A calculus for multi-level emergent behaviours in component-based systems and simulations

Chih-Chun Chen[1], Sylvia B. Nagl[2], and Christopher D. Clack[3]

[1] Department of Computer Science, University College London
[2] Department of Oncology and Biochemistry, University College London
[3] Department of Computer Science, University College London

Summary. A major issue in Complexity Science is the formal description of emergent properties and behaviours in terms of lower level properties and behaviours. As a consequence, there are few techniques for empirically investigating specific emergent properties. In this paper, we introduce a general compositional approach to specifying such properties, using constraints to define representative sets of compositions. More specifically, we propose a calculus of complex events, which are compositions of events generated from component-level rule executions. Complex event types can be assembled hierarchically, giving a formal means of relating behaviours at different levels of abstraction. In being able to specify and then identify complex events of different types in systems and simulations, we have a method for empirically discovering relationships between behaviours defined at different levels. The formalism offers two important practical advantages. Firstly, higher level properties can be defined with different degrees of specificity so they can be defined with limited knowledge; we can then further sub-classify properties after they have been detected to discover differences in their constituent properties. Secondly, the formalism is related directly to the rules driving component behaviour so that all higher level behaviours can ultimately be decomposed into rule executions; this is particularly important for desirable and dysfunctional properties, and in circumstances where intervention at the component rule level is possible.

1 Introduction

Being able to understand and precisely analyse emergent behaviours in complex systems is a key goal in both Complexity Science and complex systems engineering. Emergent behaviours are those that arise from the organisational relationships and dynamic interactions between the system's components, and which are then able to drive the trajectory of the system in a particular direction. Information theory has provided a formal framework for defining emergence, and techniques for establishing the degree of emergence in a given system have been derived from this interpretation (see, for example (11), (26), (3), (19)). However, there is currently no universal formalism for describing *specific* emergent properties in terms of their component properties (rather than the presence of emergence in general in a system). Our approach aims to address this while still assuming the information theoretic interpretation of emergence. We also work on the assumption that there may be emergent properties that can not be modelled or simulated because they are substrate-dependent and hence not computable (at least in the classical sense) (6), (1); these are not included in the current discussion.

The motivation for our work comes primarily from the growing use of agent-based modelling and simulation in studying complex systems (25). However, the formalism itself is general enough to describe any system in which higher level macro-properties (of which behaviours are a subset) can arise from the lower level properties of system components and their relationships. Many of the issues discussed here therefore apply equally to the engineering of robotic and swarm systems, where system behaviour is dependent on the collective behaviour of the system's components. There already exist several formalisms for specifying emergent behaviours in such systems e.g. temporal logics (27) but these tend to ignore the multi-level, compositional nature of many emergent properties.

In this paper, we first introduce a general approach for describing emergent properties at multiple levels. We then propose a calculus for describing emergent behaviours that allows them to be decomposed into lower level behaviours. The paper assumes familiarity with the elementary aspects of set, category and type theories (good introductions to these topics can be found in (20) and (16)).

1.1 Emergent Properties

As there is a lack of consensus as to what constitutes 'emergence' (see for example (5), (8)), we will avoid an explicit definition and instead give the assumed attributes that emergent properties are deemed to have in terms of a multi-component system.

- Emergent properties are 'macro'properties with respect to their constituents[4], i.e. they have either:
  - a greater **scope** than its constituent properties i.e. for the property to be present, it has to include more than one of the constituent properties; or
  - a lower **resolution** i.e. the property makes fewer distinctions between its constituent properties than can possibly be made. (within the information theoretic framework, we say that it has lower statistical complexity (2)); or
  - both.

[4] The converse does not hold however; not all 'macro' properties are necessarily emergent.

A more detailed account of these terms is given in Section 4.1 and formal definitions can be found in (23).

- An emergent property arises as a result, not only of the sum of its constituents (as formalised in (9)), but also as a result of their interactions or relationships with one another.
- In a multi-component system, emergent properties are those that are not explicitly defined in the component specifications. This is based on the design/observation distinction introduced in (21)

We will elaborate on these assumptions when we introduce the formalism.

1.2 Locating and composing properties

We define a property as anything that can be measured in a system. There are four central ideas behind our proposed calculus, all relating to way that properties can be located in a system or simulation.

1. Every property in a system consists of one or more (micro-)properties that can be located in an n-dimensional (hyper)space. For the lifetime of the system or simulation, properties can be located in this space by specifying the coordinates in each of the dimensions. The coordinate system used to specify the location can be global (from a whole system point of view) or local (where locations are in relation to a particular constituent *within* the system). For example, if the global coordinate (12, 1, 4, 2) represents the location of a state transition in the 12th time step (first tuple item holds time), located in coordinate (1, 4) of physical space (second and third tuple items hold space) in component with ID 2 (final tuple item holds component identity); the equivalent coordinate using a local coordinate system defined with respect to component 2 at time step 11 in the same spatial location would be (1, 0, 0, 0).
2. If two macro-properties consist of constituents of the same types and constituents of the same type have the same configuration with respect to each other in the two properties, we can say the two properties are of the same type.
3. We can describe **regions** as well as point locations in a system or subsystem space using propositional statements about the location in the system/simulation's various dimensions. For example, in a system with only time and identity represented, (before 3, 4) stands for all the states or state transitions that occur in component 4 before time step 3.
4. Higher level properties can be composed by defining organisational relationships between their constituents i.e. their **configuration**. This idea is generalisable to any dimension.

If a property is defined atemporally[5], we call that property a **state** while if the property has temporal extension, we call it a **behaviour**. In this paper, we focus mainly on the latter of these.

1.3 Outline of the paper

The remainder of the paper is organised as follows:

- Section 2 introduces the main constructs that are used in our calculus of complex events and illustrates them by using them to construct a formalism for macro-states. Our main assumptions about multi-component systems is also given.
- Section 3 introduces the calculus and describes how complex event types can be used to specify behaviour at any level of abstraction that can be realised by the system or simulation.
- Section 4 revisits the criteria for emergence we outlined in Section 1.1 and relates them explicitly to complex events. We also briefly discuss emergent laws and top-down 'causation' in terms of our formalism.
- Section 5 suggests some promising directions for further work.
- Section 6 summarises and concludes the paper.

2 Assumptions and Fundamental constructs

In this section, we begin by stating our assumptions about the way state transitions occur in the multi-component systems we are concerned with. We then introduce two key concepts: configuration and compositionality and illustrate how they can be applied to subsystems, components and states. These two concepts are also fundamental to the temporally extended version of our calculus for temporally extended properties. Key terms are also more precisely defined.

2.1 State transition rules and state transition functions in multi-component systems

We assume that a multi-component system is composed of a set of components, each of which instantiate a defined component type[6]. A component type specifies the component's behaviour with a set of state transition rules (many formalisms exist for specifying component behaviour e.g. state charts (12),

[5] An atemporal definition is one that does not *internally* refer to time e.g. being blue but the property itself can persist through time, in which case it is defined as part of a behaviour e.g. stay blue.

[6] Unencapsulated state variables representing continuous media, spatially dependent quantities etc. are also components under our definition.

Petri nets (18), communicating X-machines (13) but these ultimately reduce to a set of rules). State transition rules are defined relative to a source component and are made up of a *condition* or *trigger* and a set of state transition functions. The *condition* typically takes into account both the component's own state q_C and the state of its environment $q_{Env/C}$, where the environment of a component *for a given rule* is a set of n components $Env/C = C_1, ..., C_n$ in some configuration λ. C and Env/C together make up a subsystem Sub/C (see Definitions 5, 6 and 7). The *condition* is then the set of Sub/C states $sub_1, ...sub_m$ under which the state transition functions of the rule are executed (if the rule is non-deterministic, there is a further guard that determines whether or not the functions are executed). A state transition function is a mapping from a source state q_{source} to a target state q_{target}, where q_{source} and q_{target} are both subsystem states defined relative to the component.

Definition 1. *State transition. A state transition is a transformation of one subsystem state q to another subsystem state q'. The state q before the transformation is applied is called the source state and is denoted q_{source}, while the state q' after the transformation has been applied is called the target state and denoted q_{target}. (The definition for subsystem state is given in Definition 7).*

Definition 2. *State transition function. A state transition function is a function that maps the source subsystem state q_{source} to the target subsystem state q_{target}.*

$$STF \ :: \ q_{source} \rightarrow q_{target} \tag{1}$$

Definition 3. *State transition rule. A state transition rule is a rule that restricts the applicability a set of state transition functions to cases where the propositional formula representing the realisation of condition evaluates to true.*

$$STR \ :: \ (\{STF\}, condition) \tag{2}$$

2.2 Describing the organisation of system constituents using configurations and constraints

Definition 4. *Configuration. In general, a configuration ω_x of a set of n properties $x_1, ..., x_n$ is a set of constraints Θ_x that together describe the organisation of the properties in an m-dimensional (hyper)space. At the lowest level, each constraint is itself a configuration of a pair of properties $\Theta = \theta_{x1-x2} \times ... \times \theta_{x1-xn}$, where θ_{a-b} locates the property b relative to a.*

If the constraints in a configuration result in every constituent being point located in the m-dimensional (hyper)space, we say the configuration is *fully determined.* If any constituent location is defined as a region (after taking into account all the constraints) e.g. given a range or left undefined in some dimension, the configuration is said to be *partially determined.* Two configurations ω_x and ω_y are said to be *identical* when their constraint sets have exactly the same members; this is denoted by the identity operator $\equiv$. They are said to be *equivalent* when they apply to the same number of constituents and the application of ω_x's member constraints is able to give every organisation [7] that satisfies every one of ω_y's constraints and vice versa; equivalence is denoted by the operator $\simeq$. It is also worth noting that a partially determined configuration defines a set of fully determined configurations that satisfy all its constraints. This also means that a partially determined configuration ω_{PartA} can be a subset of another partially determined configuration ω_{PartB} i.e. every fully determined configuration in ω_{PartA} has an equivalent fully determined configuration in ω_{PartB}.

$$\omega_{PartA} \subseteq \omega_{PartB} \tag{3}$$

$$\omega_{PartA} = \{\omega_A 1, ..., \omega_A n\}, \omega_{PartB} = \{\omega_B 1, ...\omega_B m\}, \;\; m \leq n \tag{4}$$

$$\forall \omega_A \exists \omega_B (\omega_A \simeq \omega_B), \;\; \omega_A \in \omega_{PartA}, \;\; \omega_B \in \omega_{PartB} \tag{5}$$

From Definition 4, we can define what it means for a set of components to be in a configuration at a given point in time. This is then used to define subsystem states in Definition 7.

Definition 5. *Component configuration. A component configuration ω_C at time point t_i of a set of components $C_0, C_1, ..., C_n$ is the set of constraints p_i that together describe the organisation of the components at t_i, $\omega t_i = (\theta_{0-1} t_i \times \theta_1 t_i ... \times \theta_n t_i)$.*

Two (sub)systems *sysA* and *sysB* have equivalent component configurations if *sysA* and *sysB* have the same number of components and for every component $C1_{sysA}$ in *sysA*, there exists exactly one corresponding component $C1_{sysB}$ in *sysB* which satisfies the same location constraint to a second component $C2_{sysB}$ as $C1_{sysA}$ satisfies to $C2_{sysA}$, and vice versa.

Definition 6. *Subsystem. A subsystem Sub of a multi-component system is a subset of the system's components. If a system consists of a set of n components $Sys = C_1, ..., C_n$, $Sub \subseteq Sys$ (a system is therefore its own subsystem).*

Definition 7. *Subsystem state. A subsystem state of a multi-component system Sys at a given time point t_i is the composition of the states of any subset of a system's components and a configuration ω. $q_{Sub} t_i = q_{SubComponents} t_i \times \omega_{Sub} t_i$.*

[7] in the case of fully determined configurations, there will only be one possible organisation

2.3 Compositionality of configurations, subsystems, components and states

The constructs and terms defined above imply compositionality in that they can be described by the recursive syntax:

$$A \;::\; a \mid A_1 \bullet A_2, \tag{6}$$

where A is the construct, a is a primitive and $\bullet$ is the compositional operator.

In the case of configurations for example, a configuration can be defined as either a single location or a two configurations satisfying a set of constraints with respect to one another:

$$\omega \;::\; \nu \mid \omega_1 \bowtie \omega_2, \tag{7}$$

where ν is a location and the $\bowtie$ means that the right hand side configuration satisfies a set of location constraints with respect to the left hand side configuration.

Similarly, a subsytem is either a component or a set of two unordered subsystems:

$$Sub \;::\; C \mid (Sub_1, Sub_2) \tag{8}$$

So a system can be decomposed into a hierarchy of subsystems, with the largest subsystem being the system itself and the smallest subsystem being a component. The greater the number of components in a subsystem, the greater the scope and the higher the level.

Example 1. A system with three components (C_1, C_2, C_3) can be decomposed into the following subsystems:

1. (C_1, C_2, C_3)
2. (C_1, C_2), (C_1, C_3), (C_2, C_3)
3. C_1, C_2, C_3

Again using the compositional syntax, a subsystem state is either a component state or two subsystem states satisfying the constraints of some configuration:

$$q_{Sub} \;::\; q_C \mid q_{Sub1} \bowtie q_{Sub2} \tag{9}$$

So we can decompose a given subsystem state into lower level subsystem states, retaining the (sub)configurations of the components and the component states.

Example 2. A subsystem state exists in 2-dimensional space, where component C_1 is located at (0,1) relative to C_2 and C_3 is located at (3, 4) relative to C_2, and C_1, C_2 and C_3 are in states q_0, q_2 and q_1 respectively. We can denote this: $[C_1 \leftarrow q_0] \triangleright (0,1)[C_2 \leftarrow q_2] \triangleright (3,4)[C_3 \leftarrow q_1]$.

This can be decomposed into the following subsystem states:

1. $[C_1 \leftarrow q_0] \triangleright (0,1)[C_2 \leftarrow q_2] \triangleright (3,4)[C_3 \leftarrow q_1]$
2.
   - $[C_1 \leftarrow q_0] \triangleright (0,1)[C_2 \leftarrow q_2]$,
   - $[C_1 \leftarrow q_0] \triangleright (3,5)[C_3 \leftarrow q_1]$ (from the location of C_3 relative to C_2),
   - $[C_2 \leftarrow q_2] \triangleright (3,4)[C_3 \leftarrow q_1]$
3. q_0, q_2, q_1.

This compositional approach can be generalised to any property. For example, a component C consists of a set of encapsulated state variables whose values at t_i together make up the component's state q_C at t_i, $q_C t_i = var_0 t_i \times var_1 t_i, ... \times var_n t_i$. We can specify subsets and configurations for these variables within and between components and associate them with macro-properties. In fact, the component's state is itself a macro-property of its constituent variable values. Figure 1 illustrates this general compositional approach.

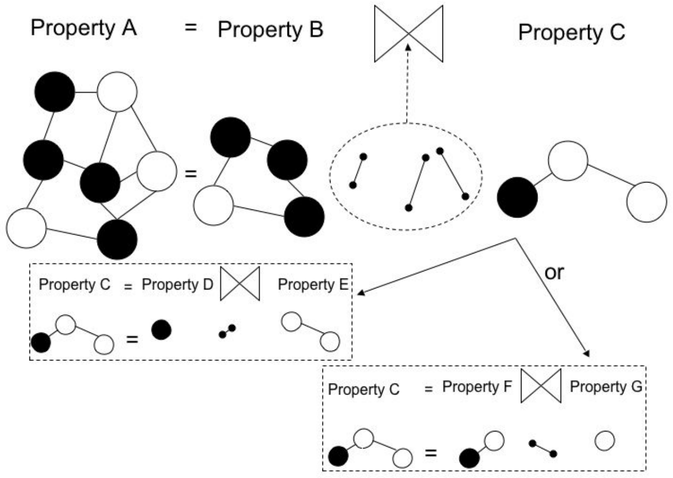

Fig. 1. Property A is a fully determined property (i.e. each of its lowest level constituent properties occupies a point location in the (hyper)space defined by a set of dimensions) composed of Property B and Property C in some configuration. B and C can be further decomposed, as can their constituents. $\bowtie$ denotes the set of constraints that determine the configuration; x, y and z are used to distinguish between two different sets of constraints.

3 Complex events

In this section, we introduce the complex event calculus and describe how complex event types can be used practically to specify and then detect macro-behaviours in systems. We then relate complex event type compositionality to scope and specificity to resolution. We take as a starting assumption the fact that behaviours are changes in state i.e. events (see Definition 8).

Definition 8. *Event. An event is a state transition defined at a particular level of abstraction.*

The compositionality of state means that state transition functions can also be decomposed into lower level state transition functions. For example, a state transition function that maps a source subsystem state q_{Sub} to another subsystem state q'_{Sub} (the target state) can be decomposed into a set of lower level subsystem state mappings, which can be further decomposed into a set of component state mappings. If variables are the lowest level of state representation, every state transition function can ultimately be reduced to a set of state transitions mapping a variable value to a new variable value.

$$q_{Sub} \rightarrow q'_{Sub} \equiv \{q_{Sub} \rightarrow q'_{Sub}\} \equiv \{q_C \rightarrow q'_C\} \equiv \{var \rightarrow var'\} \tag{10}$$

In a multi-component system, every event is the result of a state transition function being applied; we call these simple events (See Definition 9) and their level of abstraction depends on their state transition function.

Definition 9. *Simple event. A simple event is a state transition that results from the application of a state transition function.*

Two simple events that occur in a system are said to be of the same type if they result from the same state transition functions (state mappings) and the mappings are applied as a consequence of the same rule being executed.

Definition 10. *Simple event type. Two simple events are said to be of the same type if (i) they are the result of the same state transition function and (ii) the function is applied as a result of the same state transition rule.*

Figure 2 shows the relationship between component type and components and between state transition functions and simple events in the executing system.

Given these preliminaries, we can define a complex event CE as either a simple event SE or two complex events linked by $\bowtie$:

$$CE \;\; :: \;\; SE \mid CE_1 \bowtie CE_2 \tag{11}$$

$\bowtie$ was introduced in Section 2.3. Here, it denotes the fact that CE_2 satisfies a set of location constraints with respect to CE_1. Conceptually, complex events are a configuration of simple events where the configuration can be defined in a hyperspace that includes time, physical space and any other dimensions.

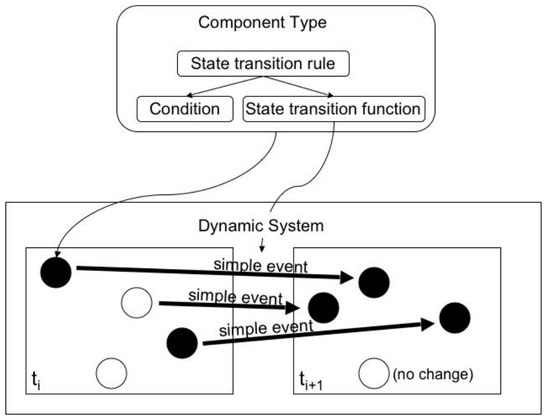

Fig. 2. Components instantiate component types in the real system or simulation while simple events can be said to instantiate state transition functions.

Definition 11. *Complex event (recursively defined). A complex event is either a simple event SE or two complex events CE_1 and CE_2 satisfying a set of location constraints with respect to each other:*

$$CE_1 \bowtie_x CE_2,\ \ CE_2 \bowtie_{x^{-1}} CE_1 \tag{12}$$

, where x^{-1} is the inverse of x so that $CE_1 \bowtie_x CE_2$ denotes satisfaction of the set of location constraints x by CE_2 with respect to CE_1 and $CE_2 \bowtie_{x-1} CE_2$ denotes the satisfaction of the set of location constraints x^{-1} by CE_1 with respect to CE_2.

Location constraints define relationships between complex events, which can be represented as a coloured multi-graph, where the coloured nodes stand for event types and coloured edges stand for the different relationship types (sets of location constraints) existing between events (4). It should also be pointed out that because location constraints can be defined at different levels of specificity (see also Section 3.2), different multigraphs can be drawn for the same complex event. These alternative multigraphs can be seen to represent different 'views' or descriptions of the same behaviour. Similarly, we can infer 'hidden' edges in a multi-graph when a relationship between one pair of events implies a relationship between another pair of events e.g $e1 < e2$, $e2 < e3$ implies $e1 < e3$.

3.1 Specifying complex event types

We have already introduced the idea that events can be typed in our discussion of simple events (see Definition 10). We now extend this to complex events.

Definition 12. *Complex event type. Two complex events CE_1 and CE_2 are said to be of the same type if, for each constituent event $e1$ in CE_1 there is exactly one event $e2$ in CE_2 satisfying the same location constraints, and $e1$ and $e2$ are events of the same type.*

To specify a complex event type therefore, we need to specify the types for each of the constituent events and the location constraints that hold between them.

It is possible to use formal expressions from different formal systems such as temporal logics to specify the location constraints. This itself is a worthy subject for research since different applications are likely to have different semantic and expressivity requirements. However, a comprehensive review and analysis of the these different formal systems remains outside the scope of the paper (good reviews can be found in (17) and (22)); here we only present a few illustrative examples.

Example 3. Temporal constraints: $\|$ concurrently, ; immediately after, $\prec$[[$\leq$, $<$, $\geq$, $>$]] t] [[less than or equal to, less than, greater than or equal to, greater than] t time units] after .

E.g.

- $E_1 \prec [< 4] E_2$ means that E_2 occurs within 4 time units of E_1.
- $E_1 (\prec [< 4]) \vee (\prec [> 8]) E_2$ means that E_2 occurs either within 4 time units of E_1 or it occurs more than 8 time units after E_1.

Example 4. Spatial constraints: within distance v $\circ(v)$, at location $(x, [y, ..., z])$ $triangleright(x, [y, ..., z])$.

E.g.

- $E_1 \triangleright (-3, 2) E_2$ means that every component in which a state changes occurs as a consequence of E_2 is $(-3, 2)$ with respect to the corresponding component in E_1.
- $E_1 \prec [< 4] \wedge \circ(3) E_2$ means that E_2 occurs within 4 time units and within a distance of 3 units from E_1.

Example 5. Token identical component $[C_A/C_B]$.

E.g.

- $E_1 [C_A/C_B] E_2$ means that E_2 occurs in the same component (with the same ID) as the component in which E_1 occurs.

3.2 Specificity of complex event types

Just as partially determined configurations define sets of fully determined configurations (see Section 2.2), complex event types can differ in specificity with a fully determined complex event type CET_{Full} being defined as one whose constituent events are in a fully determined configuration. A partially determined complex event type CET_{Part} is then one with a partially determined configuration and therefore defines a set of complex events with fully determined configurations.

$$CET_{Part} = \{CET_{Full}\} \tag{13}$$

The dimensions in which configurations are not fully specified lower the resolution of the complex event, with weaker constraints (greater ranges of possible values) implying a lower resolution in that dimension (this id defined more precisely in Section 4.1). More generally, the greater the number of complex event types with fully determined configurations that a complex event type contains, the lower its resolution (see Figure 3).

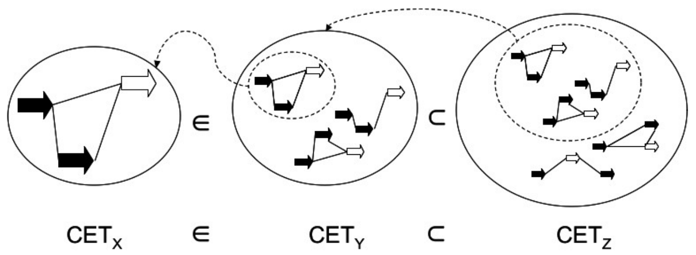

Fig. 3. CET_X is a fully determined complex event type which is an element in CET_Y (a partially determined complex event type). CET_Y is a subset of CET_Z, which is the least specific (lowest resolution) of the three complex event types.

4 When do complex event types represent emergent behaviours?

In Section 1.1, we listed three main criteria that emergent properties should fulfil, namely, (i) they should be macro-properties with respect to their constituents, (ii) they should not be reducible to the sum of their constituents and (iii) they have not been explicitly specified. In this section, we return to these criteria and explicitly link them with categories of complex event types, which can then be said to represent emergent behaviours. We also show how emergent 'laws' and top-down 'causation' can be formalised.

4.1 Scope and Resolution of complex event types

The criteria for a property being a macro-property with respect to another are that it has a greater scope, a lower resolution, or both. Complex event types represent behaviours, so the scope and resolution of a complex event type is the scope and resolution of the behaviour represented by that complex event type.

Definition 13. *A behaviour M is a macro-behaviour with respect to another behaviour m if the following is true of the complex event type CET_M representing behaviour M and the complex event type CET_m representing behaviour m:*

$$Scope_{CET_M} \geq Scope_{CET_m} \tag{14}$$

where a complex event type CET_A has a greater scope than another complex event type CET_2 if the minimum number of simple events at the lowest possible level required for CET_1 is greater than that required for than in CET_2.

$$Resolution_{CET_M} \leq Resolution_{CET_m} \tag{15}$$

where a complex event type CET_1 has a lower resolution than another complex event type CET_2 if the set of CET_{full} defined by CET_1 contains more elements than CET_2.

$$(Scope_{CET_M}, Resolution_{CET_M}) \neq (Scope_{CET_m}, Resolution_{CET_m}) \tag{16}$$

4.2 Irreducibility of emergent behaviours

The second criterion required for us to say that a particular property is emergent with respect to another is that the first can not be reduced simply to the sum of the second. Applying this to behaviour, a behaviour M is emergent with respect to another behaviour m if M can not be reduced simply to the sum of ms. In complex event type terms, we can say that a complex event type CET_M is reducible to the sum of CET_ms if:

$$CET_M = CET_m[\bowtie CET_m]* = CET_m^+ \tag{17}$$

where $*$ denotes 0 or more occurrences and $+$ denotes 1 or more occurrences. Hence, we can say that the emergent behaviour M is emergent with respect to m when their complex event types CET_M and CET_m satisfy:

$$CET_M = CET_m[\bowtie CET_m]* \neq CET_m^+ \tag{18}$$

This is consistent with the grammar-based definitions given in (7), (9) and (15) e.g. Demazeau's pseudo-equation:

$$MAS = A + E + I + O = emergence$$

, which corresponds to:

$$L(MAS) \supset \sum_{v \in vowels} (L(v))$$

, where $vowels$ is a micro-partition of a macro MAS (8).

The final criterion for a behaviour to be classified as emergent is that it is not explicitly specified in the component rules. This is the case when a complex event type is not a simple event type.

4.3 Emergent 'laws', top-down 'causation' and multi-functionality in terms of complex events types

A popular application of the multi-component paradigm is in multi-agent simulations of complex systems to understand their properties. Scientists working in this area are particularly interested in establishing rules at higher levels, which they often call emergent laws. Top-down constraints (also known as top-down 'causation'[8]) and feedback are also effects that are sought in simulations. While simple statistical techniques can be used to detect these effects at the global level, a hierarchical approach has yet to be established. Definitions 14 and 15 support such an approach by stating in terms of complex event types the criteria for behaviours at any two levels to be related in these two ways.

Definition 14. *An emergent law exists between two complex event types CET_x and CET_y when a complex event type CET_x exists such that the occurrence of CET_x implies the occurrence of some complex event of the type CET_y i.e. $CET_x \rightarrow CET_y$, and CET_x is not a simple event.*

Definition 15. *A top-down constraint effect exists between two complex event types CET_M and CET_m when an emergent law $CET_M \rightarrow CET_m$ holds and CET_m is a lower level complex event type with respect to CET_M.*

Finally, we can use the complex event type formalism to express multi-functionality. A complex event type can be decomposed into different combinations of constituent events (see also the decomposition of property C in Figure 1), e.g.:

$$CET_A = CET_{x1} \bowtie_1 CET_2 = CET_{x1} \bowtie_2 CET_3 \tag{19}$$

where $\bowtie_1$ and $\bowtie_2$ are different sets of configuration constraints.

The overlapping of constituent properties gives us a criterion by which to establish multi-functionality.

Definition 16. *A complex event type CET_x is multi-functional if it is a constituent event in more than one complex event type.*

Example 6. If

$$CET_A = CET_x \bowtie CET_y \tag{20}$$

and

$$CET_B = CET_x \bowtie CET_z \tag{21}$$

we say that CET_x is multi-functional and that it has a role in CET_A and CET_B

[8] Here, we make no assumptions about the metaphysical status of top-down constraints i.e. we take an agnostic stance on whether these constraints have real causal power, supervene on lower level laws, or are epiphenomena. This has been a long-standing debate in the Philosophy of Science, see, for example (10), (24), (14)

5 Directions for further investigation

In this section, we propose several directions for extending the formalism and enhancing its practical application.

- We have assumed that every fully defined complex event type in a partially defined complex event type set representing a behaviour is an equally good exemplar of that behaviour. However, an extended formulation in terms of fuzzy set membership functions would be worth investigating for some behaviours that can not be crisply defined. The same holds for the specification of subsystem states any other emergent properties.
- The specification of configuration constraints for complex event types and subsystem states should be further formalised. A sound approach would be to use the same formalism as used to specify the system (statecharts, petri nets etc.) to specify states and execution trajectories.
- In this paper, we have used the dimensions of physical space, time and component identity to 'locate' events and states. However, our approach could be generalised to other dimensions and to other properties so that other types of inter-level relationships can be identified.
- Quantitative techniques are required to precisely express and evaluate the participation of lower level properties in higher level ones since they often overlap.
- The implementation of complex event detection can be extremely costly from a computational point but the compositional nature of complex events could provide a promising means of optimising the search.

6 Summary and Conclusion

In this paper we have introduced a compositional approach to formalising and analysing emergent properties. In general, emergent properties can be described by constituent properties that are organisationally related in a set of dimensions. Organisational constraints can result in fully specified properties, where the configuration of the constituent properties is fully determined, or they can be partially specified so that the property is represented by a set. More specifically, we have proposed a practical formalism that can be used to categorise different system sub-trajectories so they can usefully be used to understand the mechanisms at work at multiple system levels. This allows us to analyse an emergent behaviour at any level and establish relationships between behaviours at multiple levels. Furthermore, by reducing behaviours at higher levels to smaller and smaller 'motifs' of behaviour, with simple events (and hence state transition rules) at the lowest level, we have a sound method for engineering and modifying multi-component systems to exhibit or prevent certain emergent behaviours.

References

[1] F. Boschetti and R. Gray. Emergence and computability. *Emergence: Complexity and Organisation*, pages 120–130, 2007.

[2] F. Boschetti and R. Gray. *A Turing test for Emergence*. Springer-Verlag, London, UK, 2007.

[3] F. Boschetti, M. Prokopenko, M. Macraedie, and A. Grisogono. Defining and detecting emergence in complex networks. In R. Khosla, R. J. Howlett, and L. C. Jain, editors, *Knowledge-based intelligent information and engineering systems, 9th International Conference, KES*, volume 3684 of *LNCS*, pages 573–580. Springer, September 2005.

[4] C-C. Chen, S. B. Nagl, and C. D. Clack. Specifying, detecting and analysing emergent behaviours in multi-level agent-based simulations. In *Proceedings of the Summer Simulation Conference, Agent-directed simulation*. SCS, 2007.

[5] M. Christen and L. R. Franklin. The concept of emergence in complexity science: Finding coherence between theory and practice. In *Proceedings of the Complex Systems Summer School*, 2002.

[6] B. Cooper and P. Odifreddi. *Incomputability in nature*, pages 137–160. Computability and models. Kluwer Academic, Dordrecht, 2003.

[7] J. Dassow, R. Freund, and G. Paun. Cooperating array grammar systems. *International Journal of Pattern Recognition Artificial Intelligence*, 9:1029–1053, 1995.

[8] J. Deguet, Y. Demazeau, and L. Magnin. Elements about the emergence issue: A survey of emergence definitions. *ComPlexUs*, 3:24–31, August 2006.

[9] Y. Demazeau. Steps towards multi-agent oriented programming. In *First International Workshop on Multi Agent Systems*, Boston, Mass., 1997.

[10] M. Friedman. Explanation and scientific undersatnding. *Journal of Philosophy*, 71:5–19, 1974.

[11] P. Grassberger. Toward a quantitative theory of self-generated complexity. *International Journal of Theoretical Physics*, 25:907–938, 1986.

[12] D. Harel. Statecharts: A visual formalism for complex systems. *SCP*, 8:231–274, 1987.

[13] P. Kefalas, M. Halcombe, G. Eleftherakis, and M. Gheorghe. A formal method for the development of agent-based systems. In V. Plekhanova, editor, *Intelligent Agent Software Engineering*, UK, 2003. Idea Group Publishing.

[14] J. Kim. *In emergence or reduction?*, volume 1, chapter Downward causation, pages 119–138. Walter de Gruyter & Co., 1992.

[15] A. Kubik. Toward a formalization of emergence. *Artificial Life*, 9:41–66, 2003.

[16] F. W. Lawvere and S. H. Schamiel. *Conceptual mathematics: a first introduction to categories*. Cambridge University Press, 1997.

[17] F. Moller and G. Birtwistle. *Logics for concurrency: Structure versus automata.* Lecture Notes in Computer Science. Springer, 1996.

[18] C. A. Petri. *Kommunikation mit Automaten.* PhD thesis, Institut fuer Instrumentelle Mathematik, Bonn, 1962.

[19] D. Polani. Emergence, intrinsic structure of informatio, and agenthood. In *International Conference on Complex Systems (ICCS)*, 2006.

[20] M. D. Potter. *Sets: An introduction.* Oxford Science Publications, Clarendon Press, Oxford, 1990.

[21] E. Ronald and M. Sipper. Design, observation, surprise! a test of emergence. *Artifcial Life*, 5:225–239, 1999.

[22] C. A. Rouff, M. G. Hinchey, J. Rash, W. Truszkowski, and D. Gordon-Spears, editors. *Agent technology from a formal perspective.* Springer-Verlag London, 2005.

[23] A. Ryan. Emergence is coupled to scope, not level. *Nonlinear Sciences*, 2007.

[24] W. Salmon. *Scientific explanation and the causal structure of the world.* Princeton University Press, 1984.

[25] C. R. Shalizi. *Methods and Techniques of Complex Systems Science: An Overview*, chapter Methods and Techniques of Complex Systems Science: An Overview, pages 33–114. Springer, New York, 2006.

[26] C. R. Shalizi, K. L. Shalizi, and R. Haslinger. Quantifying self-organisation with optimal predictors. *Physical Review Letters*, 93(118701), 2004.

[27] A. F. T. Winfield, J. Sa, M-C Fernandez Gago, C. Dixon, and M. Fisher. On formal specification of emergent behaviours in swarm robotic systems. *International Journal of Advanced Robotic Systems*, 2(4):363–370, December 2005.