilon\nabla} \left(r_{env}(e) \cdot s_{sen}(e) + r_{act}(d') \cdot s_{ctl}(d')\right) \cdot H$$
$$Z = \sum_{d'\epsilon\Delta', e\epsilon\nabla, e'\epsilon\nabla'} \left(r_{int}(e') + r_{sen}(e)\right) \cdot s_{act}(d') \cdot Z$$

In this new context, the Expansion Theorem of Bergstra and Klop customised for four entities may be applied in order to get the behaviour of the overall system.

$$\begin{aligned} Y_1 \parallel Y_2 \parallel Y_3 \parallel Y_4 &= Y_1 \Vdash (Y_2 \parallel Y_3 \parallel Y_4) \\ &+ Y_2 \Vdash (Y_1 \parallel Y_3 \parallel Y_4) + Y_3 \Vdash (Y_1 \parallel Y_2 \parallel Y_4) \\ &+ Y_4 \Vdash (Y_1 \parallel Y_2 \parallel Y_3) + (Y_1 | Y_2) \Vdash (Y_3 \parallel Y_4) \\ &+ (Y_1 | Y_3) \Vdash (Y_2 \parallel Y_4) + (Y_1 | Y_4) \Vdash (Y_2 \parallel Y_3) \\ &+ (Y_2 | Y_3) \Vdash (Y_1 \parallel Y_4) + (Y_2 | Y_4) \Vdash (Y_1 \parallel Y_3) \\ &+ (Y_3 | Y_4) \Vdash (Y_1 \parallel Y_2) \end{aligned} \tag{14}$$

In order for parking to be permitted, it is necessary that both intelligence channel at orchestration level and environment channel at fog level send positive control signals allowing so, otherwise, parking will not be allowed. Supposing such a case, entities H and Z are transmitting control signals to F_i (through channel ctl_i) and H_j (through channel act_j), respectively, in order to permit parking, considering that those coming from Z (intelligence) may disable those coming from H (environment).

Additionally, as Z does not have a direct channel through F nor G, those being the channels where data messages pass through, it means that all control messages to a given block *i* go through channel ctl_i. Considering that all, and supposing that initially, there are no vehicles into a block, these are resulting calculations, omitting the transferred data in the expressions for simplicity purposes:

$$\tau_I(\partial_H(F \parallel G \parallel H \parallel Z)) = r_I \cdot \tau_I(\partial_H(\{(r_{ctl} \cdot s_P + s_O) \cdot F\} \parallel G \parallel H \parallel Z)) \tag{15}$$

$$\tau_I(\partial_H(\{(r_{ctl} \cdot s_P + s_O) \cdot F\} \parallel G \parallel H \parallel Z) = s_O \cdot \tau_I(\partial_H(F \parallel G \parallel H \parallel Z)) + r_{int} \cdot r_{env} \cdot \tau_I(\partial_H(F \parallel \{s_{\hat{O}} \cdot G\} \parallel H \parallel Z))$$

$$\tau_I(\partial_H(F \parallel \{s_{\hat{O}} \cdot G\} \parallel H \parallel Z)) = r_I \cdot \tau_I(\partial_H(\{(r_{ctl} \cdot s_P + s_O) \cdot F\} \parallel \{s_{\hat{O}} \cdot G\} \parallel H \parallel Z) + s_{\hat{O}} \cdot \partial_H(F \parallel G \parallel H \parallel Z)$$

$$\tau_I(\partial_H(\{(r_{ctl} \cdot s_P + s_O) \cdot F\} \parallel \{s_{\hat{O}} \cdot G\} \parallel H \parallel Z)) = s_O \cdot \tau_I(\partial_H(F \parallel \{s_{\hat{O}} \cdot G\} \parallel H \parallel Z)) + s_{\hat{O}} \cdot \tau_I(\partial_H(\{(r_{ctl} \cdot s_P + s_O) \cdot F\} \parallel G \parallel H \parallel Z))$$

With respect to the external behaviour of the real system, the expressions are the same as those presented for a single building block, as explained in the previous section in (7). That is because intelligence and environment signals are in the control domain, whereas external behaviour is related to the data domain. Basically, control signals may delay the transition from one state to another, but by applying the fairness condition, such transitions will eventually happen.

Following the same reasoning, the external behaviour of the model is also the same as that presented for a single building block in (11), therefore, it is given by the first, third and fourth expressions just shown, as deducted previously.

Hence, by comparing both the model behaviour and the real system external behaviour, it is clear that they are both rooted branching bisimilar, hence, leading to the conclusion that the model proposed gets verified.

VII. A Case Study

In order to test the models proposed, a data set of traffic is going to be applied to the different models proposed. First of all, the single building block model with no control traffic will be used, then, control at fog level will be introduced, and finally, control at orchestration level will be added.

To start with the single building block model, it has to be noted that entities F and G deal just with data traffic, those being the vehicles getting into and out of the street patch being modelled through channels I, P, O and Ô. Considering that F and G are supposed to admit just one item at most, hence their possible events are just 0 or 1, Table I shows the whole range of possible outcome and the explanation of each of those events.

TABLE I. NO CONTROL IMPLEMENTED

F	G	Explanation
0	0	No cars in the street nor in the parking
1	0	One car in the street and none in the parking
0	1	No cars in the street but one in the parking
1	1	One car in the street and another one in the parking

After adding up control signals related to the environment at fog level, entity H comes into play, denying the possibility to park even if a parking is available, or otherwise, through the *ctl* channel, meaning that all possible events for H are just 0 (forbid parking) or 1 (permit parking). Table II presents the whole range of possibilities.

TABLE II. CONTROL IMPLEMENTED JUST AT FOG LEVEL

F	G	H	Explanation
x	x	0	No parking is allowed
0	0	1	No cars in the street nor in the parking
1	0	1	One car in the street and none in the parking
0	1	1	No cars in the street but one in the parking
1	1	1	One car in the street and one in the parking

When control signals related to intelligence are added up at orchestration level, entity Z takes kicks in, allowing the possibility to block all parkings available, or otherwise, through the *act* channels pointing to all fog facilities within that orchestration level. The possible events for Z are just 0 (parking not allowed) or 1 (parking allowed, if environment is right). Table III introduces the whole range of options.

TABLE III. CONTROL AT FOG AND AT ORCHESTRATION LEVELS

F	G	H	Z	Explanation
x	x	x	0	No parking is allowed
x	x	0	1	No parking is allowed
0	0	1	1	No cars in street nor in parking
1	0	1	1	One car in street and none in parking
0	1	1	1	No cars in street but one in parking
1	1	1	1	One car in street and one in parking

VIII. Conclusions

In this paper, a smart parking based on IoT/Fog technology have been studied. In that sense, a formal algebraic model has been proposed for different scenarios, such as a single building block and a string of them with no parking restrictions, proving its verification.

Afterwards, the model have been extended with some parking restrictions, first at fog level, and then at orchestration level as well, getting also verified. Finally, a case study has been presented with all possible options for each of the scenarios proposed and its application to real life.

References

[1] H. Ibrahim, "Car Parking Problem in Urban Areas, Causes and Solutions, " in 1st International Conference on Towards a Better Quality of Life, 2017, pp. 1-13.

[2] J.J. Barriga et al., Smart Parking: "A Literature Review from the Technological Perspective", in Applied Science, 2019, Vol. 9(21), Article: 4569, pp. 1-34.

[3] A. Khanna and R. Anand, "IoT based smart parking system, " in Proceedings of 11th International Conference on Internet of Things and Applications (IOTA), 2016, pp. 266-270.

[4] Y. Atif, J. Ding and M.A. Jeusfeld, "Internet of Things Approach to Cloud-based Smart Car Parking," in Procedia Computer Science, 2016, Vol. 98, pp. 193-198.

[5] M.J. Soundharya, N.N. Thore and V.S. Roshni, "Intelligent Parking in IoT Based on Fog Computing, " in International Journal for Research in Applied Science and Engineering Technology, 2019, Vol. 7, Issue 3, pp. 605-611.

[6] T.H. Ashrafi et al, "Service Based FOG Computing Model for IoT," in 3rd International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2017, pp. 163-172.

[7] F. Arute et al, "Quantum supremacy using a programmable superconducting processor," in Nature, 2019, Vol. 574, pp. 505-510.

[8] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," in Artificial Intelligence, 2019, Vol. 267, pp. 1-38.

[9] P. Sethi and S.R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," in Journal of Electrical and Computer Engineering, 2017, article ID 9324035, pp. 1-25.

[10] C.S.R. Prabhu, "Overview - Fog Computing and Internet-of-Things (IoT)," in EAI Endorsed Transactions on Cloud Systems, 2017, Issue 10, Article 5, pp. 1-24.

[11] O. Osanaiye et al, "From Cloud to Fog Computing: A Review and a Conceptual Live VM Migration Framework," in IEEE Access, 2017, Vol. 5, pp. 8284-8300.

[12] D.A. Padua, "Encyclopedia of Parallel Computing," Springer, 2011.

[13] J.F. Groote and M.R. Mousavi, "Modelling and Analysis of Communicating Systems," MIT Press, 2014.

[14] J.A. Bergstra and J.W. Klop, "Algebra of communicating processes with abstraction," in Theoretical Computer Science, 1985, Vol. 37, pp. 77-121.

[15] M. Gazda, W. Fokkink and V. Massaro, "Congruence from the operator's point of view," in Acta Informatica, 2019, pp. 1-23.

[16] L. Lockefeer, D.M. Williams and W. Fokkink, "Formal specification and verification of TCP extended with the Window Scale Option," in Science of Computing Programming, 2016, Vol. 118, No. 1, pp. 3-23.

[17] W. Fokkink, "Introduction to Process Algebra," Springer, 2007.