A phenomenal basis for hybrid modelling

Jon G. Hall Lucia Rapanotti
School of Computing and Communications
The Open University
Walton Hall
Milton Keynes, UK
Jon.Hall, Lucia.Rapanotti@open.ac.uk

Georgi Markov
Siemens Corporation
CT RDA SSI AVI-US
755 College Road East
Princeton, NJ 08540-6632, USA
Georgi.Markov@siemens.com

***Abstract*—This work in progress extends the *new mechanical philosophy* from science to engineering. Engineering is the practice of organising the design and construction of artifices that satisfy needs in real-world contexts. This work shows how artifices can be described in terms of their mechanisms and composed through their observable phenomena.**

Typically, the engineering of real system requires descriptions in many different languages: software components will be described in code; sensors and actuators in terms of their physical and electronic characteristics; plant in terms of differential equations, perhaps. Another aspect of this work, then, to construct a formal framework so that diverse description languages can be used to characterise sub-mechanisms.

The work is situated in Problem Oriented Engineering, a design theoretic framework engineering defined by the first two authors.

***Keywords*-mechanism description; parallel composition; problem orientation; engineering**

I. Introduction

The turn of the century has seen the *new mechanical philosophy* "[emerge] as a framework for thinking about the philosophical assumptions underlying many areas of science, especially biology, neuroscience, and psychology" [3, 4]. Under this view, the aim of science is to synthesise mechanisms the results of which are observable natural phenomena.

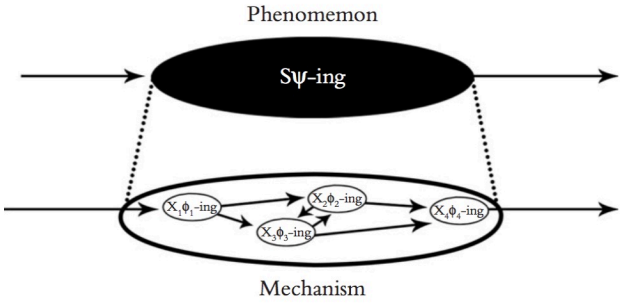

Figure 1. Craver's canonical mechanism [4]; linear time at the systems level, potentially other forms of time, branching, looping, *etc* at the mechanism level

Craver's [4] view of mechanism is shown in Figure 1. There we see two levels of behaviour. On the lower level, are many parts (the X_i) whose behaviour (the ϕ_i) is organised and influences each other to create, on the upper level, the observable behaviour (the ψ) of the system (S).

Craver's model is located in biological system science — Figure 1 is motivated by a description of neurons releasing neurotransmitters — and is created to model natural phenomena. Our interest in the new mechanical philosophy is different in that *we wish to be able to design and implement mechanisms for which the phenomena generated interact to serve a specific need in context*; i.e., we wish to design solutions to problems.

To do so, we must be able to describe mechanisms that create desired phenomena and understand their conglomeration and interaction as their phenomena combine them into larger mechanisms. Challenges exist both at the behavioural levels and in their interaction:

- At the lower level, we must be willing to consider conglomerate mechanisms whose individual descriptions necessarily share no single language: a socio-technical system — a call centre, for instance — may include software to control a Customer Relationship Management (CRM) system, operator scripts by which prospects can be approached, policies for data protection and processes by which complaints are handled. These artefacts share little in terms of 'interpreting platform': a microprocessor interpreter of (compiled) software code won't be able to interpret an operator script; a human operator wouldn't be the first choice to interpret a python script. The disparity of description reflects these differences.
  More generally, target systems might be described as combinations of flowcharts, process calculi, software code; in DNA, RNA, or as proteins; scripts, policy, process and procedure; recipes, etc.
- Even so, the constituent parts of the system must talk together to achieve their objectives. They do so through the phenomena generated by the mechanism, the occurrences of which are observable and shared: so, for instance, software may choose and dial the next prospect, indicating through a light on the screen (an observable occurrence) that the phone has been

answered. Likewise, the human operator will encode a prospect's responses into the CRM, another observable occurrence.

In terms of Craver's model, then, at the upper level the ways in which phenomena interact may constrain how their underlying mechanisms develop over time. For instance, when we interact with our smart phone the finger-touch phenomenon interacts with the screen or buttons and whilst this is in progress, the smart phone's operation is restricted to some sub-mechanism.

- We can expect the language in which the mechanism is described to be chosen so that it is suitable for that mechanism, and so representative of its characteristics, such as the ways in which the generated phenomena synchronise. In this case, the sharing of phenomena between mechanisms is subject to, what we might call, 'impedance matching' in that expectations exist for the synchronisation on both sides. We present an example of poor impedance matching later.

Additionally, the conglomerate mechanism will share a context with other mechanisms and within which it interacts via the sharing of phenomena. This shared context will typically not have a description in terms of mechanism — it may, for instance, be a country's economy for which is being designed a fiscal policy — and so we may simply know more or less about its mechanisms and phenomena.

According to [13, 6, 5], a phenomenon is the behaviour of the mechanism as a whole. We agree with this interpretation, but also see value in being able to extract, what might be called, sub-phenomena corresponding to constituent (sub-)mechanisms of a conglomerate. This will allow us to work at finer granularities of phenomena, for instance, those shared between identified sub-systems.

In this paper, we begin a characterisation of diverse systems composition as needed in the construction of mechanism that solves problems. In particular, we introduce the notion of a *mechanism description language* (*mdl*), a language for describing mechanisms. This relates back to Craver's model, although we do not limit ourselves to biological mechanisms: we wish to consider designed formal mechanisms as might, for instance, be expressed through HCSP [1], other dialects of Process Algebras [10, 15], Petri Nets [12], Statecharts [9], programming languages, *etc*.

The work is motivated by a need for a formal interpretation of the sharing mechanism used by problem oriented approaches, including Problem Oriented Engineering (or POE, [7]) and Problem Frames ([11]). As a theory of design, POE takes a balanced view of problem and solution; makes no constraints on the languages in which problems or solutions are expressed; supports formal, informal and even intuitive reasoning; and gives focus to the construction of a stakeholder-relevant design rationale which captures their notion of satisfaction. A POE problem is a need in real-world context [8]:

- the need expresses wished-for phenomena and relations between them;
- the context describes existing phenomena and their relations;

Problem solving provides an artifice which is a mechanism (or conglomerate) created so that new phenomena are created and existing phenomena and their relations are manipulated to produce those wished for. Thus, we do not expect each sub-mechanism of a system to be described in the same language, so there may be many different mdls composed in parallel in the description of a single system. This complicates the semantics of parallel composition as we cannot expect its operands to share anything other than the phenomena they use to communicate. In particular, this distances the semantics of parallel composition somewhat from conjunction.

II. Phenomena and Phenomenal Models

Definition 1 (Phenomenon): A *phenomenon* an element of the world the occurrences of which are observable.

NB: we do not say that a phenomenon should be observed for it to be a phenomenon, only that it can be.

Some phenomena arise naturally, others artificially: a protein occurs through an evolved protein synthesis mechanism whereas, from a software perspective, the mechanisms for software handshake occurrences are constructed by software engineers. The meanings of natural and artificial here relate not to the phenomena themselves but to the mechanisms that underlie their occurrence: it may be that a solution to a problem has, as a component, an artificially produced protein.

That phenomena are observable reveals their dependence on time: in being observable, a phenomenon must exist at the same time as the observer. This dependence can be simple, as might be the temperature, or more complex, as might be time-taking-creation in the case of a protein or the exercise of a software handshake. To this end, being able to talk both about point time and interval time is appropriate.

In the next section, we define *phenomenal models* as the semantic basis of our presentation. The basis of a phenomenal model is a temporal frame, specifically an interval temporal frame of which there are many variants, for instance [17, 14].

Definition 2 (Interval temporal frame, cf. [14]):
$\mathcal{T} = \langle T, \sqsubseteq, \prec, \preccurlyeq \rangle$ is an interval temporal frame, where

- T is the collection of time intervals; we define $\bar{T} = \bigcup_{I \in T} I$, the *time base* of $\mathcal{T}$;
- *is-contained-by*, $I \sqsubseteq J$ means J has an earlier starting point than I and a later ending point;
- *precedes*, $I \prec J$ means I is before J and there is a non-empty interval between them;
- *meets*, $I \preccurlyeq J$ means I is before J but there is no non-empty interval between them.

These interval relationships are illustrated in Figure 2. Actually, for our purposes, the important relation is *meets*; see Section II-A

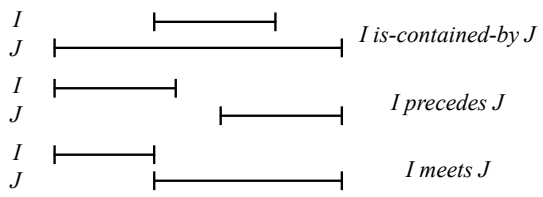

Figure 2. The relationships between intervals I and J in an Interval Temporal Frame

As interval- and instant-based models of time are interelatable, we could have chosen to base our analysis on an instant-based model of time, i.e., with T a collection of points. However, interval-based descriptions are 'cleaner' for our purposes. For completeness, however, we identify the time point t with the point-interval $[t, t]$, when contained in T. Our intention is that the universal time base will be the non-negative reals $\mathbb{R}_{\geq 0}$.

A. Phenomenal models

There are many different types of communication between systems; an incomplete (and overlapping) ontology includes synchronous and asynchronous, blocking and non-blocking, buffered and non-buffered. Here, for simplicity, we focus on message passing via explicit channels of the form defined in process algebras such as CSP ([10]), i.e., both transmitting and receiving processes wait when ready to communicate. This model of communication is carried over to Hybrid CSP (HCSP) [1] through which much of our development is done. In HCSP, a channel *ch*, say, mediates between processes and carries values between them when both are willing to participate; a process's willingness to communicate is indicated through two predicates associated with the channel:

- $ch!$, the *output predicate*, being that of the sending process;
- $ch?$, the *input predicate*, being that of the receiving process.

Channel communications are phenomena; the value communicated are occurrences visible to other processes. We collect all channels together as the set *PHEN*. In addition to phenomena, which are for sharing, we also assume a set of state variables, *STATE*, through wish internal state is captured. State variables are not shared between mechanisms.

Thus:

Definition 3 (Phenomenal Model): Suppose *PHEN* is a set of phenomena, *STATE* a set of state variables, and $PROP = \{p_1, ..., p_n\}$ a set of atomic formulae containing $ch!, ch?$ for each $ch \in PHEN$. Then a *phenomenal model* for *PROP* over *PHEN* is a 3-tuple $M = \langle \mathcal{T}, V, W \rangle$, where

- $\mathcal{T} = \langle T, \sqsubseteq, \prec, \preccurlyeq \rangle$ is an interval temporal frame over time base the non-negative reals, $\mathbb{R}_{\geq 0}$;
- V is a *valuation* assigning to every $p \in PROP$ a set of intervals $V(p) \subseteq T$ over which p is true; $V(p)$ is downwards closed, i.e., $I \in V(p) \wedge J \sqsubseteq I \Rightarrow J \in V(p)$ and is extended to $\bar{T}$ in the obvious way, i.e., for $w \in \bar{T}$, $V(p)(w) = true$ when $\exists I \in V(p) \bullet w \in I$;
- W is an *interpretation* assigning to each $ch \in PHEN$ a type $type(ch)$ and a function $ch : \bar{T} \to type(ch)$, and to each state variable $z \in STATE$ a type $type(z)$ and a function $z : \bar{T} \to type(z)$.

The class of phenomenal models is written *PMODEL*.

For simplicity, we will assume all state and channel variables have type $\mathbb{R}$.

A convenient visualisation of a phenomenal model is as a collection of graphs, one for each phenomenon and atomic proposition. The graph for the phenomenal model over $PHEN = \{ch\}$ and $STATE = \{z\}$, with $I = [c, d] \in V(ch!)$, $I \in V(\neg ch?)$, *ch* unconstrained and continuous z is shown in Figure 3. Values that are constant over the interval are shown with their constant value boxed at the beginning of the interval; unconstrained values (i.e., whose value is unknown) are dotted.

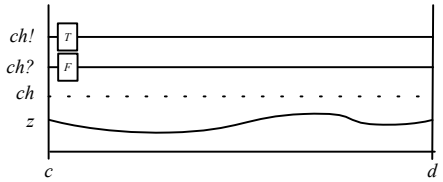

Figure 3. Visualising the phenomenal model over $PHEN = \{ch\}$ and $STATE = \{z\}$ on the interval $I = [c, d]$ and with $I \in V(ch!)$, $I \in V(\neg ch?)$, unconstrained phenomenon *ch* and continuous state variable z.

III. Mechanism Description Languages

A *mechanism description language* (*mdl*) is a language for describing mechanisms. This relates back to Craver's model, although we do not limit ourselves to biological mechanisms.

Definition 4 (Mechanism description language): Given *PROP* and *STATE*, a *mechanism description language* (*mdl*) *MECH* over *PHEN* and *STATE* is a formal system that has an embedded mechanism for synchronous input and/or output on $ch \in PHEN$ and which has a semantic function:

$$[\![ ]\!] : MECH \to 2^{PMODELS}$$

i.e., maps an mdl expression to a set of phenomenal models.

Given our wish to be inclusive, we do not further constrain an mdl's syntax; in fact, we make no assumption that the representation is textual. The reason is that we want to be able to combine disparate mdls, for instance, HCSP

with Petri Nets with process algebras with programming languages. The combination takes place through the valid phenomenal models for the expressions.

There are many ways of producing a set of phenomenal models. One traditional way to do so is to define a logic. This is the route that HCSP takes.

A. HCSP as a mechanism description language

We do not have sufficient space to recount the semantic function of HCSP, but the mapping is defined via the Duration Calculus, *DC* in [2, page 524ff][1].

The set of *PMODELS* for an HCSP expression is then the satisfying models for the *DC* formula. Thus:

$$[\![\,]\!] : HCSP \rightarrow 2^{PMODELS}$$

Example 1: Consider the HCSP terms $\tau_1 \triangleq x := 4; ch!x$ and $\tau_2 \triangleq ch?y; z := y$. These terms have semantics, respectively:

$$\begin{aligned}
[\![\tau_1]\!] \;=\; & \lceil\rceil \vee ((\lceil x = 4\rceil \wedge keep^{\backslash x}) \\
& \frown (wait(ch!) \wedge keep) \\
& \vee ((wait(ch!) \wedge keep) \frown syn(ch!, x)) \\
[\![\tau_2]\!] \;=\; & (wait(ch?) \wedge keep) \\
& \vee ((wait(ch?) \wedge keep) \frown syn(ch?, y) \\
& \frown (\lceil\rceil \vee \exists \beta \in type(y).\beta = \mathbf{p}.y \wedge \lceil y = \beta\rceil \wedge keep^{\backslash y}))
\end{aligned}$$

in which:

- *keep* is true of process whose state variables keep their value fixed throughout an interval ($keep^{\backslash x}$ true if all except x is fixed);
- $wait(\alpha)$ is true of a process if the specified channel flag is false, i.e., the process is waiting for output ($\alpha = ch!$) or input ($\alpha = ch?$);
- $syn(ch?, y)$ is true of a process if the specified channel flag is true, i.e., the process can receive input, and y's value coincides with the value on ch. $syn(ch!, x)$ is true of a process if the specified channel flag is true, i.e., the process can transmit output, and x's value coincides with the value on ch;
- $\mathbf{p}.e$ stands for the value of expression e, when any variable x of e takes its previous value $\mathbf{p}.x$ (cf. [1]).

Satisfying phenomenal models for τ_1 are of two forms, each of which has $dom(V_1) = \{ch!, ch?\}$ and $dom(W_1) = \{ch, x\}$. The satisfying models are illustrated in Figures 4 and 5. Consider an interval $I = [c, d]$.

The first is when there is no synchronisation, i.e., where, for some $e \in (c, d]$ (see the upper graph of Figure 4):

$$\begin{aligned}
V_1(ch!)(w) &= T, \quad w \in [e, d] \\
V_1(ch?)(w) &= F, \quad w \in [e, d] \\
W_1(x)(w) &= 4, \quad w \in [c, d]
\end{aligned}$$

[1]Actually, the semantics is in the continuation style, i.e., $[\![\,]\!] : DC \rightarrow DC$ but we ignore this complication here.

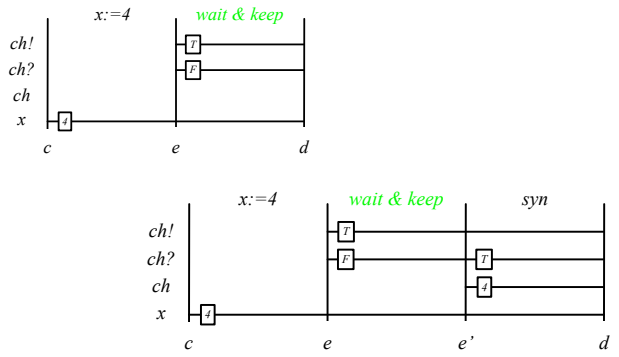

Figure 4. Models for τ_1: (above) when synchronisation not forthcoming and (below) when synchronisation forthcoming, $e \in (c, d]$, $e' \in [e, d)$. Green means interval can be empty.

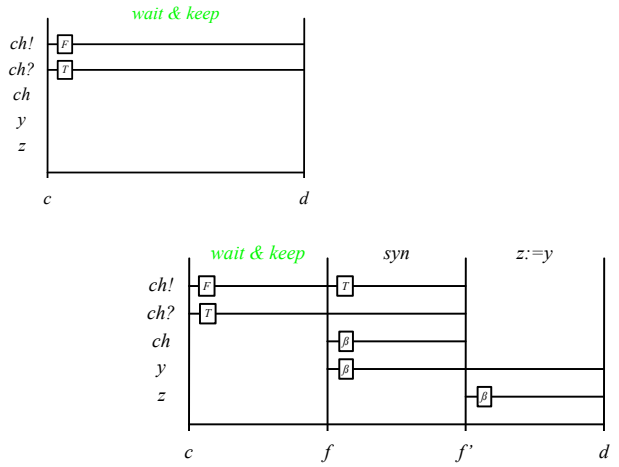

Figure 5. Models for τ_2, each β: (left) when synchronisation not forthcoming and (right) when synchronisation forthcoming, $f \in [c, d)$, $f' \in (f, d)$. Green means interval can be empty.

The second is where synchronisation is completed, i.e., where, for some $e \in (c, d]$, $e' \in [e, d)$ (see the lower graph of Figure 4):

$$\begin{aligned}
V_1(ch!)(w) &= T, \quad w \in [e, d] \\
V_1(ch?)(w) &= \begin{cases} F, & w \in [e, e') \\ T, & w \in [e', d] \end{cases} \\
W_1(ch)(w) &= 4, \quad w \in [e', d] \\
W_1(x)(w) &= 4, \quad w \in [c, d]
\end{aligned}$$

Similarly, those of τ_2 share $dom(V_2) = \{ch!, ch?\}$ and $dom(W_2) = \{ch, y, z\}$. The first is when synchronisation fails (see the upper graph of Figure 5):

$$\begin{aligned}
V_2(ch!)(w) &= F, \quad w \in [c, d] \\
V_2(ch?)(w) &= T, \quad w \in [c, d]
\end{aligned}$$

The second form is when synchronisation succeeds, for some $\beta \in type(x)$, $f \in [c, d)$ and $f' \in (f, d)$ (see the lower graph of Figure 5):

$$
\begin{array}{rcll}
V_2(ch!)(w) & = & \begin{cases} F, & w \in [c, f) \\ T, & w \in [f, f'] \end{cases} & \\
V_2(ch?)(w) & = & T, & w \in [c, f'] \\
W_2(ch)(w) & = & \beta, & w \in [f, f'] \\
W_2(y)(w) & = & \beta, & w \in [f, d] \\
W_2(z)(w) & = & \beta, & w \in [f', d]
\end{array}
$$

B. Coloured Petri Nets as mechanism description language

Coloured Petri Nets (CPNs) [12] are a popular variant of the Petri net model of concurrency used extensively in the modelling of systems, particularly in the organisation. Their semantics includes a time-based version and there are tools for their animation and analysis of properties.

The semantics of CPNs are, like HCSP, too complex to give here. However, a CPN that has similar phenomenal models (and so will synchronise with τ_2, when we have defined parallel composition) is shown in Figure 6.

For those unfamiliar with CPNs, Figure 6 shows a CPN with 5 *places* (circles; representing state) and 3 transitions (boxes; representing actions). A *transition* is *enabled* if each prior place (connected by an arc ending in the transition) has a *marking* (or value associated with it). In Figure 6, both the transition labelled 'Value to send' and that labelled 'syn' are enabled (the former as its prior places has marking 4; the latter since it has no prior places) and so can *fire*. Each fires independently of the other and, in doing so, removes the values from the prior places, populating the posterior places (those connected by an arc starting at the transition). The arcs define the way that this is achieved.

The possible 'runs' of the CPN of Figure 6 from the *initial marking* of the value 4 in place 'Value to send' are shown in Figure 7. There are three, corresponding to:

upper:	transition *ch!x ready* firing before *syn*
middle:	transition *syn* firing before *ch!x ready*
lower:	both transitions firing at the same time.

Note that, after the initial 'assignment' $x := 4$, the CPN waits for synchronisation with places *ch*! and *ch* marked.

IV. Parallel mechanism composition

Given two mdl expressions, each defining a set of phenomenal models, we can ask whether there are any phenomenal models that are *compatible* between the two. Here, by compatible, we mean each satisfies the expectations of the other for synchronisation. Formally, this is:

Definition 5 (Compatible phenomenal models): Two phenomenal models $M_i = \langle \mathcal{T}_i, V_i, W_i \rangle$, $i = 1, 2$ are compatible when they intersect and agree on $PHEN = PHEN_1 \cap PHEN_2$, and their associated channel

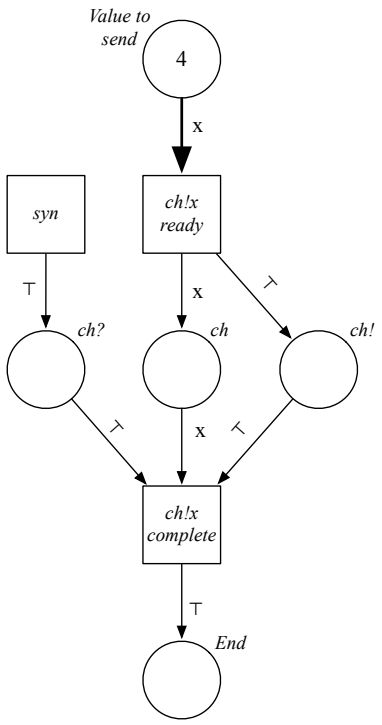

Figure 6. A Coloured Petri Net 'close to' $x := 4; ch!x$

flags[2], i.e.,

$$
\begin{array}{rcl}
\mathcal{T}_1 & = & \mathcal{T}_2 \\
dom(V_1) \cap dom(V_2) & = & PHEN \\
dom(W_1) \cap dom(W_2) & = & PHEN! \cup PHEN? \\
V_1(ch) & = & V_2(ch) \text{ for } ch \in PHEN \\
W_1(ch) & = & W_2(ch) \text{ for } ch \in PHEN! \cup PHEN?
\end{array}
$$

PHEN are called the *shared phenomena* between M_1 and M_2.

The conditions on the domains exists so that state variables are not available as a mechanism for communication. Of course, even if W_1 and W_2 share non-*PHEN* names, we can ensure the intersections are disjoint through suitable renamings, which we assume has been done in the sequel.

When compatible phenominal models do exist, their *join* satisfies both expressions at the same time, modulo the state variables and non-shared phenomena. This we will interpret as a model for their parallel composition.

We can join compatible models:

Definition 6 (Model Join): Given compatible phenomenal models M_1 and M_2 over $\mathcal{T}$ we define $M_1 \sqcup M_2$ over $PHEN_1 \cup PHEN_2$ and $STATE_1 \cup STATE_2$ thus:

$$M_1 \sqcup M_2 \triangleq \langle \mathcal{T}, V_1 \cup V_2, W_1 \cup W_2 \rangle$$

[2]In which $PHEN? = \{ch? \mid ch \in PHEN\}$ and $PHEN! = \{ch! \mid ch \in PHEN\}$

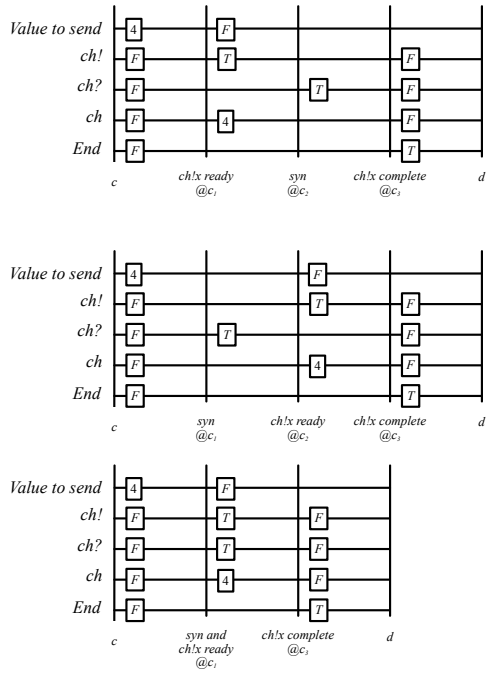

Figure 7. The phenomenal models for the CPN of Figure 6. Upper: *ch*!*x ready* fires before *syn* and, middle: *vice versa*. In lower, both *syn* and *ch*!*x ready* fire at the same time.

Proposition 1: $M_1 \sqcup M_2$ is a phenomenal model over $PHEN_1 \cup PHEN_2$ and $STATE_1 \cup STATE_2$.

With these definitions, we can define parallel composition:

Definition 7 (Parallel composition): Suppose we have two mdl expressions τ_i over $PHEN_i$ and $STATE_i$, $i = 1, 2$, respectively. Then we define

$$[\![\tau_1 \| \tau_2]\!] = \{M_1 \sqcup M_2 \mid M_i \in [\![\tau_i]\!] \wedge M_1,\ M_2 \text{ compatible}\}$$

There are a number of notable features of this definition:

- the definition is symmetric in its operands;
- the model joins identify channels and their flags, allowing the two expressions to 'synchronise and pass values'.

Example 2: For terms τ_1 and τ_2 above, the models on the right of the respective figures (the synchronisation cases) can be made compatible when $e = e' = f$ and $\beta = 4$. Taking their join as $M = M_1 \sqcup M_2$, we have the behaviour of the parallel composition as:

$$\begin{aligned} I(ch!)(w) &= \begin{cases} F, & w \in [c, f) \\ T, & w \in [f, f'] \end{cases} \\ I(ch?)(w) &= T, \quad w \in [c, f'] \\ V(ch)(w) &= 4, \quad w \in [f, f'] \\ V(x)(w) &= 4, \quad w \in [c, d] \\ V(y)(w) &= 4, \quad w \in [f, d] \\ V(z)(w) &= 4, \quad w \in [f', d] \end{aligned}$$

as illustrated in Figure 8

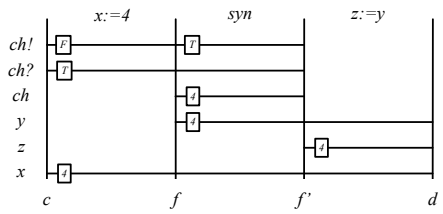

Figure 8. The phenomenal models for $\tau_1 \parallel \tau_2$: the behaviour is the join of the operand terms.

A. Impedance mismatch

We began this paper by saying that we might expect the language in which the mechanism is described to be chosen so that it is suitable for that mechanism, and so representative of its characteristics, such as the ways in which the generated phenomena synchronise. Clearly, the parallel composition of two mdls expressed in the same language will match flawlessly. However, expressions in different languages may not.

Consider, for instance, the parallel composition of HCSP term τ_2 and the CPN of Figure 6, through the phenomenal models of Figures 5 and 7. We noted that the CPN waits for synchronisation when places *ch*! and *ch* are marked.

However, for phenomenal model compatibility, we require that, like for $\tau_1 \parallel \tau_2$, the green intervals in Figure 5 must shrink to zero, and also that $c = c_1 = c_2$ in Figure 7. Given the waiting 'nature' of the CPN model for synchronisation, this may or may not be the expected outcome.

V. Discussion and Future work

The 'new mechanical philosophy' is emerging as a framework for thinking about science. Under this view, science is the discovery of mechanism the results of which are observable natural phenomena. In this paper, we have taken the first steps towards extending this philosophy to provide a mechanical basis for design and engineering, under our Problem Oriented Engineering framework. With our extension, design is the creation of (sub-)system descriptions the conglomeration of which is a solution to a problem. Because sub-systems may be diverse in their nature, we have also shown how diverse descriptions can be joined through parallel composition with phenomena sharing. This diverse sub-system composition is helpful in that it can reveal 'impedance mismatches' between the expectations for the sharing of phenomena between sub-systems.

This work in progress has, however, barely touched upon the complexity needed for a full theory, one that would, for instance, stand as the basis for POE. In particular, the notion of synchronisation that we use is that of HCSP, i.e., synchronous and unbuffered; whereas this covers other CSP process algebras, there are many other variants that would also need to be considered in a fuller theory.

Moreover, we have been very restrictive in assuming only real values state variables and channel communication in the mechanisms we have defined. Whereas this might be sufficient for real-world systems, it is too restrictive for synthesis: we might, for instance, be designing a software engineering process for which the phenomena occurrences include program code, documentation, code reviews, even scrums, *etc* or organisational governance with observable phenomena policy documents. This also raises the possibility that phenomena occurrences from one process are themselves mechanism descriptions, capable of themselves generating phenomena. This is not unlike the π-calculus [16] and would be interesting to examine any crossover.

Finally, throughout, we have relied on the visual nature of phenomenal models to illustrate more technical examples. That phenomenal models are so visual suggests non-formal and informal interactions with a mdl described system might be possible; for instance, a human generating behaviours in the form of graphs that have interpretations as phenomenal models and so being composable with formal descriptions. The third author is actively working through this idea with tools for phenomenal model visualisation.

Acknowledgements

The authors would like to thank the reviewers for their constructive comments.

References

[1] Z. Chaochen, W. Ji, and A. Ravn. A formal description of hybrid systems. *Hybrid Systems III*, pages 511–530, 1996.

[2] Zhou Chaochen, Charles Anthony Richard Hoare, and Anders P Ravn. A calculus of durations. *Information processing letters*, 40(5):269–276, 1991.

[3] Carl Craver and James Tabery. Mechanisms in science, 2015. URL https://plato.stanford.edu/entries/science-mechanisms/.

[4] Carl F Craver. *Explaining the brain: Mechanisms and the mosaic unity of neuroscience*. Oxford University Press, 2007.

[5] Stuart Glennan. Rethinking mechanistic explanation. *Philosophy of science*, 69(S3):S342–S353, 2002.

[6] Stuart S Glennan. Mechanisms and the nature of causation. *Erkenntnis*, 44(1):49–71, 1996.

[7] Jon G. Hall and Lucia Rapanotti. Assurance-driven design in Problem Oriented Engineering. *International Journal On Advances in Systems and Measurements*, 2 (1):119–130, 2009.

[8] Jon G Hall and Lucia Rapanotti. A design theory for software engineering. *Information and Software Technology*, 87:46–61, 2017.

[9] D. Harel and A. Naamad. The statemate semantics of statecharts. *ACM Transactions on Software Engineering and Methodology*, 5(4): 293-333, 1996.

[10] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, 1985.

[11] M. Jackson. *Problem Frames: Analyzing and Structuring Software Development Problems*. Addison-Wesley Publishing Company, 2001.

[12] Kurt Jensen, Lars Michael Kristensen, and Lisa Wells. Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer (STTT)*, 2007.

[13] Stuart A Kauffman. Articulation of parts explanation in biology and the rational search for them. In *Topics in the Philosophy of Biology*, pages 245–263. Springer, 1976.

[14] Peter Bernard Ladkin. *The logic of time representation*. PhD thesis, Citeseer, 1987.

[15] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[16] Robin Milner. *Communicating and mobile systems: the pi calculus*. Cambridge university press, 1999.

[17] Johan van Benthem. *The logic of time: a model-theoretic investigation into the varieties of temporal ontology and temporal discourse*, volume 156. Springer Science & Business Media, 2013.