Endowing Concurrent Kleene Algebra with Communication Actions

Jason Jaskolka, Ridha Khedri, and Qinglei Zhang

Department of Computing and Software, Faculty of Engineering
McMaster University, Hamilton, Ontario, Canada
{jaskolj,khedri,zhangq33}@mcmaster.ca

Abstract. Communication is integral to the understanding of agent interactions in concurrent systems. In this paper, we propose a mathematical framework for communication and concurrency called Communicating Concurrent Kleene Algebra (C^2KA). C^2KA extends concurrent Kleene algebra with the notion of communication actions. This extension captures both the influence of external stimuli on agent behaviour as well as the communication and concurrency of communicating agents.

Keywords: concurrency, communication, concurrent Kleene algebra, semimodules, specification, algebraic approaches to concurrency.

1 Introduction

Systems interact with other systems resulting in the development of patterns of stimuli-response relationships. Therefore, models for concurrency are commonly constructed upon the assumption of uninterruptible system execution or atomic events. Models for concurrency differ in terms of how they capture this notion. A coarse-grained classification categorises models for concurrency as either *state-based* models or *event-based* models [4]. State-based models describe the behaviour of a system in terms of the properties of its states. Typical state-based approaches consist of representing system properties as formulae of temporal logics, for example, such as LTL [26], CTL [2], or CTL* [5], and model-checking the state space of the system against them. Conversely, event-based models represent systems via structures consisting of atomic events. There is an extensive variety of examples of event-based models for concurrency including labelled transition systems [17], Petri nets [25], process calculi (e.g., CCS [22], CSP [7], ACP [1], and π-calculus [24]), Hoare traces [8], Mazurkiewicz traces [21], synchronisation trees [22], pomsets [27], and event structures [31].

Recently, Hoare et al. [9–12] proposed a formalism for modelling concurrency called *Concurrent Kleene Algebra* (CKA). CKA extends the algebraic framework provided by Kleene algebra by offering, aside from choice and finite iteration, operators for sequential and concurrent composition.

In this paper, we propose a mathematical framework for communication and concurrency called *Communicating Concurrent Kleene Algebra* (C^2KA). It extends the algebraic model of concurrent Kleene algebra and allows for the

P. Höfner et al. (Eds.): RAMiCS 2014, LNCS 8428, pp. 19–36, 2014.

separation of communicating and concurrent behaviour in a system and its environment. With C^2KA, we are able to express the influence of external stimuli on the behaviours of a system of agents resulting from the occurrence of external events either from communication among agents or from the environment of a particular agent. In this way, we can think about concurrent and communicating systems from two different perspectives: a behavioural perspective and an external event (stimulus) perspective. We can obtain a behavioural perspective by focussing on the behaviour of a particular agent in a communicating system and considering the influence of stimuli, from the rest of the world in which the agent resides, as transformations of the agent's behaviour. Similarly, we can obtain an external event perspective by considering the influence of agent behaviours as transformations of external stimuli. It provides a framework which presents a different view of communication and concurrency than what is traditionally given by existing process calculi.

The remainder of the paper is organised as follows. Section 2 discusses the notions of external stimuli and induced behaviours and introduces a hybrid view of agent communication. Section 3 provides the mathematical preliminaries needed for the remainder of this paper. Section 4 presents the proposed mathematical framework for communication and concurrency and the related results. Section 5 discusses the proposed framework and related work. Finally, Section 6 draws conclusions and points to the highlights of our current and future work.

2 Stimuli and Induced Behaviours

An essential aspect of concurrent systems is the notion of communication. As presented in [9–12], communication in CKA is not directly captured. Variables and communication channels are modelled as sets of traces. Communication can be perceived only when programs are given in terms of the dependencies of shared events [13]. One needs to instantiate the low-level model of programs and traces for CKA in order to define any sort of communication. We would like to have a way to specify communication in CKA without the need to articulate the state-based system of each action (i.e., at a convenient abstract level).

Furthermore, CKA does not directly deal with describing how the behaviours of agents in a system are influenced by external stimuli. From the perspective of behaviourism [30], a stimulus constitutes the basis for behaviour. In this way, agent behaviour can be explained without the need to consider the internal states of an agent. When examining the effects of external stimuli on agent behaviours, it is important to note that every external stimulus *invokes a response* from an agent. When the behaviour of an agent changes as a result of the response, we say that the external stimulus *influences* the behaviour of the agent. Moreover, it is important to have an understanding of how agent behaviours may evolve due to the influence of external stimuli. In particular, it is often useful to have an idea of the possible influence that any given external stimulus may have on a particular agent. We call these possible influences, the *induced behaviours* via external stimuli.

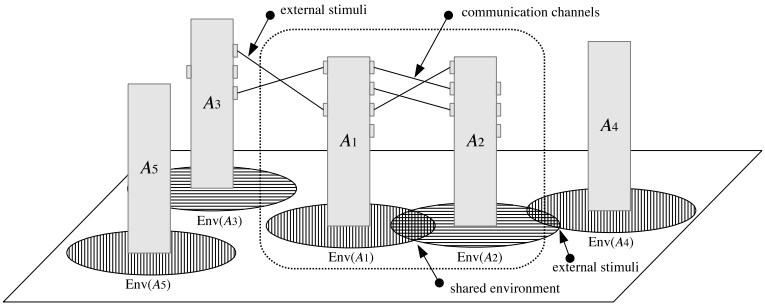

Fig. 1. A hybrid view of agent communication

Agents can communicate via their shared environment and through their local communication channels, but they may also be influenced by external stimuli. For example, if we consider agents A_1 and A_2 (dotted box) depicted in Figure 1, they have a shared environment through which they can communicate. Additionally, they have some communication channels at their disposal for sending and receiving messages. However, the behaviour of A_1 and A_2 can be influenced by the external stimuli coming from A_3, for example. The system formed by A_5 alone is a closed system and does not communicate with the rest of the world neither by external stimuli nor a shared environment. Consider the case where A_1 is subjected to an external stimulus from A_3. Then, A_1 may respond to the stimulus by changing its behaviour which can affect the communication between it and A_2. Currently, this notion cannot be directly handled with CKA. We would like to have a mathematical framework for systems of communicating agents which can capture both the influence of external stimuli on agent behaviour, as well as the communication and concurrency of agents at the abstract algebraic level.

3 Mathematical Background

In this section, we provide the mathematical preliminaries of monoids, semirings, Kleene algebras, and semimodules, and we introduce concurrent Kleene algebra.

3.1 Monoids, Semirings, Kleene Algebras, and Semimodules

A *monoid* is a mathematical structure $\big(S, \cdot, 1\big)$ consisting of a nonempty set S, together with an associative binary operation $\cdot$ and a distinguished constant 1 which is the identity with respect to $\cdot$. A monoid is called *commutative* if $\cdot$ is commutative and a monoid is called *idempotent* if $\cdot$ is idempotent.

A *semiring* is a mathematical structure $\big(S, +, \cdot, 0, 1\big)$ where $\big(S, +, 0\big)$ is a commutative monoid and $\big(S, \cdot, 1\big)$ is a monoid such that operator $\cdot$ distributes over operator $+$. We say that element 0 is *multiplicatively absorbing* if it annihilates S with respect to $\cdot$. We say that a semiring is *idempotent* if operator $+$ is idempotent. Every idempotent semiring has a natural partial order $\leq$ on S defined

by $a \leq b \iff a + b = b$. Operators $+$ and $\cdot$ are isotone on both the left and the right with respect to $\leq$.

Kleene algebra extends the notion of idempotent semirings with the addition of a unary operator for finite iteration.

Definition 1 (Kleene Algebra – e.g., [19]). *A* Kleene algebra *is a mathematical structure* $(K, +, \cdot, ^*, 0, 1)$ *where* $(K, +, \cdot, 0, 1)$ *is an idempotent semiring with a multiplicatively absorbing* 0 *and identity* 1 *and where the following axioms are satisfied for all* $a, b, c \in K$*:*

(i) $1 + a \cdot a^* = a^*$
(ii) $1 + a^* \cdot a = a^*$
(iii) $b + a \cdot c \leq c \implies a^* \cdot b \leq c$
(iv) $b + c \cdot a \leq c \implies b \cdot a^* \leq c$

An important notion required for the proposed framework for communication and concurrency is that of semimodules.

Definition 2 (Left $\mathcal{S}$-semimodule – e.g., [6]). *Let* $\mathcal{S} = (S, +, \cdot, 0_S, 1)$ *be a semiring and* $\mathcal{K} = (K, \oplus, 0_\mathcal{K})$ *be a commutative monoid. We call* ${}_\mathcal{S}(K, \oplus)$ *a* left $\mathcal{S}$-semimodule *if there exists a mapping* $S \times K \to K$ *denoted by juxtaposition such that for all* $s, t \in S$ *and* $a, b \in K$

(i) $s(a \oplus b) = sa \oplus sb$
(ii) $(s + t)a = sa \oplus sb$
(iii) $(s \cdot t)a = s(ta)$
(iv) ${}_\mathcal{S}(K, \oplus)$ *is called* unitary *if it also satisfies* $1a = a$
(v) ${}_\mathcal{S}(K, \oplus)$ *is called* zero-preserving *if it also satisfies* $0_\mathcal{S} a = 0_\mathcal{K}$

A right $\mathcal{S}$-semimodule can be defined analogously. From Definition 2, it is easy to see that each unitary left $\mathcal{S}$-semimodule ${}_\mathcal{S}(K, \oplus)$ has an embedded left $\mathcal{S}$-act ${}_\mathcal{S}K$ with respect to the monoid $(S, \cdot, 1)$. We say that ${}_\mathcal{S}K$ is a left $\mathcal{S}$-act if there exists a mapping satisfying Axioms (iii) and (iv) of Definition 2 [18].

3.2 Concurrent Kleene Algebra

Concurrent Kleene algebra is an algebraic framework extended from Kleene algebra offering operators for sequential and concurrent composition, along with those for choice and finite iteration. The operators for sequential and concurrent composition are related by an inequational form of the exchange axiom.

Definition 3 (Concurrent Kleene Algebra – e.g., [9]). *A* concurrent Kleene algebra (CKA) *is a structure* $(K, +, *, ;, ^{\circledast}, ^{\odot}, 0, 1)$ *such that* $(K, +, *, ^{\circledast}, 0, 1)$ *and* $(K, +, ;, ^{\odot}, 0, 1)$ *are Kleene algebras linked by the* exchange axiom *given by* $(a * b) ; (c * d) \leq (b ; c) * (a ; d)$.

A selection of laws for CKA which are needed for the remainder of this paper are found in [9] and are given in Proposition 1. An additional useful law is given in Proposition 2.

Proposition 1 (e.g., [9]). *For all* $a, b, c, d \in K$,

(i) $a * b = b * a$
(ii) $(a * b) ; (c * d) \leq (a ; c) * (b ; d)$
(iii) $a ; b \leq a * b$
(iv) $(a * b) ; c \leq a * (b ; c)$
(v) $a ; (b * c) \leq (a ; b) * c$

Proposition 2. *For all* $a \in K$, $a^{\textcircled{:}} \leq a^{\circledast}$.

Proof. The proof involves the application of Definition 1(iii), Definition 1(i), and Proposition 1(iii). The detailed proof can be found in Appendix A of [16].

4 The Proposed Framework

In the following subsections, we first articulate the algebraic structures which capture agent behaviours and external stimuli. After that, we use the aforementioned algebraic structures for agent behaviours and external stimuli to develop the proposed framework for communication and concurrency. Throughout this section, all omitted proofs can be found in Appendix A of [16].

4.1 A Simple Example of a System of Communicating Agents

We adapt a simple illustrative example from [23] to illustrate the basic notions of specifying a system of communicating agents using the proposed framework. Consider the behaviour of a one-place buffer. Suppose that the buffer uses two flags to indicate its current status. Let flag_1 denote the empty/full status of the buffer and let flag_2 denote the error status of the buffer. In this simple system of communicating agents, assume that there are two basic system agents, **P** and **Q**, which are responsible for controlling the buffer state flags flag_1 and flag_2, respectively. Throughout the following subsections, we illustrate how we can utilise the proposed framework to specify the communicating and concurrent behaviours of the agents **P** and **Q**, as well as the overall system behaviour of the one-place buffer.

4.2 Structure of Agent Behaviours

In [9–12], Hoare et al. presented the framework of concurrent Kleene algebra which captures the concurrent behaviour of agents. In this paper, we adopt the framework of CKA in order to describe agent behaviours in systems of communicating agents. In what follows, let $\mathcal{K} \stackrel{\text{def}}{=} \left(K, +, *, ;, ^{\circledast}, ^{\textcircled{:}}, 0, 1\right)$ be called a CKA.

It is important to note that throughout this paper, the term *agent* is used in the sense used by Milner in [23] to mean any system whose behaviour consists of discrete actions. In this way, an agent can be defined by simply describing its behaviour. Because of this, we may use the terms agents and behaviours interchangeably. With this understanding of agents, the support set K of the CKA $\mathcal{K}$ represents a set of possible behaviours. The operator $+$ is interpreted as a choice

between two behaviours, the operator ; is interpreted as a sequential composition of two behaviours, and the operator $*$ is interpreted as a parallel composition of two behaviours. The element 0 represents the behaviour of the *inactive agent* and the element 1 represents the behaviour of the *idle agent* just as in many process calculi. Moreover, associated with a CKA is a natural ordering relation $\leq_{\mathcal{K}}$ representing the sub-behaviour relation. For behaviours $a, b \in K$, $a \leq_{\mathcal{K}} b$ indicates that a is a sub-behaviour of b if and only if $a + b = b$.

For the one-place buffer example of Section 4.1, we consider the following set of events which are simple assignments to the buffer status flags:

$$P_1 \stackrel{\text{def}}{=} (\mathit{flag}_1 := \mathit{off}) \qquad Q_1 \stackrel{\text{def}}{=} (\mathit{flag}_2 := \mathit{off})$$
$$P_2 \stackrel{\text{def}}{=} (\mathit{flag}_1 := \mathit{on}) \qquad Q_2 \stackrel{\text{def}}{=} (\mathit{flag}_2 := \mathit{on})$$

In this way, K is generated by the set of basic behaviours $\{P_1, P_2, Q_1, Q_2, 0, 1\}$ where 0 is interpreted as abort and 1 is interpreted as skip.

4.3 Structure of External Stimuli

As mentioned in Section 2, a stimulus constitutes the basis for behaviour. Because of this, each discrete, observable event introduced to a system, such as that which occurs through the communication among agents or from the system environment, is considered to be an external stimulus which invokes a response from each system agent.

Definition 4 (Stimulus Structure). *Let* $\mathcal{S} \stackrel{\text{def}}{=} \big(S, \oplus, \odot, \mathfrak{d}, \mathfrak{n}\big)$ *be an idempotent semiring with a multiplicatively absorbing* $\mathfrak{d}$ *and identity* $\mathfrak{n}$*. We call* $\mathcal{S}$ *a* stimulus structure*.*

Within the context of external stimuli, S is the set of external stimuli which may be introduced to a system. The operator $\oplus$ is interpreted as a choice between two external stimuli and the operator $\odot$ is interpreted as a sequential composition of two external stimuli. The element $\mathfrak{d}$ represents the *deactivation stimulus* which influences all agents to become inactive and the element $\mathfrak{n}$ represents the *neutral stimulus* which has no influence on the behaviour of all agents. Furthermore, each stimulus structure has a natural ordering relation $\leq_{\mathcal{S}}$ representing the sub-stimulus relation. For external stimuli $s, t \in S$, we write $s \leq_{\mathcal{S}} t$ and say that s is sub-stimulus of t if and only if $s \oplus t = t$.

Continuing with the one-place buffer example of Section 4.1, suppose that the behaviour of each agent in the one-place buffer system is influenced by a number of external stimuli which either place an item in the buffer, remove an item from the buffer, or generate an error. We denote these stimuli by *in*, *out*, and *error* respectively. These external stimuli form a stimulus structure $\mathcal{S}$ where S is generated by the set of basic external stimuli $\{\mathit{in}, \mathit{out}, \mathit{error}, \mathfrak{d}, \mathfrak{n}\}$ where we interpret $\mathfrak{d}$ as a kill signal and $\mathfrak{n}$ as any stimulus with no influence that belongs to the complement of the set of external stimuli which may be introduced to a system.

4.4 Communicating Concurrent Kleene Algebra (C^2KA)

C^2KA extends the algebraic foundation of CKA with the notions of semimodules and stimulus structures to capture the influence of external stimuli on the behaviour of system agents.

Definition 5 (Communicating Concurrent Kleene Algebra). *A* Communicating Concurrent Kleene Algebra (C^2KA) *is a system* $\big(\mathcal{S}, \mathcal{K}\big)$*, where* $\mathcal{S} = \big(S, \oplus, \odot, \mathfrak{d}, \mathfrak{n}\big)$ *is a stimulus structure and* $\mathcal{K} = \big(K, +, *, \,;, \circledast, \odot, 0, 1\big)$ *is a* CKA *such that* $\big({}_{\mathcal{S}}K, +\big)$ *is a unitary and zero-preserving* left $\mathcal{S}$-semimodule *with mapping* $\circ : S \times K \to K$ *and* $\big(S_{\mathcal{K}}, \oplus\big)$ *is a unitary and zero-preserving* right $\mathcal{K}$-semimodule *with mapping*[1] $\lambda : S \times K \to S$*, and where the following axioms are satisfied for all* $a, b, c \in K$ *and* $s, t \in S$*:*

$$\textit{(i)}\ \ s \circ (a\,;b) = (s \circ a)\,;\big(\lambda(s,a) \circ b\big) \qquad \textit{(iii)}\ \ \lambda(s \odot t, a) = \lambda\big(s, (t \circ a)\big) \odot \lambda(t, a)$$
$$\textit{(ii)}\ \ c \leq_{\mathcal{K}} a \ \vee\ (s \circ a)\,;\big(\lambda(s,c) \circ b\big) = 0$$

In essence, a C^2KA consists of two semimodules which describe how the stimulus structure $\mathcal{S}$ and the CKA $\mathcal{K}$ mutually act upon one another in order to characterise the response invoked by an external stimulus on the behaviour of an agent as a next behaviour and a next stimulus.

First, the left $\mathcal{S}$-semimodule $\big({}_{\mathcal{S}}K, +\big)$ describes how the stimulus structure $\mathcal{S}$ acts upon the CKA $\mathcal{K}$ via the mapping $\circ$. We call $\circ$ the *next behaviour mapping* since it describes how an external stimulus invokes a behavioural response from a given agent. From $\big({}_{\mathcal{S}}K, +\big)$, we have that the next behaviour mapping $\circ$ distributes over $+$ and $\oplus$. Additionally, since $\big({}_{\mathcal{S}}K, +\big)$ is unitary, we have that the neutral stimulus has no influence on the behaviour of all agents and since it is zero-preserving, the deactivation stimulus influences all agents to become inactive. Second, the right $\mathcal{K}$-semimodule $\big(S_{\mathcal{K}}, \oplus\big)$ describes how the CKA $\mathcal{K}$ acts upon the stimulus structure $\mathcal{S}$ via the mapping λ. We call λ the *next stimulus mapping* since it describes how a new stimulus is generated as a result of the response invoked by a given external stimulus on an agent behaviour. From $\big(S_{\mathcal{K}}, \oplus\big)$, we have that the next stimulus mapping λ distributes over $\oplus$ and $+$. Also, since $\big(S_{\mathcal{K}}, \oplus\big)$ is unitary, we have that the idle agent forwards any external stimulus that acts on it and since $\big(S_{\mathcal{K}}, \oplus\big)$ is zero-preserving, the inactive agent always generates the deactivation stimulus.

In Definition 5, Axiom (i) describes the interaction of the next behaviour mapping $\circ$ with the sequential composition operator $;$ for agent behaviours. This axiom corresponds to the definition of the transition function for the cascading product (or synchronous serial composition) of Mealy automata [14]. Axiom (ii), which we call the *cascading output law*, states that when an external

[1] We use an infix notation for the next behaviour mapping $\circ$ and a prefix notation for the next stimulus mapping λ. We adopt these notations in an effort to reach out to those in the communities of monoid acts and Mealy automata since they adopt a similar non-uniform notation.

stimulus is introduced to the sequential composition $(a\,;b)$, then the cascaded stimulus must be generated by a sub-behaviour of a. In this way, the cascading output law ensures consistency between the next behaviour and next stimulus mappings with respect to the sequential composition of agent behaviours. It allows distributivity of $\circ$ over $;$ to be applied indiscriminately. Finally, Axiom (iii) describes the interaction of the next stimulus mapping λ with the sequential composition operator $\odot$ for external stimuli. This can be viewed as the analog of Axiom (i) with respect to the next stimulus mapping λ when considering the action of $\big(S_{\mathcal{K}}, \oplus\big)$.

In a given system of communicating agents, agent behaviour can be initiated in two ways. The first way to initiate agent behaviour in a system of communicating agents is by reactivation. We say that a C^{2}KA is *with reactivation* if $s \circ 1 \neq 1$ for some $s \in S\backslash\{\mathfrak{d}\}$. Consider the case where the idle agent 1 is not fixed with respect to some given external stimulus. Then, the passive idle agent could be influenced to behave as any active agent. In this case, we say that the agent has been *reactivated* as it then begins to actively participate in the system operation. If a C^{2}KA is without reactivation, then the idle agent 1 reflects an idle behaviour that is not influenced by any external stimulus other than the deactivation stimulus. In this case, the idle agent does not actively participate in the operation of a system and it cannot initiate agent behaviours. The second way in which agent behaviour can be initiated in a system of communicating agents is by external stimuli. In a C^{2}KA, we say that an agent $a \in K\backslash\{0,1\}$ is a *stimulus initiator* if and only if $\lambda(\mathfrak{n}, a) \neq \mathfrak{n}$. When an agent is a stimulus initiator then that agent may generate a new stimulus without outside influence. Because $\big(S_{\mathcal{K}}, \oplus\big)$ is unitary and zero-preserving, the inactive agent 0 and the idle agent 1 cannot be stimulus initiators. Intuitively, the inactive agent is not a stimulus initiator since it can only generate the deactivation stimulus to influence all other agents to cease their behaviours and become inactive. Likewise, the idle agent is not a stimulus initiator since it can be seen as having no state-changing observed behaviour and therefore it cannot generate any stimuli.

A Comment on a Model for C^{2}KA. In [9–12], we find the following model for CKA. Let EV be a set of event occurrences. A trace is a set of events and a program is a set of traces. The set of all traces over EV is denoted by $TR(EV) \stackrel{\text{def}}{=} \mathcal{P}(EV)$ and the set of all programs is denoted by $PR(EV) \stackrel{\text{def}}{=} \mathcal{P}(TR(EV))$. Obviously, $\big(PR(EV), \cup, *, ;, {}^{\circledast}, {}^{\odot}, \emptyset, \{\emptyset\}\big)$ is a CKA [9–12]. Moreover, the structure of external stimuli is modelled by sets of strings. In this way, it is easy to see that $\big(\mathcal{P}(\Lambda), \cup, \bullet, \emptyset, \{\epsilon\}\big)$ is a stimulus structure where Λ is a set of alphabet symbols, $\bullet$ denotes set concatenation, and ϵ is the empty string.

In a C^{2}KA, the semimodules $\big({}_{\mathcal{S}}K, +\big)$ and $\big(S_{\mathcal{K}}, \oplus\big)$ contain a left $\mathcal{S}$-act ${}_{\mathcal{S}}K$ and a right $\mathcal{K}$-act $K_{\mathcal{S}}$, respectively. It is well known that monoid acts can be considered as semiautomata [18, pg. 45]. By combining these two semiautomata, we obtain a Mealy automaton. A Mealy automaton is given by a five-tuple $\big(Q, \Sigma, \Theta, F, G\big)$ [14]. The set of states Q is a subset of $PR(EV)$ (i.e., the set K). In this way, each state of the Mealy automaton represents a possible program that can be executed by the system as a reaction to the stimulus

(input) leading to the state. The input alphabet Σ and output alphabet Θ are given by the stimulus structure such that $\Sigma = \Theta = S$. Finally, the transition function $F : \Sigma \times Q \to Q$ and the output function $G : \Sigma \times Q \to \Theta$ correspond to the next behaviour mapping $\circ : S \times K \to K$ and next stimulus mapping $\lambda : S \times K \to S$, respectively. These mappings respectively correspond to the transition functions from the semiautomata representations of ${}_{\mathcal{S}}K$ and $K_{\mathcal{S}}$.

The proposed model is also equipped with two operations for Mealy automata. The operation ; is associative and the operation + is associative, idempotent, and commutative. The ; operation corresponds to the *cascading product* of Mealy automata and the operation + corresponds to the *full direct product* of Mealy automata [14].

Proposition 3. *Let* $(\mathcal{S}, \mathcal{K})$ *be a* C^{2}KA*. For all* $a, b \in K$ *and* $s, t \in S$*:*

(i) $a \leq_{\mathcal{K}} b \;\wedge\; s \leq_{\mathcal{S}} t \implies s \circ a \leq_{\mathcal{K}} t \circ b$
(ii) $a \leq_{\mathcal{K}} b \;\wedge\; s \leq_{\mathcal{S}} t \implies \lambda(s,a) \leq_{\mathcal{S}} \lambda(t,b)$

The isotonicity laws of Corollary 1 follow immediately from Proposition 3. In [9], an idempotent semiring is called a *quantale* if the natural order induces a complete lattice and multiplication distributes over arbitrary suprema.

Corollary 1. *In a* C^{2}KA *where the underlying* CKA *and stimulus structure are built up from quantales, the following laws hold:*

(i) $a \leq_{\mathcal{K}} b \implies s \circ a \leq_{\mathcal{K}} s \circ b$
(ii) $s \leq_{\mathcal{S}} t \implies s \circ a \leq_{\mathcal{K}} t \circ a$
(iii) $s \circ (a\,;b + b\,;a) \leq_{\mathcal{K}} s \circ (a * b)$
(iv) $s \circ a^{\textcircled{:}} \leq_{\mathcal{K}} s \circ a^{\circledast}$
(v) $s \circ a^{\textcircled{:}} = +(n \mid n \geq 0 : s \circ a^n)$
(vi) $s \leq_{\mathcal{S}} t \implies \lambda(s,a) \leq_{\mathcal{S}} \lambda(t,a)$
(vii) $a \leq_{\mathcal{K}} b \implies \lambda(s,a) \leq_{\mathcal{S}} \lambda(s,b)$
(viii) $\lambda(s,(a\,;b + b\,;a)) \leq_{\mathcal{S}} \lambda(s,(a * b))$
(ix) $\lambda(s, a^{\textcircled{:}}) \leq_{\mathcal{S}} \lambda(s, a^{\circledast})$
(x) $\lambda(s, a^{\textcircled{:}}) = \oplus(n \mid n \geq 0 : \lambda(s, a^n))$

4.5 Specifying Systems of Communicating Agents with C^{2}KA

In order to specify a system of communicating agents using C^{2}KA, we have three levels of specification. Using the illustrative example of the one-place buffer from Section 4.1, we show how to specify the system agents using the proposed framework.

The *stimulus-response specification of agents* level gives the specification of the next behaviour mapping $\circ$ and the next stimulus mapping λ for each agent in the system. Assuming that we have a C^{2}KA without reactivation, the agent behaviours of **P** and **Q** are compactly specified as shown in Table 1. By composing the behaviours of **P** and **Q**, we are able to obtain the complete behaviour of the one-place buffer. The full stimulus-response specification of the buffer agent can be found in Table 3 in [16].

The *abstract behaviour specification* level restricts the specification to the desired behaviour of an agent in the communicating system by computing the responses to the external stimuli that can be introduced into the system in the given context. In the one-place buffer example, consider a context in which we

Table 1. Stimulus-Response Specification for Agents **P** and **Q**

$\mathbf{P} \stackrel{\text{def}}{=} P_1 + P_2$

$\circ_{\mathbf{P}}$	$\mathfrak{n}$	*in*	*out*	*error*
P_1	P_1	P_2	P_1	P_1
P_2	P_2	P_2	P_1	P_2

$\lambda_{\mathbf{P}}$	$\mathfrak{n}$	*in*	*out*	*error*
P_1	$\mathfrak{n}$	$\mathfrak{n}$	*error*	$\mathfrak{n}$
P_2	$\mathfrak{n}$	*error*	$\mathfrak{n}$	$\mathfrak{n}$

$\mathbf{Q} \stackrel{\text{def}}{=} Q_1 + Q_2$

$\circ_{\mathbf{Q}}$	$\mathfrak{n}$	*in*	*out*	*error*
Q_1	Q_1	Q_1	Q_1	Q_2
Q_2	Q_2	Q_2	Q_2	Q_2

$\lambda_{\mathbf{Q}}$	$\mathfrak{n}$	*in*	*out*	*error*
Q_1	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
Q_2	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$

$$\forall (P_i, Q_i \mid 1 \leq i \leq 2 : \mathfrak{d} \circ P_i = 0 \wedge \mathfrak{d} \circ Q_i = 0 \wedge \lambda(\mathfrak{d}, P_i) = \mathfrak{d} \wedge \lambda(\mathfrak{d}, Q_i) = \mathfrak{d})$$

only consider the buffer as behaving either as an empty buffer or as a full buffer. Furthermore, assume that the behaviour of the buffer may only be influenced by the introduction of *in* and *out* stimuli since these are the only stimuli that another external agent may have control over. This is to say that an external agent cannot issue an *error* since this is an uncontrollable stimulus which cannot be issued at will. In this way, after simple computation, we find that the abstract behaviour of the one-place buffer is given by $P_1 ; Q_1 + P_1 ; Q_2 + P_2 ; Q_1 + P_2 ; Q_2$. At the abstract behaviour specification level, C^{2}KA can be viewed as an event-based model of communication. In C^{2}KA, the left $\mathcal{S}$-semimodule $\left({}_{\mathcal{S}}K, +\right)$ and the right $\mathcal{K}$-semimodule $\left(S_{\mathcal{K}}, \oplus\right)$ allow us to specify how the external stimuli influence the behaviour of each agent in a given system. For this reason, this level of specification is best suited for describing message passing communication where agents transfer information explicitly through the exchange of data structures, either synchronously or asynchronously.

Finally, the *concrete behaviour specification* level provides the state-level specification of each agent behaviour (i.e., each program). At this level, we define the concrete programs for each of the CKA terms which specify each agent behaviour. The concrete behaviour specification provides the following state-level programs for each behaviour of the one-place buffer.

$$\begin{aligned}
\textsc{empty} &\stackrel{\text{def}}{=} P_1 ; Q_1 = \quad (\mathit{flag}_1 := \mathit{off} \ ; \ \mathit{flag}_2 := \mathit{off}) \\
\textsc{full} &\stackrel{\text{def}}{=} P_2 ; Q_1 = \quad (\mathit{flag}_1 := \mathit{on} \ ; \ \mathit{flag}_2 := \mathit{off}) \\
\textsc{underflow} &\stackrel{\text{def}}{=} P_1 ; Q_2 = \quad (\mathit{flag}_1 := \mathit{off} \ ; \ \mathit{flag}_2 := \mathit{on}) \\
\textsc{overflow} &\stackrel{\text{def}}{=} P_2 ; Q_2 = \quad (\mathit{flag}_1 := \mathit{on} \ ; \ \mathit{flag}_2 := \mathit{on})
\end{aligned}$$

Fig. 2. Concrete behaviour specification of the one-place buffer

Since C^{2}KA extends concurrent Kleene algebra, it inherits this model of communication from CKA. Just as in CKA, the instantiation of a low-level model of programs and traces for C^{2}KA affords the ability to specify communication through shared events and the dependencies between them. Because of this, this level of specification is best suited for shared-variable communication where agents transfer information through a shared medium such as variables, memory locations, etc.

Depending on which level of specification we are working at, the model can be viewed as either event-based or state-based. This gives flexibility in allowing us to choose which level is most suitable for the given problem. The context of the given problem will help to dictate at which level we need to work. For a full treatment of the illustrative example of the one-place buffer, the reader is referred to [16].

4.6 C²KA and Orbits, Stabilisers, and Fixed Points

Orbits, stabilisers, and fixed points are notions that allow us to perceive a kind of topology of a system with respect to the stimulus-response relationships among the system agents. Because of this, we are able to gain some insight into the communication channels that can be established among system agents. For example, with C²KA, we are able to compute the strong orbits (presented below) of the agent behaviours in a given system. The strong orbits represent the strongly connected agent behaviours in the system and therefore can provide some insight into the abilities of the agents in the same strong orbit to influence one another's behaviour through communication. Furthermore, having an idea of the topology of the system allows for the abstraction of components of the overall system behaviour. This kind of abstraction can aid in separating the communicating and concurrent behaviour in a system and its environment. Moreover, computing the orbits and stabilisers of agent behaviours can aid in the analysis and verification of systems of communicating agents, since it allows us to model the possible reaction of a system to a stimulus. Also, they allow us, in some cases, to reduce the analysis to only some relevant orbits of a system. Similarly, stabilisers allow us to reduce the analysis to studying only the stimuli that influence the behaviour of an agent. We conjecture that such reduction could, for example, alleviate the state explosion problem in model checking.

Since a C²KA consists of two semimodules $\left({}_{\mathcal{S}}K, +\right)$ and $\left(S_{\mathcal{K}}, \oplus\right)$ for which we have a left $\mathcal{S}$-act ${}_{\mathcal{S}}K$ and a right $\mathcal{K}$-act $S_{\mathcal{K}}$, we have two complementary notions of orbits, stabilisers, and fixed points within the context of agent behaviours and external stimuli, respectively. In this way, one can use these notions to think about concurrent and communicating systems from two different perspectives, namely the behavioural perspective provided by the action of external stimuli on agent behaviours described by $\left({}_{\mathcal{S}}K, +\right)$ and the external event (stimulus) perspective provided by the action of agent behaviours on external stimuli described by $\left(S_{\mathcal{K}}, \oplus\right)$. In this section, we focus only on the treatment of these notions with respect to the left $\mathcal{S}$-semimodule $\left({}_{\mathcal{S}}K, +\right)$ and agent behaviours. In a very similar way, we can present the same notions for the right $\mathcal{K}$-semimodule $\left(S_{\mathcal{K}}, \oplus\right)$ and external stimuli.

Definition 6 recalls the notions of orbits, stabilisers, and fixed points from the mathematical theory of monoids acting on sets [18].

Definition 6. *Let* $\left({}_{\mathcal{S}}K, +\right)$ *be the unitary and zero-preserving left* $\mathcal{S}$*-semimodule of a* C²KA *and let* $a \in K$.

(i) The orbit *of* a *in* $\mathcal{S}$ *is the set given by* $\mathrm{Orb}(a) = \{s \circ a \mid s \in S\}$.
(ii) The strong orbit *of* a *in* $\mathcal{S}$ *is the set given by* $\mathrm{Orb}_{\mathrm{S}}(a) = \{b \in K \mid \mathrm{Orb}(b) = \mathrm{Orb}(a)\}$.

(iii) The stabiliser *of a in $\mathcal{S}$ is the set given by* $\mathrm{Stab}(a) = \{s \in S \mid s \circ a = a\}$.
(iv) An element $a \in K$ is called a fixed point *if* $\forall(s \mid s \in S \backslash \{\mathfrak{d}\} : s \circ a = a)$.

We can define a preorder on K as $a \preceq_{\mathcal{K}} b \iff \mathrm{Orb}(a) \subseteq \mathrm{Orb}(b)$. Given this preorder, we can obtain an equivalence relation $\sim_{\mathcal{K}}$ from the intersection of $\preceq_{\mathcal{K}}$ and $\succeq_{\mathcal{K}}$. The equivalence classes of $\sim_{\mathcal{K}}$ give the strong orbits [20]. The strong orbits can also be viewed as the strongly connected components of a directed graph [29]. Additionally, when $a \in K$ is a fixed point, $\mathrm{Orb}(a) = \{0, a\}$ and $\mathrm{Stab}(a) = S \backslash \{\mathfrak{d}\}$. It is important to note that since $\big({}_{\mathcal{S}}K, +\big)$ is zero-preserving, every agent behaviour becomes inactive when subjected to the deactivation stimulus $\mathfrak{d}$. Because of this, we exclude this special case when discussing fixed agent behaviours.

Before we discuss the interplay between C^{2}KA and the notions of orbits, stabilisers, and fixed points, we first extend the partial order of sub-behaviours $\leq_{\mathcal{K}}$ to sets in order to express sets of agent behaviours encompassing one another.

Definition 7 (Encompassing Relation). *Let $A, B \subseteq K$ be two subsets of agent behaviours. We write $A \lessdot_{\mathcal{K}} B$ and say that A* is encompassed by *B (or B* encompasses *A) if and only if* $\forall\big(a \mid a \in A : \exists(b \mid b \in B : a \leq_{\mathcal{K}} b)\big)$.

The encompassing relation $\lessdot_{\mathcal{S}}$ for external stimuli can be defined similarly.

Orbits. The orbit of an agent $a \in K$ represents the set of all possible behavioural responses from an agent behaving as a to any external stimulus from $\mathcal{S}$. In this way, the orbit of a given agent can be perceived as the set of all possible future behaviours for that agent. With regard to the specification of the one-place buffer, we can compute the orbits of each of the buffer behaviours. For instance, $\mathrm{Orb}(\textsc{empty}) = \{\textsc{empty}, \textsc{full}, \textsc{underflow}, \textsc{overflow}\}$.

Proposition 4 provides an isotonicity law with respect to the orbits and the encompassing relation for agent behaviours.

Proposition 4. *Let $(\mathcal{S}, \mathcal{K})$ be a* C^{2}KA. *Then, $a \leq_{\mathcal{K}} b \implies \mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(b)$ for all $a, b \in K$.*

A selection of additional properties follow immediately from Proposition 4 and are given in Corollary 2.

Corollary 2. *In a* C^{2}KA *the following laws hold for all $a, b, c \in K$:*

(i) $\mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(a + b)$
(ii) $\mathrm{Orb}((a * b) ; (c * d)) \lessdot_{\mathcal{K}} \mathrm{Orb}((a ; c) * (b ; d))$
(iii) $\mathrm{Orb}(a ; b) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * b)$
(iv) $\mathrm{Orb}(a ; b + b ; a) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * b)$
(v) $\mathrm{Orb}((a * b) ; c) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * (b ; c))$
(vi) $\mathrm{Orb}(a ; (b * c)) \lessdot_{\mathcal{K}} \mathrm{Orb}((a ; b) * c)$
(vii) $\mathrm{Orb}(a^{\textcircled{\scriptsize :}}) \lessdot_{\mathcal{K}} \mathrm{Orb}(a^{\circledast})$
(viii) $\mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(c) \wedge \mathrm{Orb}(b) \lessdot_{\mathcal{K}} \mathrm{Orb}(c) \iff \mathrm{Orb}(a) \cup \mathrm{Orb}(b) \lessdot_{\mathcal{K}} \mathrm{Orb}(c)$

As stated before, without discussing the properties derived from the right $\mathcal{K}$-semimodule $\big(S_{\mathcal{K}}, \oplus\big)$, due to the cascading output law (see Definition 5 (ii)), we also have that $\mathrm{Orb}((s \circ a) ; \big(\lambda(s, c) \circ b\big)) = \{0\}$ for any $(a ; b) \in K$ and $\neg(c \leq_{\mathcal{K}} a)$.

Another Interpretation of Orbits. As mentioned in Section 2, we call the influence of external stimuli on agent behaviours the *induced behaviours* via external stimuli. The notion of induced behaviours allows us to make some predictions about the evolution of agent behaviours in a given system by providing some insight into the topology of the system and how different agents can respond to any external stimuli. Here, we provide a formal treatment of the notion of induced behaviours. While studying induced behaviours, we focus particularly on the next behaviour mapping $\circ$ and the effects of external stimuli on agent behaviours since we are interested in examining the evolution of agent behaviours via the influence of external stimuli in a given system of communicating agents.

Definition 8 (Induced Behaviour). *Let $a, b \in K$ be agent behaviours such that $a \neq b$. We say that b is* induced by a via external stimuli *(denoted by $a \lhd b$) if and only if* $\exists(s \mid s \in S : s \circ a = b)$.

Equivalently, we can express $a \lhd b \iff b \in \mathrm{Orb}(a)$ for $a \neq b$. In this way, it can be seen that the orbit of a behaviour a represents the set of all behaviours which are induced by a via external stimuli. Considering the one-place buffer example, it is plain to see, for instance, that EMPTY $\lhd$ UNDERFLOW via the external stimulus *out* and EMPTY $\lhd$ OVERFLOW via the external stimulus $in \odot in$.

Strong Orbits. Two agents are in the same strong orbit, denoted $a \sim_{\mathcal{K}} b$ for $a, b \in K$, if and only if their orbits are identical. This is to say when $a \sim_{\mathcal{K}} b$, if an agent behaving as a is influenced by an external stimulus to behave as b, then there exists an external stimulus which influences the agent, now behaving as b, to revert back to its original behaviour a. Furthermore, if $a \sim_{\mathcal{K}} b$, then $\exists(s, t \mid s, t \in S : s \circ a = b \wedge t \circ b = a)$. In this case, the external stimuli s and t can be perceived as *inverses* of one another and allow us to revert an agent back to its original behaviour since $t \circ s \circ a = a$ and $s \circ t \circ b = b$ (i.e., $s \odot t \in \mathrm{Stab}(a)$ and $t \odot s \in \mathrm{Stab}(b)$). In the specification of the one-place buffer, we have two strong orbits, namely, those given by $\{\text{EMPTY}, \text{FULL}\}$ and $\{\text{UNDERFLOW}, \text{OVERFLOW}\}$ which represent the behaviours from agents $\mathbf{P}$ and $\mathbf{Q}$, respectively. This is to say that we have (EMPTY $\sim_{\mathcal{K}$ FULL) and (UNDERFLOW $\sim_{\mathcal{K}$ OVERFLOW).

Stabilisers. For any agent $a \in K$, the stabiliser of a represents the set of external stimuli which have no observable influence (or act as neutral stimuli) on an agent behaving as a. In the illustrative example of the one-place buffer, we can compute the stabilisers of each of the buffer behaviours from the specification of the buffer agent. For example, $\mathrm{Stab}(\text{EMPTY})$ is generated by $\{error, in \odot out\}$.

By straightforward calculation and the definition of the encompassing relation $\preccurlyeq_{\mathcal{S}}$ for external stimuli, we have that $\mathrm{Stab}(a) \cap \mathrm{Stab}(b) \preccurlyeq_{\mathcal{S}} \mathrm{Stab}(a + b)$ for $a, b \in K$. However, consider a case where $\exists(s \mid s \in S : s \circ a = b \wedge s \circ b = a)$. Then, $s \notin \mathrm{Stab}(a)$ and $s \notin \mathrm{Stab}(b)$ but $s \in \mathrm{Stab}(a + b)$. Therefore, it is easy to see that in general $\neg\big(\mathrm{Stab}(a + b) \preccurlyeq_{\mathcal{S}} \big(\mathrm{Stab}(a) \cap \mathrm{Stab}(b)\big)\big)$ and $\neg\big(\mathrm{Stab}(a + b) \preccurlyeq_{\mathcal{S}} \big(\mathrm{Stab}(a) \cup \mathrm{Stab}(b)\big)\big)$.

Fixed Points. Depending on the given specification of a system of communicating agents, there may be any number of fixed points with respect to the next behaviour mapping $\circ$. When an agent behaviour is a fixed point, it is not influenced by any external stimulus other than the deactivation stimulus $\mathfrak{d}$. For example, with regard to the specification of agents for the one-place buffer example, it is easy to see that the behaviour Q_2 is a fixed point. The existence of fixed point behaviours is important when considering how agents can communicate via external stimuli. For instance, an agent that has a fixed point behaviour, does not have any observable response to any external stimuli (except for the deactivation stimulus) and therefore it can be seen that such an agent cannot be a receiver in any sort of communication via external stimuli.

Proposition 5 gives a selection of properties regarding fixed agent behaviours.

Proposition 5. *Let* $(\mathcal{S}, \mathcal{K})$ *be a* $\mathrm{C}^2\mathrm{KA}$ *and let* $a, b \in K$ *such that* a *and* b *are fixed points. We have:*

(i) 0 *is a fixed point*
(ii) $a + b$ *is a fixed point*
(iii) $a\,;b$ *is a fixed point*
(iv) $a^{\textcircled{\scriptsize :}}$ *is a fixed point if additionally* $(\mathcal{S}, \mathcal{K})$ *is without reactivation*

In Proposition 5, Identity (i) states that the inactive agent 0 is a fixed point with respect to the next behaviour mapping $\circ$. In this way, the inactive agent is not influenced by any external stimulus. Similarly, we can see that the deactivation stimulus $\mathfrak{d}$ is a fixed point with respect to the next stimulus mapping λ if we consider the notion of a fixed point in terms of external stimuli. Identity (ii) (resp. (iii) and (iv)) state that the choice (resp. sequential composition and sequential iteration) of fixed point behaviours results in a fixed point behaviour. In general, even if $a, b \in K$ are both fixed points, we are unable to say anything about $(a * b)$ as a fixed point.

Proposition 6 provides further insight into how the topology of a system of communicating agents can be perceived using $\mathrm{C}^2\mathrm{KA}$ and the notion of induced behaviours.

Proposition 6. *Let* $a, b, c \in K$ *be agent behaviours.*

(i) a *is a fixed point* $\implies \forall(b \mid b \in K \ \wedge\ b \neq 0 \ \wedge\ b \neq a \ :\ \neg(a \lhd b))$
(ii) $a \sim_{\mathcal{K}} b \implies a \lhd b \ \wedge\ b \lhd a$
(iii) $a \sim_{\mathcal{K}} b \implies (a \lhd c \iff b \lhd c)$

Proposition 6(i) states that if an agent has a fixed point behaviour, then it does not induce any agent behaviours via external stimuli besides the inactive behaviour 0. This is a direct consequence of the fact that an agent with a fixed point behaviour is not influenced by any external stimuli (except for the deactivation stimulus $\mathfrak{d}$) and therefore remains behaving as it is. Proposition 6(ii) states that all agent behaviours which belong to the same strong orbit are mutually induced

via some (possibly different) external stimuli. This is to say that if two agent behaviours are in the same strong orbit, then there exists inverse stimuli for each agent behaviour in a strong orbit allowing an agent to revert back to its original behaviour. Finally, Proposition 6(iii) states that if two agent behaviours are in the same strong orbit, then a third behaviour can be induced via external stimuli by either of the behaviours within the strong orbit. This is to say that each behaviour in a strong orbit can induce the same set of behaviours (perhaps via different external stimuli). Therefore, the strong orbit to which these behaviours belong can be abstracted and perceived as an equivalent agent behaviour with respect to the behaviours which it can induce via external stimuli.

5 Related Work and Discussion

Existing state-based and event-based formalisms for communication and concurrency such as temporal logics, labelled transition systems, Petri nets, and process calculi are primarily interested in modelling the behaviour of a system either in terms of the properties of its states or in terms of the observability of events. However, they do not directly, if at all, provide a hybrid model of communication and concurrency which encompass the characteristics of both state-based and event-based models. Concurrent Kleene algebra is perhaps the closest formalism to providing such a hybrid model. While CKA can be perceived as a hybrid model for concurrency, the same cannot be said for communication since communication in CKA is not directly evident.

C^2KA offers an algebraic setting which can capture both the influence of external stimuli on agent behaviour as well the communication and concurrency of agents at the abstract algebraic level. It uses notions from classical algebra to extend the algebraic foundation provided by CKA. If we consider a C^2KA with a trivial stimulus structure (i.e., $S = \{\mathfrak{n}\}$), then the next behaviour and next stimulus mappings are trivial and the C^2KA reduces to a CKA.

In the past, communication has been studied in process algebras such as CCS and CSP. As discussed in [9, 11, 12], some analogies can be made between relating CKA with process algebras. Therefore, if we consider the case where we have a trivial stimulus structure, then we can make the same kind of analogies relating C^2KA with existing process algebras.

In [9–12], Hoare et al. have taken steps towards investigating some aspects of communication through the derivation of rules for a simplified rely/guarantee calculus using CKA. However, this kind of communication is only captured via shared events. Since the proposed framework provides an extension of CKA, it is also capable of achieving these results. Furthermore, C^2KA supports the ability to work in either a state-based model (as illustrated by Figure 2) or an event-based model (as illustrated by Table 1) for the specification of concurrent and communicating systems. It gives us the ability to separate the communicating and concurrent behaviour in a system and its environment. This separation of concerns allows us to consider the influence of stimuli from the world in which the agent resides as transformations of agent behaviour and

yields the three levels of specification offered by C^2KA. With these levels of specification, C^2KA is able to capture the notions of message passing communication and shared-variable communication consistent with the hybrid view of agent communication depicted in Figure 1. Specifically, at the abstract behaviour specification level, we are interested only in the behaviour of an agent as dictated by the stimulus-response relationships that exist in the given system. In this way, the behaviour of an agent is dictated by its responses to external stimuli without the need to articulate the internal state-based system of each behaviour. On the other hand, by instantiating a concrete model of agent behaviour, such as that of programs and traces similar to what is done with CKA [9–12] at the concrete behaviour specification level, we have the ability to define the state-based model of agent behaviour. In this way, if the given problem requires insight into how external stimuli are processed by an agent, the concrete behaviour specification level affords the ability to specify such internal states of agent behaviours in terms of programs on concrete state variables. Because of this, C^2KA is flexible in allowing the context of the given problem to dictate which level of abstraction is most suitable. For example, if the given problem need not worry about the internal states of agent behaviours, then we can specify the system at the abstract behaviour specification level without any modifications to the proposed framework. Moreover, C^2KA inherits the algebraic foundation of CKA with all of its models and theory.

6 Conclusion and Future Work

In this paper, we proposed a mathematical framework for communication and concurrency called Communicating Concurrent Kleene Algebra (C^2KA). C^2KA extends the algebraic setting of concurrent Kleene algebra with semimodules in order to capture the influence of external stimuli on the behaviour of system agents in addition to the communication among agents through shared variables and communication channels. C^2KA supports the ability to work in either a state-based or event-based model for both the specification of communicating and concurrent behaviour by providing three levels of specification which reflect different levels of abstraction for the behaviour of agents in a given system. To the best of our knowledge, such a formalism does not currently exist in the literature and is required for dealing with problems such as studying the necessary conditions for covert channel existence [15]. A hybrid view of communication among agents and the influence of external stimuli on agent behaviour needs to be considered when examining the potential for communication condition for covert channels. Because of the separation of communicating and concurrent behaviour, we expect that C^2KA can aid in designing and analysing systems which are robust against covert communication channels. Since it provides a means for specifying systems of communicating agents, C^2KA can be an integral part of verifying the necessary conditions for covert channels [15]. We are using it to formalise and verify the potential for communication condition for covert channel existence. Also, we are developing a prototype tool using the Maude term

rewriting system [3] to support the automated computation and specification of systems of communicating agents using C^2KA. In future work, we aim to examine the ability to adapt C^2KA for use in solving interface equations (e.g., [28]) which can allow for implicit agent behaviour specifications in a variety of application domains. Furthermore, we intend to further investigate the theory and use of C^2KA to capture and explain the influence of external stimuli on agent behaviour in social networking environments.

Acknowledgements. This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the grant RGPIN227806-09 and the NSERC PGS D program. We would also like to thank the anonymous reviewers for their valuable comments which helped us considerably improve the quality of the paper.

References

1. Bergstra, J., Klop, J.: Process algebra for synchronous communication. Information and Control 60(1-3), 109–137 (1984)
2. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Kozen, D. (ed.) Logic of Programs 1981. LNCS, vol. 131, pp. 52–71. Springer, Heidelberg (1982)
3. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: The Maude 2.0 System. In: Nieuwenhuis, R. (ed.) RTA 2003. LNCS, vol. 2706, pp. 76–87. Springer, Heidelberg (2003)
4. Cleaveland, R., Smolka, S.: Strategic directions in concurrency research. ACM Computing Surveys 28(4), 607–625 (1996)
5. Emerson, E., Halpern, J.: "sometimes" and "not never" revisited: On branching versus linear time temporal logic. Journal of the ACM 33(1), 151–178 (1986)
6. Hebisch, U., Weinert, H.: Semirings: Algebraic Theory and Applications in Computer Science. Series in Algebra, vol. 5. World Scientific (1993)
7. Hoare, C.: Communicating sequential processes. Communications of the ACM 21(8), 666–677 (1978)
8. Hoare, C.: Communicating Sequential Processes. Prentice-Hall (1985)
9. Hoare, C., Möller, B., Struth, G., Wehrman, I.: Concurrent Kleene algebra. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009. LNCS, vol. 5710, pp. 399–414. Springer, Heidelberg (2009)
10. Hoare, C., Möller, B., Struth, G., Wehrman, I.: Foundations of concurrent Kleene algebra. In: Berghammer, R., Jaoua, A.M., Möller, B. (eds.) RelMiCS/AKA 2009. LNCS, vol. 5827, pp. 166–186. Springer, Heidelberg (2009)
11. Hoare, C., Möller, B., Struth, G., Wehrman, I.: Concurrent Kleene algebra and its foundations. Tech. Rep. CS-10-04, University of Sheffield, Department of Computer Science, Sheffield, UK (August 2010), `http://www.dcs.shef.ac.uk/~georg/ka/`
12. Hoare, C., Möller, B., Struth, G., Wehrman, I.: Concurrent Kleene algebra and its foundations. Journal of Logic and Algebraic Programming 80(6), 266–296 (2011)
13. Hoare, C., Wickerson, J.: Unifying models of data flow. In: Broy, M., Leuxner, C., Hoare, C. (eds.) Proceedings of the 2010 Marktoberdorf Summer School on Software and Systems Safety, pp. 211–230. IOS Press (August 2011)

14. Holcombe, W.: Algebraic Automata Theory. Cambridge Studies in Advanced Mathematics. Cambridge University Press (2004)
15. Jaskolka, J., Khedri, R., Zhang, Q.: On the necessary conditions for covert channel existence: A state-of-the-art survey. Procedia Computer Science 10, 458–465 (2012); Proceedings of the 3rd International Conference on Ambient Systems, Networks and Technologies, ANT 2012 (2012)
16. Jaskolka, J., Khedri, R., Zhang, Q.: Foundations of communicating concurrent Kleene algebra. Tech. Rep. CAS-13-07-RK, McMaster University, Hamilton, Ontario, Canada (November 2013), http://www.cas.mcmaster.ca/cas/0template1.php?601
17. Keller, R.: Formal verification of parallel programs. Communications of the ACM 19(7), 371–384 (1976)
18. Kilp, M., Knauer, U., Mikhalev, A.: Monoids, Acts And Categories With Applications to Wreath Products and Graphs: A Handbook for Students and Researchers. De Gruyter Expositions in Mathematics Series, vol. 29. Walter de Gruyter (2000)
19. Kozen, D.: Automata and Computability. Undergraduate Texts in Computer Science. Springer (1997)
20. Linton, S., Pfeiffer, G., Robertson, E., Ruškuc, N.: Computing transformation semigroups. Journal of Symbolic Computation 33(2), 145–162 (2002)
21. Mazurkiewicz, A.: Trace theory. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) APN 1986. LNCS, vol. 255, pp. 278–324. Springer, Heidelberg (1987)
22. Milner, R.: A Calculus of Communication Systems. LNCS, vol. 92. Springer, Heidelberg (1980)
23. Milner, R.: Communication and Concurrency. Prentice-Hall International Series in Computer Science. Prentice Hall (1989)
24. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes part I. Information and Computation 100(1), 1–40 (1992)
25. Petri, C.: Kommunikation mit Automaten. Ph.D. thesis, Institut für instrumentelle Mathematik, Bonn, Germany (1962), English translation available as: Communication with Automata, Technical Report RADC-TR-65-377, vol. 1, supplement 1, Applied Data Research, Princeton, NJ (1966)
26. Pnueli, A.: The temporal logic of programs. In: Proceedings of the 18th Annual Symposium on Foundations of Computer Science, pp. 46–57 (1977)
27. Pratt, V.: Modeling concurrency with partial orders. International Journal of Parallel Programming 15(1), 33–71 (1986)
28. Shields, M.: Implicit system specification and the interface equation. The Computer Journal 32(5), 399–412 (1989)
29. Steinberg, B.: A theory of transformation monoids: Combinatorics and representation theory. The Electronic Journal of Combinatorics 17(1) (2010)
30. Watson, J.: Behaviorism. University of Chicago Press (1930)
31. Winskel, G.: Event structures. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) APN 1986. LNCS, vol. 255, pp. 325–392. Springer, Heidelberg (1987)