Multi-Scale Model-based Explanations for Cyber-Physical Systems: the Urban Traffic Case

Ada Diaconescu
ada.diaconescu@telecom-paris.fr
Telecom Paris, IP Paris
Paris, France

Etienne Houze
etienne.houze@telecom-paris.fr
Telecom Paris, IP Paris;
EDF Labs, Palaiseau
Paris, France

Jean-Louis Dessalles
jean-louis.dessalles@telecom-paris.fr
Telecom Paris, IP Paris
Paris, France

Hans Vangheluwe
Hans.Vangheluwe@uantwerpen.be
Universiteit Antwerpen
Flanders Make
Antwerpen, Belgium

Romain Franceschini
franceschini.romain@gmail.com
Lukky
Corte, France

ABSTRACT

Automated control in Cyber-Physical Systems (CPS) generates behaviours that may surprise non-expert users. Relevant explanations are required to maintain user trust. Large CPS (e.g., autonomous car networks and smart grids) raise additional scaleability issues for the explanatory processes and complexity issues for generated explanations. We propose a multi-scale system modelling and explanation technique to address these concerns. The idea is to increase the scale, or abstraction level, of the modelled CPS, whenever possible without loss of salient information, so as to produce smaller system representations and hence to reduce the complexity of the explanatory process and of the generated explanations. We illustrate our proposal via an urban traffic case study, modelling traffic at two different scales (i.e., modelling individual cars at a lower-scale; and traffic jams at a higher-scale). We show how a multi-scale explanatory process can use the lower- and higher-scale models to generate either longer (more detailed) explanations, or shorter (more abstract) explanations, respectively. This proof-of-concept illustration offers a basis for further research towards a comprehensive multi-scale explanatory solution for CPS.

KEYWORDS

multi-scale model and explanation, traffic simulation, cyber-physical system

MODELS '22 Companion, October 23–28, 2022, Montreal, QC, Canada

ACM ISBN 978-1-4503-9467-3/22/10…$15.00
https://doi.org/10.1145/3550356.3561554

ACM Reference Format:
Ada Diaconescu, Etienne Houze, Jean-Louis Dessalles, Hans Vangheluwe, and Romain Franceschini. 2022. Multi-Scale Model-based Explanations for Cyber-Physical Systems: the Urban Traffic Case. In *ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems (MODELS '22 Companion), October 23–28, 2022, Montreal, QC, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3550356.3561554

1 INTRODUCTION

Cyber-physical systems (CPS), such as autonomous cars, smart homes and power grids, are increasingly equipped with Artificial Intelligence and autonomic controllers. This shifts numerous adaptation decisions from the user to the CPS (e.g., tuning a car's speed depending on traffic; or scheduling a smart device's usage to minimise consumption). Such automation may lead to cases where CPS behaviour surprises users, who notice a difference between the observed and the expected CPS behaviour (e.g., why is my car driving so slowly? why did the automatic blinds go down?). To maintain user trust, CPS must provide pertinent explanations to such questions, case-by-case [1]. The EU General Data Protection Regulation (GDPR) goes as far as to evoke the "right to explanation" for users of AI systems [10].

An increasing amount of research is carried-out in this direction under the umbrella of Explanatory AI (XAI). The focus is often on "opening the black box" of opaque AI models (i.e., neural networks). LIME [16] or SHAP [13] methods for example propose to explain classifier outputs in terms of feature relevance – i.e., determine which feature of the input data was most discriminating in the classifier result (e.g., a person's age was the prime decision factor in a bank's loan validation algorithm). Several XAI approaches were proposed for various AI models and applications [3]. Still, most XAI techniques feature significant limitations. Notably, they aim to explain static AI models and hence cannot adapt to run-time changes. This drawback becomes severe in CPS, which often evolve by adding, updating or removing components (e.g., cars joining and leaving platoons; or smart devices plugged in and out of smart micro-grids). Moreover, most

XAI explanations are unsuitable for end-users, as they require low-level domain-specific expertise [14].

To address these issues, we previously proposed a generic adaptive (but not multi-scale) approach for generating intuitive explanations for users: Decentralised Conflict-Abduction-Simulation (D-CAS) [11], [7] (Cf. sec. 3.2). D-CAS generates explanations on-the-fly, by dynamically selecting Local Explanatory Components (LECs) from a flexible pool and composing their partial explanations into coherent answers. Each LEC is linked to a CPS resource and holds specific expertise for explaining that resource. When users ask a question, a generic D-CAS coordinator, called Spotlight, forwards the question to a LEC that holds relevant expertise for that question. The LEC is identified based on key terms, or 'predicates', in the question. Based on this LEC's answer (again, using employed 'predicates'), the Spotlight identifies another LEC to question for further explanations, sequentially, to the LECs that hold relevant expertise for that question. The Spotlight identifies these LECs dynamically (and is hence adaptive), one-by-one: first based on key terms (or 'predicates') in the user question; and then based on the predicates in the previous LEC's partial answer. The process is recursive and guaranteed to end, at the latest, when all relevant LECs have provided all their partial answers. At that point, the Spotlight returns all partial explanations (composed in a sequence) that it received from all questioned LECs.

While promising to address the above limitations, D-CAS raises further issues when applied to large CPS (e.g., urban traffic analysis involving tens of thousands of cars; or smart grids interconnecting millions of smart devices). In such cases, both the *scalability* of the explanatory process and the *complexity* of explanations provided to users may severely reduce the usability of the approach. In this position paper, we propose a *multi-scale modelling and explanation* approach to start addressing the above issues. We base our proposal on the fact that scalability and complexity issues in large CPS partially stem from the sheer number of CPS resources, and their associated explanatory concepts, that must be considered as input / output to the explanatory process. We aim to reduce this number via a multi-scale approach [5], [6], applied to the XAI domain.

In brief, we consider an *explanation* as a *statement*[1] – i.e., set of words, arranged sequentially according to some grammar, and providing some semantics, or meaning, to the user recipient. We further associate certain *words*[2] – such as nouns, verbs, adjectives and adverbs – to *explanatory concepts* – i.e., representations, conceived in the mind, of either: abstract objects (e.g., ideas, principles and notions); or of concrete cyber-physical objects (e.g., autonomous cars and smart devices). This notion of explanatory concept corresponds, for all practical purposes, to a type in a system's meta-model. Some might argue that in some cases an "open-world" framework based on ontologies might be more appropriate. Given our use of simulation, whereby a simulator implements the operational semantics of a language whose abstract syntax is specified by means of a meta-model, a "closed-world" framework based on meta-modelling seems more suitable.

Further, we consider explanatory concepts to represent objects at various abstraction levels, or *scales* (Cf. below). We then propose to reduce the length of complex explanations, when possible, by replacing several low-scale concepts (i.e., representing less abstract or smaller objects) by a single higher-scale concept (i.e., representing a more abstract or larger – often aggregate – object). The higher concept is supposed to convey the same kind of information as the set of lower concepts it replaces, except in a more concise manner. This simplifies explanations to users, while maintaining relevant semantic content, under those conditions that the scale change is a proper abstraction (i.e., preserving information concerning properties of interest that is relevant in a given context, and forgetting information that is irrelevant in that context, concerning those properties of interest).

Generally, we define a *scale* as the *granularity of observation* of an object [5], [6]. E.g., traffic in an urban area can be observed at the granularity of every vehicle, of discrete platoons, or of continuous congestion level (density). A coarser granularity corresponds to a higher abstraction level – providing less detailed information about the observed object – compared to a lower granularity. In this context, the notion of *multi-scale* refers to the simultaneous observation of an object at multiple scales. E.g., urban traffic can be modelled simultaneously: at the scale of each car, of each platoon and of its congestion level.

We apply the multi-scale notion to explanatory concepts as follows. A lower-scale explanatory concept refers to an object observed at a finer granularity (i.e., less abstract, often smaller in space, or shorter in time), relative to a higher-scale concept that refers to an object observed at a coarser granularity (i.e., more abstract, often larger in space, or longer in time). E.g., in urban traffic, the 'car' concept is lower-scale relative to the traffic 'jam' concept, as it involves a finer observation granularity, at a smaller spatial scope. We employ multi-scale explanatory concepts to attain scalable explanations as follows. We consider that, to achieve its purpose efficiently, an explanation must be: i) 'relevant' – i.e., solve the problem raised by the question that demanded the explanation; and, ii) as simple as possible (but not simpler) – i.e. be expressed via the shortest description that is still relevant. Hence, we propose to shorten explanations by employing words associated with the highest-scale concepts, which still ensure the explanation's relevance to the user.

We illustrate the viability of this approach via a concrete case study targeting urban traffic (Fig. 1). First, we employ a multi-agent simulation (subsec. 2.2) to provide an adaptive *multi-scale model* of urban traffic. In the model, low-scale concepts refer to individual *cars* and higher-scale concepts to traffic *jams*. Further, we extend our D-CAS implementation [7] with traffic-specific LECs, customised for cars and jams.

[1] Statement definition based on the Sentence definition from the Oxford dictionary online: "a set of words expressing a statement, a question or an order, usually containing a subject and a verb."
[2] Word definition from the Oxford dictionary online: "a single unit of language that means something and can be spoken or written."

This D-CAS version is integrated with the traffic model via monitoring data (i.e., the model provides information about the positions and speeds of cars and jams on the roads).

A user can ask questions about the traffic to the explanatory system such as "why is my car driving so slowly?". We illustrate the multi-scale aspect of the explanatory system by providing alternative answers. The first one uses the explanatory concept of 'car', and states that: the car is slow because the car in front is slow, and so on, recursively, until a car at the end of the line is found to have a mechanical problem – this explanation employs low-scale concepts resulting in a long statement. The second explanation variant uses the explanatory concept of 'jam': the car is slow because there is a jam in front, which is due to a front car that has a mechanical problem – this variant employs higher-scale concepts resulting in a more concise response. This proof-of-concept illustration provides an encouraging basis for further developing our multi-scale modelling and explanation approach, so as to provide a scalable solution for explainable CPS.

2 APPROACH OVERVIEW

2.1 Generic Architecture

Fig. 1 depicts our generic system design. An *Observed System* (i.e., urban traffic) is represented via a *System Model* to highlight properties of interest (e.g., traffic fluency) under various conditions (e.g., car number). An *Explanatory System* enables *Users* to ask questions about the Observed System, at run-time. To respond, the Explanatory System relies upon:

(1) monitoring data from the System Model, which it maps into its domain-specific explanatory concepts, called *predicates* (Cf. subsec. 3.2.3) – e.g. mapping car 'speed' data into 'slow' or 'fast' predicates;
(2) a reasoning process based on D-CAS and on expert-specific abduction processes (Cf. subsec. 3.2.2).

The System Model may change dynamically to reflect changes in the Observed System. The Model relies on concepts (i.e., meta-models) that may also vary dynamically, to improve the effectiveness and efficiency of the System's representation with respect to targeted properties of interest within various contexts – e.g., via adaptive abstractions.

This follows Multi-Paradigm Modelling (MPM) principles: to explicitly model all parts and aspects of a system, using the most appropriate formalism(s) and abstraction(s). This includes the explicit modelling of often complex workflows [2].

2.2 Adaptive Multi-scale Model for Urban Traffic

To illustrate our general proposal for multi-scale modelling and explanations, we consider a concrete case study: urban traffic. The transport network is modelled as a graph, with nodes as departure/destination points and edges as routes between them. Vehicles circulate at various speeds, which depend on the network topology, their intended destinations and interaction with other vehicles and road signs. When too many cars attempt to pass through one node, they must wait for each other, hence lowering their speeds and forming a traffic jam.

This basic case study is relevant to our approach as it allows traffic modelling at various scales, which may change over time. When traffic is fluent, individual cars may travel at various speeds. Hence, each car is modelled separately, forming a low-scale model. Conversely, when cars are caught in a traffic jam, they move at approximately the same speed. Hence, a higher-scale model can aggregates all the cars in a jam. When the traffic becomes fluent again the model switches-back to its lower-scale version. Modelling adaptive abstraction requires

- a model of the detailed *dynamics* (i.e., how the state of the system evolves), in our case, describing individual car dynamics;
- a model of a *monitor* which checks whether the conditions for switching to a less detailed (more abstract) model of the dynamics, are satisfied. In our case, if a collection of cars are close to each other and have almost the same velocities, and this for an extended period of time, this collection of cars can be replaced by a single jam;
- a model of how to *transfer* the states of the cars in the collection to the initial state of the new jam;
- a model of the less detailed dynamics, in our case describing jam dynamics (including how jams interact with cars, with other jams, with traffic lights, ...).
- a model of a *monitor* which checks whether the conditions for using the jam abstraction are still satisfied. As traffic conditions change, the distribution of locations and velocities of cars in the jam (encoded in the jam's state) may become increasingly smeared out. At some point, the jam abstraction no longer makes sense;
- a model of how to *transfer* the state of a jam to the initial states of individual cars the jam is made up of.

The above, combined with a simulation engine, gives an adaptive abstraction simulation which offers both improved simulation performance and higher explainability. For full details on adaptive abstraction and our implementation using the NetLogo agent-based platform (NetLogo[3]) implementation, see [9].

3 GENERATIVE EXPLANATIONS: BACKGROUND AND MULTI-SCALE EXTENSION

3.1 Conflict Abduction Negation (CAN) Explanation Method

According to [4], a user typically requires an explanation to resolve a *conflict* between an expected and an observed situation (e.g., traffic slower than expected). An explanation consists of a chain of explanatory concepts, which represent *causes* [15] of the perceived conflict. The explanatory process identifies each cause individually, in sequence. It can

[3]NetLogo homepage: https://ccl.northwestern.edu/netlogo/

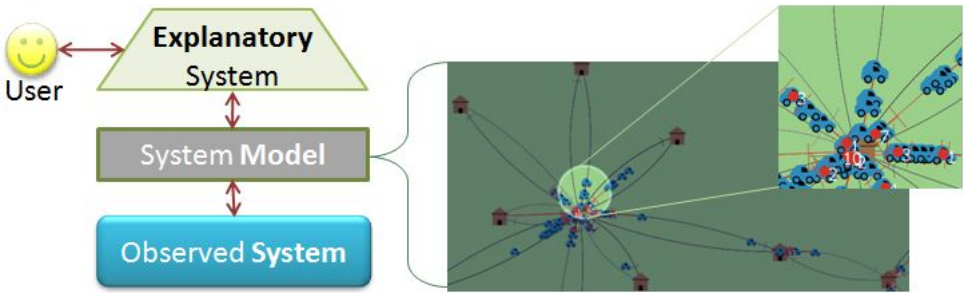


Figure 1: Generic System Architecture: an Observed System is represented by a Model, which provides a basis for an Explanatory System for Users. Our traffic case study uses a multi-agent NetLogo traffic model.

then return the resulting causal chain to the user as an explanation for the perceived conflict. [4] proposes a generic abductive method that can be employed to generate such explanations – *Conflict-Abduction-Negation (CAN)*. CAN formalises the above explanatory process via four generic steps (to be implemented specifically for each system), executed sequentially:

- **conflict**: detects a discrepancy between expectation and observation and associates a *necessity* (intensity) to it;
- **abduction**: identifies the most probable cause of the conflict (using various abduction methods);
- **negation**: considers situations without the conflict ('counterfactuals' [15]) and evaluates the consequences;
- **solution**: solves the conflict by reconsidering knowledge (e.g., false-positive conflict), by acting on the world (e.g., change the conflicting state) or by abandoning the conflict (i.e., avoid blockage or infinite loops).

While providing a suitable reasoning process to obtain explanations, CAN features several limitations when implemented as a centralised, monolithic process. Notably, it cannot deal with highly heterogeneous and dynamic systems (e.g., traffic systems), where various resource types may be included or removed dynamically. To address this requirement, we proposed a modular, decentralised CAN version, called D-CAS.

3.2 Decentralised Conflict Abduction Simulation (D-CAS) Explanation Method

3.2.1 Key Design Objectives. To ensure the explanatory system's adaptability to the observed system, we aim to:

(1) *Avoid hard-coded ontology*: the employed vocabulary (i.e., words and their semantics) should evolve at run-time to reflect changes in the observed system (e.g., adding a 'queue' to a traffic model requires a new word); as well as changes in the user's perception (e.g., user- and context-dependent semantics of the word 'slow').
(2) *Avoid hard-coded reasoning rules*: the employed abduction logic should be adaptable at run-time, e.g., to consider vocabulary updates and include new reasoning methods.

These design objectives ensure the flexibility of the explanatory process – in contrast to methods where both vocabulary and reasoning are tightly coupled to the observed system. D-CAS proposes a generic framework that supports such dynamic vocabulary and reasoning processes. While its current prototype still hard-codes vocabulary and provides simple abduction methods, these initial implementations can be replaced seamlessly by more flexible variants (Cf. [7]).

To achieve objective (1), system monitoring data is decoupled from explanatory concepts via *events*, *predicates* and *propositions* (subsec. 3.2.3). These are specified in the Interpretation Module of every LEC. To achieve (2), each LEC supports a plug-and-play set of Abduction Modules, each including a specific kind of abduction method (e.g., [7], [8]).

3.2.2 D-CAS Design. Decentralised Conflict, Abduction, Simulation (D-CAS) is a decentralised version of the CAN method. It proposes to implement CAN's Negation step via Simulation. D-CAS features a modular plug-and-play design (Fig. 2), where vendor-specific expert components – called *Local Explanatory Components (LECs)* – can be integrated and employed within the explanatory process on-demand, whenever relevant to an explanation. We associate one LEC with each type of observable system resource that is of interest to the explanatory system. Each LEC holds:

(1) a *vocabulary* (i.e., a set of explanatory concepts, or 'predicates') for statements about the associated resource (e.g., 'car' and 'slow' for a Car-specific LEC). The LEC *interprets* these predicates dynamically, based on monitoring data from the system model (e.g., a car is considered slow if its monitored speed is lower than a threshold);
(2) a set of *abduction* techniques for finding the causes of conflicts that are related to the associated resource (e.g., a problem's cause is: the last memorable event [7]; same as last time; indicated by a causal Bayesian network [8]).

To select LECs that are relevant to each explanation and to compose their answers into a coherent response for the user, we introduce a generic coordinator, called Spotlight. The

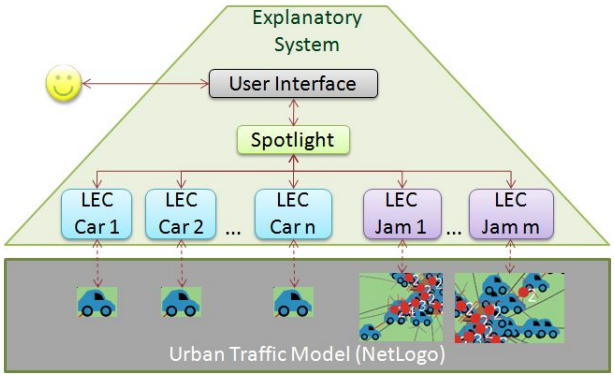


Figure 2: D-CAS Generic Architecture, exemplified for the Urban Traffic case study

Spotlight's role is similar to that of a Naming and Directory Service, or Repository (e.g., Yellow Pages). It keeps track of all LECs in the system and of the kind of problems, or conflicts, that they can address. In brief, when the user questions the Spotlight, the latter: i) identifies the LECs that hold the expertise relevant to that question; ii) questions the identified LECs sequentially; and, iii) returns their composed answers to the user, as a single coherent statement. Importantly, the Spotlight ignores all details related to the system resources and their explanations (these are only encoded into the LECs' Interpretation and Abduction Modules). This means that though centralised, the Spotlight remains a lightweight and generic entity that does not need to evolve and adapt to various systems, or to their run-time changes. D-CAS approach is similar to an expert system that relies on a set of problem-specific experts, which can be added, removed and updated at run-time. To answer a question (or solve a problem) the relevant experts can be identified and questioned case-by-case. This modular and loosely-coupled design helps to address some of the scalability and adaptability issues inherent in the original CAN version.

The formal D-CAS process for LEC selection and coordination is listed in Algorithm 1. In more detail, the Spotlight receives a question in the form of a conflict (subsec. 3.1). It identifies the LEC holding expertise about that conflict (based on contained predicates) and forwards the question to this LEC. E.g., the question 'why a car is slow' will go to a Car-specific LEC, provided by the car vendor (as it is the expert on its cars). Next, the LEC checks whether the conflict is valid based on its monitoring data (e.g., is the car slow according to its speed?). If the conflict is invalid, the LEC returns this fact to the Spotlight, which informs the user. Otherwise, the LEC uses its abduction logic to determine the cause of the conflict (e.g., the car in front is also slow). The LEC returns the cause to the Spotlight, as a new conflict, which will be forwarded to an appropriate LEC.

The process continues, recursively, forming a sequential reasoning chain, until a LEC provides an answer that can no longer be followed (e.g., invalid conflict; or, conflict solved by an action; or, no further expertise is available to continue the reasoning chain). This guarantees the end of any reasoning chain. The causal chain obtained from the LECs' questioning sequence is returned to the user as an explanation (e.g., Fig. 3). When a reasoning chain is completed, the Spotlight may also question previous LECs again to provide an alternative cause. This starts a new reasoning chain and results in a tree-like topology (rather than a line) for the Spotlight's questioning sequence – not shown here, Cf. [7].

Input: A request `req` from the user
Result: A conflict-solving process whose trace can be exposed as an explanation
Data: Pointers to the LECs in the system
C a set of examined conflicts, G a set of considered give-ups

```
(P, N) ← analyzeRequest(req);
responsible ← locate(P);
while responsible ≠ self do
    if responsible = null then
        Backtrack() ;
    end
    answer = responsible.investigate((P, N)) ;
    switch answer do
        case ABDUCTION do
            (P, N) ← Answer.Hypothesis ;
            responsible ← locate(P) ;
        end
        case GIVE UP do
            Backtrack();
        end
        case ACTION do
            simulator.run(Answer.Action) ;
            Conflict ← waitForProblems() ;
        end
    end
end
```

Algorithm 1: D-CAS Algorithm: the Spotlight successively considers conflicts and forwards them to relevant LECs

3.2.3 D-CAS Conceptual Formalism: Events, Predicates and Propositions. *Predicates* are Boolean functions over *events*, which are issued from the observed system's data [12], [7]. Evaluating a predicate results in a Boolean *proposition*. E.g., in the urban traffic case study: a car's model provides monitoring data about its speed; a Car LEC's $Slow()$ predicate evaluates this data and returns a proposition $Slow = true$ if the car's speed is lower than a threshold; and a proposition $Slow = false$ otherwise. Hence, predicates provide the mapping function for dynamically transforming the observed system's data into the explanatory system's vocabulary. While in the current implementation these predicates are hard-coded, future versions will replace these by dynamically defined functions (e.g., via online learning and dialogue).

3.3 Multi-scale Explanations

To support multi-scale explanations, D-CAS simply needs to integrate multi-scale explanatory concepts (i.e., predicates) into its vocabulary. It may then reuse its existing abduction methods to reason upon the new higher-scale concepts. Alternatively, it may integrate new abduction methods that are specific to these higher concepts. The above extensions only require to provide new LECs for the higher-concepts; the rest of the D-CAS framework stays unchanged. The new LECs may also reuse the generic LEC implementation provided by D-CAS framework; and plug into it the specific Interpretation and Abduction Modules for holding the higher-scale vocabulary and abduction logic, respectively.

4 TRAFFIC SIMULATION CASE-STUDY

4.1 Multi-scale Model

To provide the multi-scale system model, we relied on a multi-agent implementation (in NetLogo) of the traffic case study in subsec. 2.2. The simulation provides adaptive abstractions [9] (or scales) in the sense that it models each car individually when traffic is fluent (i.e., low-scale concept) and aggregates all cars caught-up in traffic congestion into a single jam (i.e., higher-level concept). One version (following the "puppeteer" pattern) of the simulation keeps both high- and low-scale models in parallel, so it can seamlessly switch between them as the traffic context changes. This does not increase simulation performance but makes generating explanations at different levels easier.

The model provides data about each simulated car (e.g., car_{id}, spatial coordinates, speed, travelled road). When detecting a traffic jam ,the simulation adds to the model a virtual aggregated car that represents all cars in the jam. Jam data includes the jam_{id}, position in space, affected road, number and identity of contained cars. In one of our small experiments, wee ran the simulation with 200 cars and 15 destinations linked in a star-like road network (Fig. 1); we used traffic data from one of the roads. Initially, cars are distributed randomly in the destinations and start moving randomly via the roads available at each destination. Runtime simulation data is stored in a log file, which is then provided as input to the Explanatory System. An online architecture whereby simulation data is taken from a running simulator is also possible.

4.2 Multi-scale Explanation: Implementation, Experiments and Results

4.2.1 Design and Implementation overview. We reused the generic Explanatory System implementation (in Python) in [7] and customised it with domain-specific LECs for the urban traffic case (Fig. 2). We implemented two types of customised LECs: Car LEC and Jam LEC. These are based on the generic LEC implementation [7] with specific modules for Abduction and Interpretation (i.e., specific reasoning and vocabulary, respectively). LEC instances were created for each car and jam (i.e., by hand here but can be done dynamically upon automatic car / jam detection in the model).

4.2.2 Predicates in the Interpretation Modules. The following predicates were defined in the Interpretation modules of Car and Jam LECs, respectively (i.e., the first four belong to the Car LEC; the last two to the Jam LEC):

(1) $Car(car_{id})$: true if the the car has the given ID;
(2) $AheadOf(car_{id})$: true if the car is in front of another car with the given ID;
(3) $Slow()$: true if the car to which the question is asked moves at a speed that is lower than a given threshold (i.e., 0.05 in the experiments, based on the simulation values);
(4) $OnRoad(road_{id})$: true if the car to which the question is asked drives on a road with the given ID;
(5) $Jam(jam_{id})$: true if the jam has the given ID;
(6) $ContainsCar(car_{id})$: true if the jam contains the car with given ID.

4.2.3 Causal Propositions in the Abduction Modules. The Car LEC's Abduction Module specifies two alternative *causal propositions* (note: not to be confused with Boolean 'propositions' in the vocabulary) to determine the causes for a slow car: either due to a slow car in front ($CausalProposition_A$), or because there is a jam in front ($CausalProposition_B$). A further proposition ($CausalProposition_C$, not simulated) states that a jam's first car (i.e., the car with $car_{id} = 95$ in our example) is slow because of a $MechanicalFailure$. More precisely, the Car LEC's Abduction Module provides the following alternative causes for questions containing the $Slow$ predicate:

- $CausalProposition_A$: the cause is the car in front of the car in question (i.e. via the $AheadOf$ predicate);
- $CausalProposition_B$: the cause is the jam that contains the car in question (i.e. via the $ContainsCar$ predicate);
- $CausalProposition_C$: if the car in question has $car_{id} = 95$ then the cause is a $MechanicalFailure$.

The Jam LEC's Abduction Module includes a causal proposition ($CausalProposition_D$) which states that the jam is slow because of the first slow car that it contains (i.e., car_{95} in our example). Hence, when asked a question that contains the $Slow$ predicate, the Jam LEC's Abduction Module provides the following cause:

- $CausalProposition_D$: the cause is the first car in the jam, which is car with $car_{id} = 95$ in our simulation.

4.2.4 D-CAS Experiments. In brief, the overall explanatory process was obtained as follows. The traffic simulation (subsec. 4.1) was executed and monitoring data on cars and jams were recorded in a log file. This log data was then transferred to the Explanatory System's knowledge database (BDD Redis) (Cf. details in [7]). We then sent commands to the Explanatory System to create the Car and Jam LECs, configured with the special-purpose Interpretation and Abduction modules

specified above. At this point, the Explanatory System was operational and ready to answer questions. Before running an experiment, we select, by hand, one of the two causal propositions for the Car LEC's abduction module. This choice should be performed automatically in the future; the current experiments are illustrative. This results in two kinds of experiments:

- *Lower-Scale Explanation Experiment*: uses $CausalProposition_A$ to determine causes based on individual cars;
- *Higher-Scale Explanation Experiment*: uses $CausalProposition_B$ to also employ the higher-scale jam concept.

At the start of each experiment, we asked the following question to the Explanatory System's Spotlight: "$Why\ Car(422)\ Slow()$", where 422 was the car_{id} of a car that entered a jam in the traffic simulation. This question may be asked by the driver of the car in question. The Explanatory System returned one of the alternative explanations in Fig. 3. In both experiments, the Spotlight received the question and forwarded it to the LEC of car_{422} – the object of the question. Afterwards, the answer given depended on which casual proposition was selected in the Car LEC's Abduction Module.

In the *Low-Scale Explanation Experiment*, by means of the $CausalProposition_A$: the Car LEC's Abduction Module states that the cause of a Slow car status (i.e., the car's *Slow* proposition is 'true') is the car in front of the car in question. Applying the $AheadOf$ predicate determines that the car in front is car_{419}. Hence, the Spotlight forwards the correspondingly updated question to this car's LEC ("$Why\ Car(419)\ Slow()$"). Thus, the explanatory process repeats recursively, identifying all slow cars in the traffic chain, one by one, and forwarding the updated question to them. The process halts when reaching car_{95}, which is the first one in the jam. Its Abduction Module states (via $CausalProposition_C$) that the cause of the *Slow* state is a $MechanicalFailure$. As this cause does not correspond to any predicate that can be handled by the available LECs, the Spotlight halts the explanation and returns the list of causes identified so far (Fig. 3a).

Alternatively, in the *Higher-Scale Explanation Experiment*, by means of the $CausalProposition_B$: the Car LEC's Abduction Module states that the cause of a Slow car status is the 'jam' containing the car in question. Applying the $ContainsCar(422)$ predicate identifies the corresponding jam_{410}. Fig. 3b depicts the explanation obtained so far. If the User further wishes to know the reason for the identified jam, they may re-inquire the Spotlight ("$Why\ Jam(410)\ Slow()$"). The Spotlight forwards the question to LEC of jam_{410}. The Jam LEC's Abduction Module ($CausalProposition_D$) states that the cause is the first car in the jam (i.e., car_{95}). The Spotlight forwards the question to car_{95}, which states that the cause is a $MechanicalFailure$. The process ends and the sequence of causes is returned as explanation to the user (Cf. Fig. 3c).

These examples show how multi-scale explanations can adjust the level of detail for explanations provided to users. Higher-scale explanations (using jams) provide shorter, more concise explanations; while lower-scale explanations (using cars) provide longer, more detailed explanations.

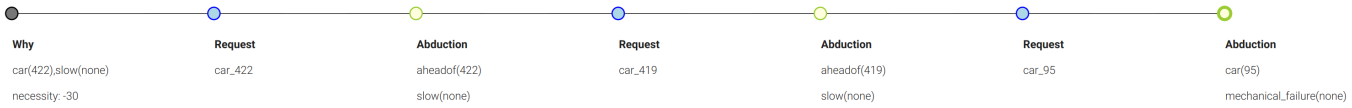
5 CONCLUSIONS AND FUTURE RESEARCH

This position paper proposed to use multi-scale models as a means to generate multi-scale explanations about observed systems. This multi-scale approach aims to deal with scalability and complexity issues when explanations target very large systems. The key idea is to model systems via fewer higher-scale concepts, so as to replace, via abstraction, several lower-scale concepts. Higher-scale models reduce the amount of system facts to consider when generating relevant explanations; and also simplify explanations returned to users. As the appropriate scale for system modelling may vary dynamically, we propose to use adaptive multi-scale models and associated explanatory processes that can adjust the scales employed at run-time.

We illustrated our proposal through a multi-scale simulation of urban traffic (implemented in NetLogo). We extended an explanatory framework from previous work (i.e., D-CAS) to include multi-scale concepts for the traffic case. We exemplified the system's multi-scale explanatory process by asking a question about a slow car. Two alternative answers were provided – a lower-scale one, considering all individual cars; and a higher-scale one, considering jams as aggregates of several slow-moving cars. This illustrated how the higher-scale explanation was considerably shorter and simpler to produce than the lower-scale one.

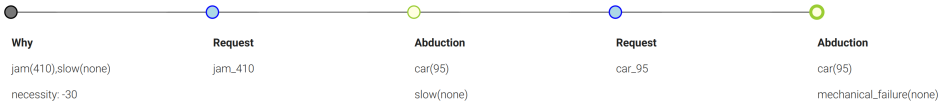
Future work will focus on determining the appropriate explanation scale for each question, within each given context; and switching between scales automatically during run-time. Other aspects of the D-CAS framework will also be studied to provide more advanced versions – e.g., automatic identification of higher-scale explanatory concepts and multi-scale abduction methods. The abduction logic is currently hard-coded, yet could be implemented in a dynamic manner, e.g., via online learning or contrast-based methods. Initially, the Explanatory System can dispose of all alternative causal propositions and select among them based on a given criteria. Such causal propositions can also be provided at run-time – e.g., when the modelling system introduces the 'jam' concept to provide a higher-scale traffic representation, the Jam LEC can be automatically included into the explanatory system and provide its alternative jam-based explanation.

An important criterion to select between multiple explanation scales will rely on the fact that the answer to a question should resolve the conflict raised by that question, while featuring minimal complexity (e.g., as defined by Algorithmic Information Theory). Intuitively, the explanation based on the jam is simpler than the one based on the chain of individual slow-moving cars, because its description is shorter. Once the simplest explanation is given initially, the user may



(a) low-scale explanation: individual cars only

(b) higher-scale explanation: jam only



(c) higher-scale explanation: jam's cause

Figure 3: Explanation for a driver on why her car is slow: a) low-scale: long chain of individual cars moving slowly in front (more detailed, complicated explanation); b) higher-scale: traffic jam in front (more abstract, simpler explanation); c) jam cause (hybrid)

require further details by asking more questions (e.g., 'why is there a jam' or 'how long is the jam'). This proposal sets a preliminary basis for developing more comprehensive multi-scale explanation solutions for large CPS. We assume that LECs are provided by CPS resource vendors (e.g., a car-specific LEC, containing relevant explanatory predicates and logic, bundled within an autonomous car). It is recognised that if vendors do not provide these LECs, our technique is not usable in practice. Given the increasing demand for explainability, we believe that market demand will force vendors to include LECs in their products. We realise that the Spotlight is a potential bottleneck which might impede scalability. Note that a large part of the scalability problem is addressed by the multi-scale approach. A hierarchical organisation of Spotlights should however still be investigated.

ACKNOWLEDGMENTS

The work of H. Vangheluwe was partially funded by Flanders Make. The work of R. Franceschini was carried out while a postdoctoral researcher at the University of Antwerp, funded by a fellowship of the University of Corsica (France).

REFERENCES

[1] 2016. *Explainable Artificial Intelligence*. Broad Agency Announcement DARPA-BAA-16-53. DARPA.

[2] Moussa Amrani, Dominique Blouin, Robert Heinrich, Arend Rensink, Hans Vangheluwe, and Andreas Wortmann. 2021. Multi-paradigm modelling for cyber–physical systems: a descriptive framework. *Software and Systems Modeling* 20, 3 (2021), 611–639. https://doi.org/10.1007/s10270-021-00876-z

[3] Alejandro Barredo Arrieta and et. al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82–115. Publisher: Elsevier.

[4] Jean-Louis Dessalles. 2016. A Cognitive Approach to Relevant Argument Generation. In *Principles and Practice of Multi-Agent Systems*. 3–15.

[5] A. Diaconescu, L. Di Felice, and P. Mellodge. 2021. An Information-oriented View of Multi-Scale Systems. In *IEEE SISSY / ACSOS*. 154–159.

[6] Ada Diaconescu, Louisa Jane Di Felice, and Patricia Mellodge. 2021. Exogenous coordination in multi-scale systems: How information flows and timing affect system properties. *Future Generation Computer Systems* 114 (2021), 403–426. https://doi.org/10.1016/j.future.2020.07.034

[7] J.L. Dessalles E. Houze†, A. Diaconescu and D. Menga. 2022. A generic and modular architecture for self-explainable smart homes. In *IEEE ACSOS*.

[8] Kanvaly Fadiga, Etienne Houzé, Ada Diaconescu, and Jean-Louis Dessalles. 2021. To do or not to do: finding causal relations in smart homes. In *IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. IEEE, Online, 110–119.

[9] Romain Franceschini, Bentley James Oakes, Simon Van Mierlo, Moharram Challenger, and Hans Vangheluwe. 2020. *Towards Adaptive Abstraction for Continuous Time Models with Dynamic Structure*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3417990.3421443

[10] B. Goodman and S. Flaxman. 2017. EU Regulations on Algorithmic Decision-Making and a "Right to Explanation". *AI Magazine* 38, 3 (2017), 50–57.

[11] Étienne Houzé, Jean-Louis Dessalles, Ada Diaconescu, David Menga, and Mathieu Schumann. 2022. A Decentralized Explanatory System for Intelligent Cyber-Physical Systems. In *Intelligent Systems and Applications*, Kohei Arai (Ed.). Springer International Publishing, Cham, 719–738.

[12] Étienne Houzé, Jean-Louis Dessalles, Ada Diaconescu, and David Menga. 2022. What Should I Notice? Using Algorithmic Information Theory to Evaluate the Memorability of Events in Smart Homes. *Entropy* 24, 3 (2022). https://doi.org/10.3390/e24030346

[13] S.M. Lundberg and S.I. Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Info. Processing Systems*. 4765–4774.

[14] Tim Miller, Piers Howe, and Liz Sonenberg. 2017. Explainable AI: Beware of inmates running the asylum or: How I learnt to stop worrying and love the social and behavioural sciences. *arXiv preprint arXiv:1712.00547* (2017).

[15] J. Pearl. 2018. The Book of Why: The New Science of Cause and Effect. *Science* 361, 6405 (2018), 855–855.

[16] M.T. Ribeiro, S. Singh, and C. Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In *ACM SIGKDD ECAI*. 1135–1144.