Foundations of Communicating Concurrent Kleene Algebra

by

Jason Jaskolka, Ridha Khedri, and Qinglei Zhang

CAS Report Series CAS-13-07-RK

Department of Computing and Software November 2013

Information Technology Building

McMaster University

1280 Main Street West Hamilton, Ontario, Canada L8S 4K1

Foundations of Communicating Concurrent Kleene Algebra

Jason Jaskolka, Ridha Khedri and Qinglei Zhang

Department of Computing and Software, Faculty of Engineering,
McMaster University, Hamilton, Ontario, Canada

Technical Report CAS-13-07-RK
Department of Computing and Software
McMaster University

November 11, 2013*

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 supports the ability to work in either a state-based or event-based model for the specification of concurrent and communicating systems by extending 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. We also illustrate the different levels of abstraction for the behaviour of agents that are offered by the proposed framework with the specification of a simple illustrative example.

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

**Revised*: February 5, 2015

Contents

1 Introduction and Motivation

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 [3]. 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 [27], CTL [2], or CTL* [4], 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 [16], Petri nets [26], process calculi (e.g., CCS [22], CSP [6], ACP [1], and π-calculus [24]), Hoare traces [7], Mazurkiewicz traces [20], synchronisation trees [22], pomsets [28], and event structures [32].

Recently, Hoare et al. [8, 9, 10, 11] 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. It can be perceived as a hybrid model of concurrency as it encompasses the characteristics of both state-based models and event-based models. If we consider CKA at the abstract algebraic level, it is similar to an event-based model. In the algebraic structure of CKA, the carrier set is a set of programs and the language of the algebra is used to specify the behaviour of the system. On the other hand, if we consider CKA instantiated with the concrete model where programs are explicitly given as commands (as the ones used in [13, 25]), it can be viewed as a state-based model. In this way, the executions of the programs show how the current system state evolves and properties of each system state can be checked with pre- and post-condition assertions.

In this paper, we propose a mathematical framework for communication and concurrency called *Communicating Concurrent Kleene Algebra* (C^{2}KA). It extends the algebraic model of concurrent Kleene algebra and allows for the separation of communicating and concurrent behaviour in a system and its environment. With C^{2}KA, we are able to express the influence of external stimuli on the behaviours of system agents resulting from the occurrence of external events, from communication among agents, such as sending and receiving actions, or from the environment of a particular agent, such as an environmental parameter passing a certain threshold. In this way, we can think about concurrent and communicating systems from two different perspectives. 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 (stimulus) perspective by considering the influence of agent behaviours as transformations of external stimuli. C^{2}KA also allows for the specification of message passing and shared-variable communication as found in many existing formalisms for concurrency and communication. 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 this paper is organised as follows. In Section 2, we discuss the notions of external stimuli and induced behaviours and introduces a hybrid view of agent communication. In Section 3, we provide the mathematical preliminaries needed for the remainder of this paper. In Section 4, we present the proposed mathematical framework for communication and concurrency and the related results. In Section 5, we give a simple illustrative example of how to specify a system of communicating agents using the proposed framework. In Section 6, we discuss the proposed framework and related work. Lastly, in Section 7 we draw conclusions and point 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 [8, 9, 10, 11], 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 [12]. 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). This would allow us to work at the abstract algebraic level and then instantiate a concrete model when needed.

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, 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 [31]. 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. This is to say that induced behaviours indicate the influence of external stimuli on agent behaviours.

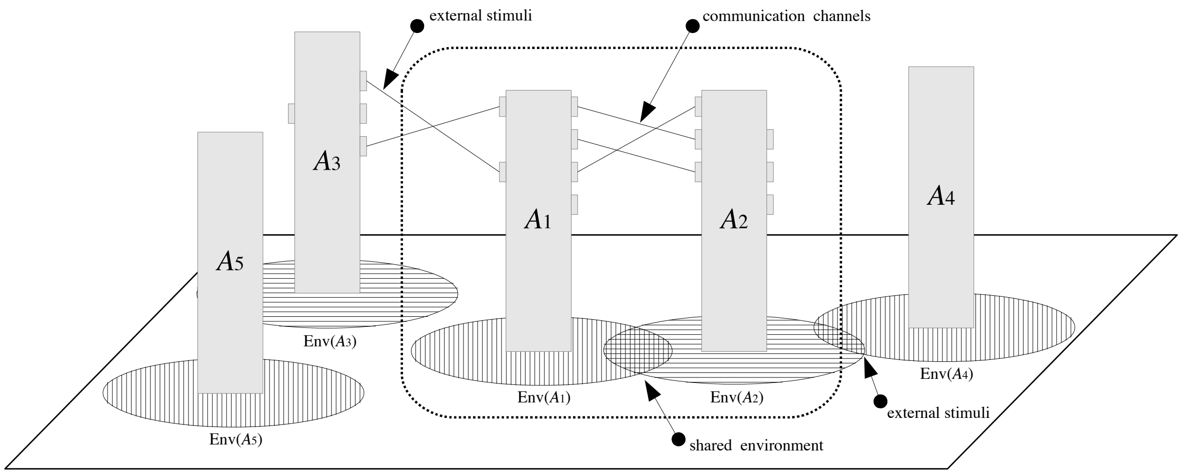

Figure 1: A hybrid view of agent communication.

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, and Kleene algebras, introduce concurrent Kleene algebra, and give the required background of semimodules.

3.1 Monoids, Semirings, and Kleene Algebras

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., [18]). *A* Kleene algebra *is a mathematical structure* $\big(K, +, \cdot, ^*, 0, 1\big)$ *where* $\big(K, +, \cdot, 0, 1\big)$ *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) $b + c \cdot a \leq c \implies b \cdot a^* \leq c$

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 2 (Concurrent Kleene Algebra – e.g., [8])**.** *A* concurrent Kleene algebra (CKA) *is a structure* $\big(K, +, *, \,;\, , ^{\circledast}, ^{\textcircled{;}}, 0, 1\big)$ *such that* $\big(K, +, *, ^{\circledast}, 0, 1\big)$ *and* $\big(K, +, \,;\, , ^{\textcircled{;}}, 0, 1\big)$ *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 [8] and are given in Proposition 1.

Proposition 1 (e.g., [8])**.** *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) $a \,;\, (b * c) \leq (a \,;\, b) * c$

An additional useful law is given in Proposition 2.

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 is given in Appendix A.1. □

3.3 Semimodules

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

Definition 3 (Left $\mathcal{S}$-semimodule – e.g., [5])**.** *Let* $\mathcal{S} = \big(S, +, \cdot, 0_{\mathcal{S}}, 1\big)$ *be a semiring and* $\mathcal{K} = \big(K, \oplus, 0_{\mathcal{K}}\big)$ *be a commutative monoid. We call* $\big({}_{\mathcal{S}}K, \oplus\big)$ *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) $\big({}_{\mathcal{S}}K, \oplus\big)$ *is called* unitary *if it also satisfies* $1a = a$

(v) $\big({}_{\mathcal{S}}K, \oplus\big)$ *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 3, it is easy to see that each unitary left $\mathcal{S}$-semimodule $\big({}_{\mathcal{S}}K, \oplus\big)$ has an embedded left $\mathcal{S}$-act ${}_{\mathcal{S}}K$ with respect to the monoid $\big(S, \cdot, 1\big)$.

4 The Proposed Framework

In the following sections, 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.

4.1 Structure of Agent Behaviours

In [8, 9, 10, 11], 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 $\mathcal{K} \stackrel{\text{def}}{=} \big(K, +, *, \,;\,, ^{\circledast}, ^{\odot}, 0, 1\big)$ 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 $a + b = b$.

4.2 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 $s \oplus t = t$.

4.3 Communicating Concurrent Kleene Algebra (C^{2}KA)

C^{2}KA 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^{2}KA) *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 $\lambda : S \times K \to S$, and where the following axioms are satisfied for all $a, b, c \in K$ and $s, t \in S$:*

(i) $s \circ (a \,;b) = (s \circ a) \,; \big(\lambda(s,a) \circ b\big)$ *(iii)* $\lambda(s \odot t, a) = \lambda\big(s, (t \circ a)\big) \odot \lambda(t,a)$

(ii) $a \leq_{\mathcal{K}} c \ \vee \ b = 1 \vee (s \circ a) \,; \big(\lambda(s,c) \circ b\big) = 0$

In essence, a C^{2}KA 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 roughly corresponds to the definition of the transition function for the cascading product (or synchronous serial composition) of Mealy automata [14]. Axiom (ii), which is referred to as the *cascading output law*, states that when an external stimulus is introduced to the sequential composition $(a \,; b)$, then either the cascaded stimulus must be generated by the behaviour a, or the behaviour b must be the idle agent behaviour 1. It allows distributivity of $\circ$ over $\,;\,$ to be applied indiscriminately and ensures consistency between the next behaviour and next stimulus mappings with respect to the sequential composition of agent behaviours. Finally, Axiom (iii) describes the interaction of the next stimulus mapping λ with the sequential composition operator $\odot$ for external stimuli.

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.

Proposition 3 provides some consequences of the axiomatisation of C^{2}KA. In [8], an idempotent semiring is called a *quantale* if the natural order induces a complete lattice and multiplication distributes over arbitrary suprema.

Proposition 3. *Let $\big(\mathcal{S},\mathcal{K}\big)$ be a C^{2}KA where the underlying CKA and stimulus structure are built from quantales. For all $a, b \in K$ and $s, t \in S$:*

(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) $a \leq_{\mathcal{K}} b \ \wedge\ s \leq_{\mathcal{S}} t \implies s \circ a \leq_{\mathcal{K}} t \circ b$

(iv) $s \circ (a \,;b + b\,; a) \leq_{\mathcal{K}} s \circ (a * b)$

(v) $s \circ a^{\circledcirc} \leq_{\mathcal{K}} s \circ a^{\circledast}$

(vi) $s \circ a^{\circledcirc} = \ +(n \mid n \geq 0 \ :\ s \circ a^n\,)$

(vii) $s \leq_{\mathcal{S}} t \implies \lambda(s,a) \leq_{\mathcal{S}} \lambda(t,a)$

(viii) $a \leq_{\mathcal{K}} b \implies \lambda(s,a) \leq_{\mathcal{S}} \lambda(s,b)$

(ix) $a \leq_{\mathcal{K}} b \ \wedge\ s \leq_{\mathcal{S}} t \implies \lambda(s,a) \leq_{\mathcal{S}} \lambda(t,b)$

(x) $\lambda(s,(a\,;b + b\,;a)) \leq_{\mathcal{S}} \lambda(s,(a * b))$

(xi) $\lambda(s,a^{\circledcirc}) \leq_{\mathcal{S}} \lambda(s,a^{\circledast})$

(xii) $\lambda(s,a^{\circledcirc}) = \ \oplus(n \mid n \geq 0 \ :\ \lambda(s,a^n)\,)$

Proof. The detailed proofs can be found in Appendix A.2. □

In Proposition 3, Identity (i) (resp. (vii)) shows that $\circ$ (resp. λ) is isotone with respect to $\leq_{\mathcal{K}}$ (resp. $\leq_{\mathcal{S}}$). Identities (ii), (iii), (viii), and (ix) relate sub-stimuli and sub-behaviours with respect to the next behaviour mapping $\circ$ and the next stimulus mapping λ. Lastly, identities (iv)–(vi) (resp. (x)–(xii)) relate the influence of an external stimulus (resp. the external stimuli generated by) on the interleaving and parallel composition of agent behaviours and the sequential iteration and parallel iteration of agent behaviours.

4.4 C^{2}KA 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 C^{2}KA, 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.

Since a C^{2}KA consists of two semimodules $\big({}_{\mathcal{S}}K,+\big)$ and $\big(S_{\mathcal{K}},\oplus\big)$ 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 [17].

Definition 6. *Let $\left({}_{\mathcal{S}}K, +\right)$ be the unitary and zero-preserving left $\mathcal{S}$-semimodule of a C^{2}KA 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) The 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 [19]. The strong orbits can also be viewed as the strongly connected components of a directed graph [30]. 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 $\left({}_{\mathcal{S}}K, +\right)$ 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.

4.4.1 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.

Proposition 4 provides a selection of properties with respect to the orbits and the encompassing relation for agent behaviours.

Proposition 4. *Let $(\mathcal{S}, \mathcal{K})$ be a C^{2}KA. For all $a, b, c \in K$:*

(i) $a \leq_{\mathcal{K}} b \implies \mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(b)$

(ii) $\mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(a + b)$

(iii) $\mathrm{Orb}((a * b) \,;(c * d)) \lessdot_{\mathcal{K}} \mathrm{Orb}((a \,; c) * (b \,; d))$

(iv) $\mathrm{Orb}(a\,;b) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * b)$

(v) $\mathrm{Orb}(a\,;b + b\,;a) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * b)$

(vi) $\mathrm{Orb}((a * b)\,;c) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * (b\,;c))$

(vii) $\mathrm{Orb}(a\,;(b * c)) \lessdot_{\mathcal{K}} \mathrm{Orb}((a\,;b) * c)$

(viii) $\mathrm{Orb}(a^{\odot}) \lessdot_{\mathcal{K}} \mathrm{Orb}(a^{\circledast})$

(ix) $\mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(c) \;\wedge\; \mathrm{Orb}(b) \lessdot_{\mathcal{K}} \mathrm{Orb}(c) \iff \mathrm{Orb}(a) \,\cup\, \mathrm{Orb}(b) \lessdot_{\mathcal{K}} \mathrm{Orb}(c)$

Proof. The detailed proofs can be found in Appendix A.3. □

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 $\neg(c \leq_{\mathcal{K}} a)$.

4.4.2 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.

4.4.3 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 $t \odot s \in \mathrm{Stab}(b)$).

4.4.4 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 the behaviour of an agent behaving as a. By straightforward calculation and the definition of the encompassing relation $\lessdot_{\mathcal{S}}$ for external stimuli, we have that $\mathrm{Stab}(a) \cap \mathrm{Stab}(b) \lessdot_{\mathcal{S}} \mathrm{Stab}(a+b)$ for $a, b \in K$ (see Appendix A.4 for the detailed proof). 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 $\neg\big(\mathrm{Stab}(a+b) \lessdot_{\mathcal{S}} \big(\mathrm{Stab}(a) \cap \mathrm{Stab}(b)\big)\big)$ and $\neg\big(\mathrm{Stab}(a+b) \lessdot_{\mathcal{S}} \big(\mathrm{Stab}(a) \cup \mathrm{Stab}(b)\big)\big)$.

4.4.5 Fixed Points.

An agent behaviour is a fixed point if it is not influenced by any external stimuli other than the deactivation stimulus $\mathfrak{d}$.

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

Proposition 5. *Let $\big(\mathcal{S}, \mathcal{K}\big)$ be a C^{2}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) $a^{\odot}$ is a fixed point if additionally $\big(\mathcal{S}, \mathcal{K}\big)$ is without reactivation

Proof. The proofs each use Definition 6(iv). The proof for (i) is straightforward from the axiomatisation of C^{2}KA. The proof for (ii) involves Definition 3(i) for $\big({}_{\mathcal{S}}K, +\big)$ and the proof for (iii) uses Definition 5(i). The proof for (iv) uses Proposition 3(vi), the application of (iii), and the definition of $a^{\odot}$. The detailed proofs are given in Appendix A.5. $\square$

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 behaviours results in a fixed 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 C^{2}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) $a \sim_{\mathcal{K}} b \implies (a \lhd c \iff b \lhd c)$

Proof. The proof for (i) follows straightforwardly from Definition 8 and the definition of the orbit of a fixed point. The proof for (ii) is straightforward from Definition 8 and the definition of $\sim_{\mathcal{K}}$. The proof for (iii) involves the shunting rule, the definition of $\sim_{\mathcal{K}}$, and Definition 8. The detailed proofs can be found in Appendix A.6. □

Proposition 6(i) states that if an agent has a fixed 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 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 Specifying Systems of Communicating Agents

In order to specify a system of communicating agents using C^{2}KA, we have three levels of specification. To aid in explaining each of these levels of specification, we utilise a simple illustrative example of a one-place buffer adapted from [23].

5.1 Scenario

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. We consider the following set of events which are simple assignments to the buffer status flags:

$$\begin{array}{lclclcl} P_1 & \overset{\text{def}}{=} & \mathit{flag}_1 := \mathit{off} & \qquad & Q_1 & \overset{\text{def}}{=} & \mathit{flag}_2 := \mathit{off} \\ P_2 & \overset{\text{def}}{=} & \mathit{flag}_1 := \mathit{on} & \qquad & Q_2 & \overset{\text{def}}{=} & \mathit{flag}_2 := \mathit{on} \end{array}$$

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

Furthermore, suppose that the behaviour of the 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 $\{\mathit{in},\ \mathit{out},\ \mathit{error},\ \mathfrak{d},\ \mathfrak{n}\}$. Note that in this example, the external stimuli are related to the external ports of the buffer as described in [23], in the sense that the external stimuli are introduced at the external ports of the buffer.

5.2 Stimulus-Response Specification of Agents

At the stimulus-response specification of agents level, we give the specification of the next behaviour mapping $\circ$ and the next stimulus mapping λ for each agent in the system. This involves providing a specification of the form $\big(a, \circ_{\mathbf{A}}, \lambda_{\mathbf{A}}\big)$ for each agent $\mathbf{A}$, where $a \in K$ is a CKA term describing the behaviour of agent $\mathbf{A}$ and where the mappings $\circ_{\mathbf{A}}$ and $\lambda_{\mathbf{A}}$ define the stimulus-response specification for agent $\mathbf{A}$ in terms of the next behaviours and next stimuli, respectively.

In order to specify the one-place buffer using C^{2}KA, assume that there are two basic system agents, $\mathbf{P}$ and $\mathbf{Q}$, which are responsible for controlling the buffer state flags flag_1 and flag_2, respectively. Also assume that we have a C^{2}KA without reactivation (i.e., $s \circ 1 = 1$ for all $s \in S \backslash \{\mathfrak{d}\}$). In this way, the agent behaviours can be compactly specified as the following CKA terms:

$$\mathbf{P} \overset{\text{def}}{=} P_1 + P_2 \qquad\qquad \mathbf{Q} \overset{\text{def}}{=} Q_1 + Q_2$$

This means that agent $\mathbf{P}$ can behave as either P_1 or P_2 and similarly, agent $\mathbf{Q}$ can behave as either Q_1 or Q_2.

Next, we specify the next behaviour mapping $\circ$ and the next stimulus mapping λ for agents $\mathbf{P}$ and $\mathbf{Q}$. In other words, we articulate how each agent responds to each external stimulus. For our example of the one-place buffer, the stimulus-response specification for agents $\mathbf{P}$ and $\mathbf{Q}$ are given in Table 1 and Table 2, respectively. In this example, it is easy to see that the behaviour Q_2 is a fixed point. Furthermore, it is clear that agent $\mathbf{Q}$ is not a stimulus initiator since $\lambda(\mathfrak{n}, \mathbf{Q}) = \mathfrak{n}$.

Table 1: Stimulus-Response Specification for Agent $\mathbf{P}$

$\circ_{\mathbf{P}}$	$\mathfrak{n}$	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	$\mathfrak{n}$	$error$	$\mathfrak{n}$	$\mathfrak{n}$

Table 2: Stimulus-Response Specification for Agent $\mathbf{Q}$

$\circ_{\mathbf{Q}}$	$\mathfrak{n}$	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	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$

By composing the behaviours of $\mathbf{P}$ and $\mathbf{Q}$, we are able to obtain the complete behaviour of the one-place buffer. The behaviour of the one-place buffer is given by the following CKA term:

$$\mathbf{Buffer} \overset{\text{def}}{=} \mathbf{P} \,;\, \mathbf{Q} = (P_1 + P_2) \,;\, (Q_1 + Q_2)$$

The **Buffer** agent has four possible behaviours as a result of the composition of agents $\mathbf{P}$ and $\mathbf{Q}$.

$$\begin{array}{llll} \text{EMPTY} \overset{\text{def}}{=} P_1 \,;\, Q_1 & \qquad \text{UNDERFLOW} \overset{\text{def}}{=} P_1 \,;\, Q_2 \\ \text{FULL} \overset{\text{def}}{=} P_2 \,;\, Q_1 & \qquad \text{OVERFLOW} \overset{\text{def}}{=} P_2 \,;\, Q_2 \end{array}$$

This is to say that the one-place buffer may behave as if it is empty, as if it is full, or as an underflow error or an overflow error. For example, the behaviour $P_1 ; Q_1$ denotes the sequential composition of the assignments to $flag_1$ and $flag_2$ to indicate that the buffer is empty and in a non-error state which, at the state-level, represents that the buffer is in an empty state. Similarly, the behaviour $P_2 ; Q_2$ denotes that the buffer is an overflow error state.

We compute the stimulus-response specification for the **Buffer** agent as shown in Table 3.

Table 3: Stimulus-Response Specification for **Buffer**

$\circ_{\mathbf{Buffer}}$	$\mathfrak{n}$	*in*	*out*	*error*
EMPTY	EMPTY	FULL	UNDERFLOW	EMPTY
FULL	FULL	OVERFLOW	EMPTY	FULL
UNDERFLOW	UNDERFLOW	OVERFLOW	UNDERFLOW	UNDERFLOW
OVERFLOW	OVERFLOW	OVERFLOW	UNDERFLOW	OVERFLOW

$\lambda_{\mathbf{Buffer}}$	$\mathfrak{n}$	*in*	*out*	*error*
EMPTY	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
FULL	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
UNDERFLOW	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
OVERFLOW	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$

Below, we give an example computation showing the behaviour of the **Buffer** agent in response to the external stimulus *in*.

$$\begin{aligned}
& in \circ \mathbf{Buffer} \\
= & \quad \langle \text{ Definition of } \mathbf{Buffer} \rangle \\
& in \circ \big((P_1 + P_2) ; (Q_1 + Q_2)\big) \\
= & \quad \langle \text{ Definition 5(i) } \rangle \\
& \big(in \circ (P_1 + P_2)\big) ; \big(\lambda\big(in, (P_1 + P_2)\big) \circ (Q_1 + Q_2)\big) \\
= & \quad \langle \text{ Definition 3(i) for } \big({}_{\mathcal{S}}K, +\big) \rangle \\
& \big(in \circ P_1 + in \circ P_2\big) ; \big(\lambda\big(in, (P_1 + P_2)\big) \circ (Q_1 + Q_2)\big) \\
= & \quad \langle \text{ Definition 3(ii) for } \big(S_{\mathcal{K}}, \oplus\big) \rangle \\
& \big(in \circ P_1 + in \circ P_2\big) ; \big(\big(\lambda\big(in, P_1\big) \oplus \lambda\big(in, P_2\big)\big) \circ (Q_1 + Q_2)\big) \\
= & \quad \langle \text{ Definition 3(ii) for } \big({}_{\mathcal{S}}K, +\big) \rangle \\
& \big(in \circ P_1 + in \circ P_2\big) ; \big(\lambda\big(in, P_1\big) \circ (Q_1 + Q_2) + \lambda\big(in, P_2\big) \circ (Q_1 + Q_2)\big) \\
= & \quad \langle \text{ Definition 3(i) for } \big({}_{\mathcal{S}}K, +\big) \rangle \\
& \big(in \circ P_1 + in \circ P_2\big) ; \big(\lambda\big(in, P_1\big) \circ Q_1 + \lambda\big(in, P_1\big) \circ Q_2 + \lambda\big(in, P_2\big) \circ Q_1 + \lambda\big(in, P_2\big) \circ Q_2\big) \\
= & \quad \langle \text{ Distributivity of } ; \text{ over } + \rangle \\
& \big(in \circ P_1 + in \circ P_2\big) ; \big(\lambda\big(in, P_1\big) \circ Q_1\big) + \big(in \circ P_1 + in \circ P_2\big) ; \big(\lambda\big(in, P_1\big) \circ Q_2\big) + \\
& \big(in \circ P_1 + in \circ P_2\big) ; \big(\lambda\big(in, P_2\big) \circ Q_1\big) + \big(in \circ P_1 + in \circ P_2\big) ; \big(\lambda\big(in, P_2\big) \circ Q_2\big) \\
= & \quad \langle \text{ Distributivity of } ; \text{ over } + \rangle
\end{aligned}$$

$$
\begin{aligned}
& in \circ P_1 \,;\, \big(\lambda\big(in, P_1\big) \circ Q_1\big) + in \circ P_2 \,;\, \big(\lambda\big(in, P_1\big) \circ Q_1\big) + \\
& in \circ P_1 \,;\, \big(\lambda\big(in, P_1\big) \circ Q_2\big) + in \circ P_2 \,;\, \big(\lambda\big(in, P_1\big) \circ Q_2\big) + \\
& in \circ P_1 \,;\, \big(\lambda\big(in, P_2\big) \circ Q_1\big) + in \circ P_2 \,;\, \big(\lambda\big(in, P_2\big) \circ Q_1\big) + \\
& in \circ P_1 \,;\, \big(\lambda\big(in, P_2\big) \circ Q_2\big) + in \circ P_2 \,;\, \big(\lambda\big(in, P_2\big) \circ Q_2\big) \\
= \quad & \langle \text{ Definition 5(ii)} \quad \& \quad \text{Identity of } + \rangle \\
& in \circ P_1 \,;\, \big(\lambda\big(in, P_1\big) \circ Q_1\big) + in \circ P_1 \,;\, \big(\lambda\big(in, P_1\big) \circ Q_2\big) + \\
& in \circ P_2 \,;\, \big(\lambda\big(in, P_2\big) \circ Q_1\big) + in \circ P_2 \,;\, \big(\lambda\big(in, P_2\big) \circ Q_2\big) \\
= \quad & \langle \text{ Definition of } \circ_{\mathbf{P}} \quad \& \quad \text{Definition of } \lambda_{\mathbf{P}} \rangle \\
& P_2 \,;\, (\mathfrak{n} \circ Q_1) + P_2 \,;\, (\mathfrak{n} \circ Q_2) + P_2 \,;\, (error \circ Q_1) + P_2 \,;\, (error \circ Q_2) \\
= \quad & \langle \text{ Definition of } \circ_{\mathbf{Q}} \rangle \\
& P_2 \,;\, Q_1 + P_2 \,;\, Q_2 + P_2 \,;\, Q_2 + P_2 \,;\, Q_2 \\
= \quad & \langle \text{ Idempotence of } + \rangle \\
& P_2 \,;\, Q_1 + P_2 \,;\, Q_2 \\
= \quad & \langle \text{ Definition of the Behaviours of } \textbf{Buffer} \rangle \\
& \textsc{full} + \textsc{overflow}
\end{aligned}
$$

From this example computation, we see that when the **Buffer** agent is subjected to the the external stimulus *in*, it responds by behaving as a full buffer or as an overflow error which can be seen by the column corresponding to the external stimulus *in* in Table 3 for $\circ_{\mathbf{Buffer}}$. Furthermore, with regard to the specification of the **Buffer** agent, we can compute the orbits of each of the buffer behaviours. For instance, Orb(EMPTY) = {EMPTY, FULL, UNDERFLOW, OVERFLOW}. It is plain to see, for example, that the behaviour UNDERFLOW is induced by the behaviour EMPTY via the external stimulus *out* and the behaviour OVERFLOW is induced by the behaviour EMPTY via the external stimulus $in \odot in$. In the specification of the **Buffer** agent, we can also see that we have two strong orbits, namely, those given by {EMPTY, FULL} and {UNDERFLOW, OVERFLOW} which represent the behaviours from agents **P** and **Q**, respectively. This is to say that we have (EMPTY $\sim_{\mathcal{K}}$ FULL) and (UNDERFLOW $\sim_{\mathcal{K}}$ OVERFLOW). Finally, we can compute the stabilisers of each of the buffer behaviours from the specification of the **Buffer** agent. For example, Stab(EMPTY) is generated by $\{error, in \odot out\}$.

5.3 Abstract Behaviour Specification

The second level of specification gives a specification of the abstract behaviour of each system agent along with the external stimuli that it needs to respond to. Whereas at the stimulus-response specification level, we specified all possible responses to all external stimuli for each agent, at the abstract behaviour specification level, we restrict 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.

For the purpose of this example, consider a context in which we only consider the **Buffer** agent as behaving either as an empty buffer or as a full buffer. Furthermore, assume that the behaviour of the **Buffer** agent 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. Continuing with our illustrative example with this context, we specify the abstract behaviour of the one-place buffer as follows:

$$
\begin{array}{ll}
 & \textbf{Buffer} \\
= & \qquad \langle \text{ Definition of } \textbf{Buffer} \text{ in the given context } \rangle \\
 & (\mathit{in} \oplus \mathit{out}) \circ (\textsc{empty} + \textsc{full}) \\
= & \qquad \langle \text{ Definition 3(i) for } \big({}_{\mathcal{S}}K, +\big) \text{ } \rangle \\
 & (\mathit{in} \oplus \mathit{out}) \circ \textsc{empty} + (\mathit{in} \oplus \mathit{out}) \circ \textsc{full} \\
= & \qquad \langle \text{ Definition 3(ii) for } \big({}_{\mathcal{S}}K, +\big) \text{ } \rangle \\
 & \mathit{in} \circ \textsc{empty} + \mathit{out} \circ \textsc{empty} + \mathit{in} \circ \textsc{full} + \mathit{out} \circ \textsc{full} \\
= & \qquad \langle \text{ Definition of } \circ_{\textbf{Buffer}} \text{ } \rangle \\
 & \textsc{full} + \textsc{underflow} + \textsc{overflow} + \textsc{empty}
\end{array}
$$

It is important to note that the specification of the **Buffer** agent at the abstract behaviour specification level is a subset of the specification of the next behaviour and next stimulus mappings for the **Buffer** agent given at the stimulus-response specification level. Additionally, an interesting observation can be made at this level of specification, namely that even though the specification in the given context makes no mention of the **Buffer** agent's error behaviour, through the mathematics of C^{2}KA, we find that the **Buffer** agent does indeed have behaviours for its underflow and overflow error states resulting from the influence of the external stimuli that can be introduced into the system.

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 $\big({}_{\mathcal{S}}K, +\big)$ and the right $\mathcal{K}$-semimodule $\big(S_{\mathcal{K}}, \oplus\big)$ allow us to specify how the external stimuli influence the behaviour of each agent in a given system. For this reason, this level is specification is best suited for describing message passing communication where agents transfer information explicitly through the exchange of data structures, either synchronously or asynchronously.

5.4 Concrete Behaviour Specification

The last level of specification involves providing the state-level specification of each agent behaviour. The state-level behaviours of agents are represented as programs which are defined over a set of events and that can be executed by the system (i.e., the set of basic behaviours $\{P_1, P_2, Q_1, Q_2, 0, 1\}$). 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 **Buffer** agent.

$$
\begin{array}{lcl}
\textsc{empty} & \overset{\text{def}}{=} & \mathit{flag}_1 := \mathit{off} \quad ; \quad \mathit{flag}_2 := \mathit{off} \\
\textsc{full} & \overset{\text{def}}{=} & \mathit{flag}_1 := \mathit{on} \quad ; \quad \mathit{flag}_2 := \mathit{off} \\
\textsc{underflow} & \overset{\text{def}}{=} & \mathit{flag}_1 := \mathit{off} \quad ; \quad \mathit{flag}_2 := \mathit{on} \\
\textsc{overflow} & \overset{\text{def}}{=} & \mathit{flag}_1 := \mathit{on} \quad ; \quad \mathit{flag}_2 := \mathit{on}
\end{array}
$$

When considering the concrete behaviour specification level, C^{2}KA can be viewed as a state-based model of communication. 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 is specification is best suited for shared-variable communication where agents transfer information through a shared medium such as variables, memory locations, etc.

C^{2}KA provides a hybrid mathematical framework which is able to capture both the influence of external stimuli on agent behaviour as well the communication and concurrency of agents at the abstract algebraic level in systems of communicating agents. 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.

6 Related Work and Discussion

Existing state-based and event-based formalisms for communication and concurrency such as LTL [27], CTL [2], CTL* [4], labelled transition systems [16], Petri nets [26], process calculi (e.g., CCS [22], CSP [6], ACP [1], and π-calculus [24]), Hoare traces [7], Mazurkiewicz traces [20], synchronisation trees [22], pomsets [28], and event structures [32] 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^{2}KA offers a powerful 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^{2}KA with a trivial stimulus structure (i.e., $S = \{\mathfrak{n}\}$), then the next behaviour and next stimulus mappings are trivial and the C^{2}KA reduces to a CKA.

Furthermore, C^{2}KA supports the ability to work in either a state-based or event-based model 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^{2}KA. With these levels of specification, C^{2}KA 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 [8, 9, 10, 11] 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, $\mathrm{C^2KA}$ 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, $\mathrm{C^2KA}$ inherits the algebraic foundation of CKA with all of its models and theory.

7 Conclusion and Future Work

In this paper, we proposed a mathematical framework for communication and concurrency called Communicating Concurrent Kleene Algebra ($\mathrm{C^2KA}$). $\mathrm{C^2KA}$ 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. $\mathrm{C^2KA}$ 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 $\mathrm{C^2KA}$ 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, $\mathrm{C^2KA}$ 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. We also aim to examine the ability to adapt $\mathrm{C^2KA}$ for use in solving interface equations (e.g., [29]) 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 $\mathrm{C^2KA}$ to capture and explain the influence of external stimuli on agent behaviour in social networking environments.

References

[1] J. Bergstra and J. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1-3):109–137, 1984.

[2] E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In D. Kozen, editor, *Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer Berlin Heidelberg, 1982.

[3] R. Cleaveland and S. Smolka. Strategic directions in concurrency research. *ACM Computing Surveys*, 28(4):607–625, December 1996.

[4] E. Emerson and J. Halpern. “sometimes” and “not never” revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[5] U. Hebisch and H. Weinert. *Semirings: Algebraic Theory and Applications in Computer Science*, volume 5 of *Series in Algebra.* World Scientific, 1993.

[6] C. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, August 1978.

[7] C. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[8] C. Hoare, B. Möller, G. Struth, and I. Wehrman. Concurrent Kleene algebra. In M. Bravetti and G. Zavattaro, editors, *CONCUR 2009 - Concurrency Theory*, volume 5710 of *Lecture Notes in Computer Science*, pages 399–414. Springer Berlin / Heidelberg, 2009.

[9] C. Hoare, B. Möller, G. Struth, and I. Wehrman. Foundations of concurrent Kleene algebra. In R. Berghammer, A. Jaoua, and B. Möller, editors, *Relations and Kleene Algebra in Computer Science*, volume 5827 of *Lecture Notes in Computer Science*, pages 166–186. Springer Berlin / Heidelberg, 2009.

[10] C. Hoare, B. Möller, G. Struth, and I. Wehrman. Concurrent Kleene algebra and its foundations. Technical Report CS-10-04, University of Sheffield, Department of Computer Science, Sheffield, UK, August 2010. Available: http://www.dcs.shef.ac.uk/~georg/ka/.

[11] C. Hoare, B. Möller, G. Struth, and I. Wehrman. Concurrent Kleene algebra and its foundations. *Journal of Logic and Algebraic Programming*, 80(6):266 – 296, 2011.

[12] C. Hoare and J. Wickerson. Unifying models fo data flow. In M. Broy, C. Leuxner, and C. Hoare, editors, *Proceedings of the 2010 Marktoberdorf Summer School on Software and Systems Safety*, pages 211 – 230. IOS Press, August 2011.

[13] P. Höfner, R. Khedri, and B. Möller. Supplementing product families with behaviour. *International Journal of Software and Informatics*, 5(1-2):245–266, 2011.

[14] W. Holcombe. *Algebraic Automata Theory.* Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2004.

[15] J. Jaskolka, R. Khedri, and Q. Zhang. On the necessary conditions for covert channel existence: A state-of-the-art survey. *Procedia Computer Science*, 10:458 – 465, August 2012. Proceedings of the 3rd International Conference on Ambient Systems, Networks and Technologies, ANT 2012.

[16] R. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, July 1976.

[17] M. Kilp, U. Knauer, and A. Mikhalev. *Monoids, Acts And Categories With Applications to Wreath Products and Graphs: A Handbook for Students and Researchers*, volume 29 of *De Gruyter Expositions in Mathematics Series.* Walter de Gruyter, 2000.

[18] D. Kozen. *Automata and Computability.* Undergraduate Texts in Computer Science. Springer, 1997.

[19] S. Linton, G. Pfeiffer, E. Robertson, and N. Ruškuc. Computing transformation semigroups. *Journal of Symbolic Computation*, 33(2):145 – 162, 2002.

[20] A. Mazurkiewicz. Trace theory. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *Lecture Notes in Computer Science*, pages 279–324. Springer Berlin / Heidelberg, 1987.

[21] W. McCune. Prover9 and mace4. Available: `http://www.cs.unm.edu/~mccune/prover9/` (Accessed: November 7, 2013), 2005.

[22] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer-Verlag, 1980.

[23] R. Milner. *Communication and Concurrency.* Prentice-Hall International Series in Computer Science. Prentice Hall, 1989.

[24] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes part I. *Information and Computation*, 100(1):1–40, September 1992.

[25] B. Möller and G. Struth. wp is wlp. In W. MacCaull, M. Winter, and I. Düntsch, editors, *Relational Methods in Computer Science*, volume 3929 of *Lecture Notes in Computer Science*, pages 200–211. Springer Berlin Heidelberg, 2006.

[26] C. Petri. *Kommunikation mit Automaten.* PhD thesis, Institut für instrumentelle Mathematik, Bonn, Germany, 1962. English translation available as: *Communication with Automata*, Technical Report RADC-TR-65-377, volume 1, supplement 1, Applied Data Research, Princeton, NJ, 1966.

[27] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[28] V. Pratt. Modeling concurrency with partial orders. *International Journal of Parallel Programming*, 15(1):33–71, February 1986.

[29] M. Shields. Implicit system specification and the interface equation. *The Computer Journal*, 32(5):399 – 412, October 1989.

[30] B. Steinberg. A theory of transformation monoids: Combinatorics and representation theory. *The Electronic Journal of Combinatorics*, 17(1), 2010.

[31] J. Watson. *Behaviorism.* University of Chicago Press, 1930.

[32] G. Winskel. Event structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer Berlin / Heidelberg, 1987.

A Detailed Proofs of Propositions

A.1 Detailed Proof of Proposition 2

For all $a \in K$, $a^{\odot} \leq a^{\circledast}$.

$$\begin{array}{lll}
 & a^{\odot} \leq a^{\circledast} & \\
\Longleftrightarrow & & \langle \text{ Right Identity of } ; \rangle \\
 & a^{\odot} ; 1 \leq a^{\circledast} & \\
\Longleftarrow & & \langle \text{ Definition 1(iii) for } ; \rangle \\
 & 1 + a ; a^{\circledast} \leq a^{\circledast} & \\
\Longleftrightarrow & & \langle \text{ Definition 1(i) for } ^{\circledast} \rangle \\
 & 1 + a ; a^{\circledast} \leq 1 + a * a^{\circledast} & \\
\Longleftrightarrow & & \langle \text{ Proposition 1(iii) } \rangle \\
 & \mathsf{true} &
\end{array}$$

A.2 Detailed Proof of Proposition 3

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

(i) $a \leq_{\mathcal{K}} b \implies s \circ a \leq_{\mathcal{K}} s \circ b$

$$\begin{array}{lll}
 & s \circ a \leq_{\mathcal{K}} s \circ b & \\
\Longleftrightarrow & & \langle \text{ Definition of } \leq_{\mathcal{K}} \rangle \\
 & s \circ a + s \circ b = s \circ b & \\
\Longleftrightarrow & & \langle \text{ Definition 3(i) for } ({}_{\mathcal{S}}K, +) \rangle \\
 & s \circ (a + b) = s \circ b & \\
\Longleftarrow & & \langle \text{ Hypothesis: } a \leq_{\mathcal{K}} b \rangle \\
 & s \circ b = s \circ b & \\
\Longleftrightarrow & & \langle \text{ Reflexivity of } = \rangle \\
 & \mathsf{true} &
\end{array}$$

(ii) $s \leq_{\mathcal{S}} t \implies s \circ a \leq_{\mathcal{K}} t \circ a$

$$\begin{array}{lll}
 & s \circ a \leq_{\mathcal{K}} t \circ a & \\
\Longleftrightarrow & & \langle \text{ Definition of } \leq_{\mathcal{K}} \rangle \\
 & s \circ a + t \circ a = t \circ a & \\
\Longleftrightarrow & & \langle \text{ Definition 3(ii) for } ({}_{\mathcal{S}}K, +) \rangle \\
 & (s \oplus t) \circ a = t \circ a & \\
\Longleftarrow & & \langle \text{ Hypothesis: } s \leq_{\mathcal{S}} t \rangle \\
 & t \circ a = t \circ a & \\
\Longleftrightarrow & & \langle \text{ Reflexivity of } = \rangle \\
 & \mathsf{true} &
\end{array}$$

(iii) $a \leq_{\mathcal{K}} b \ \wedge \ s \leq_{\mathcal{S}} t \implies s \circ a \leq_{\mathcal{K}} t \circ b$

$$\begin{array}{ll}
& s \circ a \leq_{\mathcal{K}} t \circ b \\
\iff & \langle \text{ Definition of } \leq_{\mathcal{K}} \rangle \\
& s \circ a + t \circ b = t \circ b \\
\Longleftarrow & \langle \text{ Hypothesis: } s \leq_{\mathcal{S}} t \rangle \\
& s \circ a + (s \oplus t) \circ b = t \circ b \\
\iff & \langle \text{ Definition 3(ii) for } \big({}_{\mathcal{S}}K, +\big) \rangle \\
& s \circ a + s \circ b + t \circ b = t \circ b \\
\iff & \langle \text{ Definition 3(i) for } \big({}_{\mathcal{S}}K, +\big) \rangle \\
& s \circ (a + b) + t \circ b = t \circ b \\
\Longleftarrow & \langle \text{ Hypothesis: } a \leq_{\mathcal{K}} b \rangle \\
& s \circ b + t \circ b = t \circ b \\
\iff & \langle \text{ Definition 3(ii) for } \big({}_{\mathcal{S}}K, +\big) \rangle \\
& (s \oplus t) \circ b = t \circ b \\
\Longleftarrow & \langle \text{ Hypothesis: } s \leq_{\mathcal{S}} t \rangle \\
& t \circ b = t \circ b \\
\iff & \langle \text{ Reflexivity of } = \rangle \\
& \mathsf{true}
\end{array}$$

(iv) $s \circ (a \,;b + b\,;a) \leq_{\mathcal{K}} s \circ (a * b)$

$$\begin{array}{ll}
& s \circ (a \,;b + b\,;a) \leq_{\mathcal{K}} s \circ (a * b) \\
\Longleftarrow & \langle \text{ Proposition 3(i) } \rangle \\
& a\,;b + b\,;a \leq_{\mathcal{K}} a * b \\
\iff & \langle \text{ Proposition 1(i) } \quad \& \quad \text{Proposition 1(iii)} \quad \& \quad \text{Idempotence of } + \rangle \\
& \mathsf{true}
\end{array}$$

(v) $s \circ a^{\textcircled{;}} \leq_{\mathcal{K}} s \circ a^{\circledast}$

$$\begin{array}{ll}
& s \circ a^{\textcircled{;}} \leq_{\mathcal{K}} s \circ a^{\circledast} \\
\Longleftarrow & \langle \text{ Proposition 3(i) } \rangle \\
& a^{\textcircled{;}} \leq_{\mathcal{K}} a^{\circledast} \\
\iff & \langle \text{ Proposition 2 } \rangle \\
& \mathsf{true}
\end{array}$$

(vi) $s \circ a^{\textcircled{;}} = +(n \mid n \geq 0 : s \circ a^n)$ Recall the definition of $a^{\textcircled{;}}$:

$$a^{\textcircled{;}} = +(n \mid n \geq 0 : a^n) \tag{1}$$

where

$$\begin{aligned} a^0 &\overset{\text{def}}{=} 1 \\ a^{n+1} &\overset{\text{def}}{=} a^n \,;\, a \end{aligned}$$

$$\begin{aligned}
& s \circ a^{\textcircled{;}} \\
= \quad & \langle \text{ Definition of } a^{\textcircled{;}}\text{: Equation (1) } \rangle \\
& s \circ +(n \mid n \geq 0 : a^n) \\
= \quad & \langle \text{ Definition 3(i) for } \big({}_{\mathcal{S}}K, +\big) \rangle \\
& +(n \mid n \geq 0 : s \circ a^n)
\end{aligned}$$

(vii) $s \leq_{\mathcal{S}} t \implies \lambda(s,a) \leq_{\mathcal{S}} \lambda(t,a)$

$$\begin{aligned}
& \lambda(s,a) \leq_{\mathcal{S}} \lambda(t,a) \\
\iff \quad & \langle \text{ Definition of } \leq_{\mathcal{S}} \rangle \\
& \lambda(s,a) \oplus \lambda(t,a) = \lambda(t,a) \\
\iff \quad & \langle \text{ Definition 3(i) for } \big(S_{\mathcal{K}}, \oplus\big) \rangle \\
& \lambda\big((s \oplus t), a\big) = \lambda(t,a) \\
\Longleftarrow \quad & \langle \text{ Hypothesis: } s \leq_{\mathcal{S}} t \rangle \\
& \lambda(t,a) = \lambda(t,a) \\
\iff \quad & \langle \text{ Reflexivity of } = \rangle \\
& \mathsf{true}
\end{aligned}$$

(viii) $a \leq_{\mathcal{K}} b \implies \lambda(s,a) \leq_{\mathcal{S}} \lambda(s,b)$

$$\begin{aligned}
& \lambda(s,a) \leq_{\mathcal{S}} \lambda(s,b) \\
\iff \quad & \langle \text{ Definition of } \leq_{\mathcal{S}} \rangle \\
& \lambda(s,a) \oplus \lambda(s,b) = \lambda(s,b) \\
\iff \quad & \langle \text{ Definition 3(ii) for } \big(S_{\mathcal{K}}, \oplus\big) \rangle \\
& \lambda\big(s, (a + b)\big) = \lambda(s,b) \\
\Longleftarrow \quad & \langle \text{ Hypothesis: } a \leq_{\mathcal{K}} b \rangle \\
& \lambda(s,b) = \lambda(s,b) \\
\iff \quad & \langle \text{ Reflexivity of } = \rangle \\
& \mathsf{true}
\end{aligned}$$

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

$$\begin{aligned}
&\lambda(s,a) \leq_{\mathcal{S}} \lambda(t,b) \\
\iff &\qquad \langle \text{ Definition of } \leq_{\mathcal{S}} \rangle \\
&\lambda(s,a) \oplus \lambda(t,b) = \lambda(t,b) \\
\Longleftarrow &\qquad \langle \text{ Hypothesis: } s \leq_{\mathcal{S}} t \rangle \\
&\lambda(s,a) \oplus \lambda\big((s \oplus t), b\big) = \lambda(t,b) \\
\iff &\qquad \langle \text{ Definition 3(i) for } \big(S_{\mathcal{K}}, \oplus\big) \rangle \\
&\lambda(s,a) \oplus \lambda(s,b) \oplus \lambda(t,b) = \lambda(t,b) \\
\iff &\qquad \langle \text{ Definition 3(ii) for } \big(S_{\mathcal{K}}, \oplus\big) \rangle \\
&\lambda\big(s, (a+b)\big) \oplus \lambda(t,b) = t \circ b \\
\Longleftarrow &\qquad \langle \text{ Hypothesis: } a \leq_{\mathcal{K}} b \rangle \\
&\lambda(s,b) \oplus \lambda(t,b) = \lambda(t,b) \\
\iff &\qquad \langle \text{ Definition 3(i) for } \big(S_{\mathcal{K}}, \oplus\big) \rangle \\
&\lambda\big((s \oplus t), b\big) = \lambda(t,b) \\
\Longleftarrow &\qquad \langle \text{ Hypothesis: } s \leq_{\mathcal{S}} t \rangle \\
&\lambda(t,b) = \lambda(t,b) \\
\iff &\qquad \langle \text{ Reflexivity of } = \rangle \\
&\mathsf{true}
\end{aligned}$$

(x) $\lambda(s, (a\,;b + b\,;a)) \leq_{\mathcal{S}} \lambda(s, (a * b))$

$$\begin{aligned}
&\lambda(s, (a\,;b + b\,;a)) \leq_{\mathcal{S}} \lambda(s, (a * b)) \\
\Longleftarrow &\qquad \langle \text{ Proposition 3(viii) } \rangle \\
&a\,;b + b\,;a \leq_{\mathcal{K}} a * b \\
\iff &\qquad \langle \text{ Proposition 1(i) } \quad \& \quad \text{Proposition 1(iii)} \quad \& \quad \text{Idempotence of } + \rangle \\
&\mathsf{true}
\end{aligned}$$

(xi) $\lambda(s, a^{\textcircled{;}}) \leq_{\mathcal{S}} \lambda(s, a^{\circledast})$

$$\begin{aligned}
&\lambda(s, a^{\textcircled{;}}) \leq_{\mathcal{S}} \lambda(s, a^{\circledast}) \\
\Longleftarrow &\qquad \langle \text{ Proposition 3(viii) } \rangle \\
&a^{\textcircled{;}} \leq_{\mathcal{K}} a^{\circledast} \\
\iff &\qquad \langle \text{ Proposition 2 } \rangle \\
&\mathsf{true}
\end{aligned}$$

(xii) $\lambda(s, a^{\odot}) = \oplus(n \mid n \geq 0 : \lambda(s, a^n))$

$$\begin{array}{ll}
& \lambda(s, a^{\odot}) \\
= & \qquad \langle \text{ Definition of } a^{\odot}\text{: Equation (1) } \rangle \\
& \lambda(s,\ +(n \mid n \geq 0 : a^n)) \\
= & \qquad \langle \text{ Definition 3(ii) for } (S_{\mathcal{K}}, \oplus) \rangle \\
& \oplus(n \mid n \geq 0 : s \circ a^n)
\end{array}$$

A.3 Detailed Proof of Proposition 4

Let $(\mathcal{S}, \mathcal{K})$ be a C^{2}KA. For all $a, b, c \in K$:

(i) $a \leq_{\mathcal{K}} b \implies \mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(b)$

$$\begin{array}{ll}
& \mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(b) \\
\iff & \qquad \langle \text{ Definition 7 } \rangle \\
& \forall\big(x \mid x \in \mathrm{Orb}(a) : \exists(y \mid y \in \mathrm{Orb}(b) : x \leq_{\mathcal{K}} y)\big) \\
\iff & \qquad \langle \text{ Trading for } \forall \quad \& \quad \text{Trading for } \exists \rangle \\
& \forall\big(x \mid: x \in \mathrm{Orb}(a) \implies \exists(y \mid: y \in \mathrm{Orb}(b) \wedge x \leq_{\mathcal{K}} y)\big) \\
\iff & \qquad \langle \text{ Set Membership Axiom } \rangle \\
& \forall\big(x \mid: \exists(s \mid s \in S : s \circ a = x) \implies \exists\big(y \mid: \exists(s \mid s \in S : s \circ b = y) \wedge x \leq_{\mathcal{K}} y\big)\big) \\
\iff & \qquad \langle \text{ Distributivity of } \wedge \text{ over } \exists \rangle \\
& \forall\big(x \mid: \exists(s \mid s \in S : s \circ a = x) \implies \exists\big(y \mid: \exists(s \mid s \in S : s \circ b = y \wedge x \leq_{\mathcal{K}} y)\big)\big) \\
\iff & \qquad \langle \text{ Interchange of Dummies } \rangle \\
& \forall\big(x \mid: \exists(s \mid s \in S : s \circ a = x) \implies \exists\big(s \mid s \in S : \exists(y \mid: s \circ b = y \wedge x \leq_{\mathcal{K}} y)\big)\big) \\
\iff & \qquad \langle \text{ Trading for } \exists \rangle \\
& \forall\big(x \mid: \exists(s \mid s \in S : s \circ a = x) \implies \exists\big(s \mid s \in S : \exists(y \mid s \circ b = y : x \leq_{\mathcal{K}} y)\big)\big) \\
\iff & \qquad \langle \text{ One-point Rule } \rangle \\
& \forall\big(x \mid: \exists(s \mid s \in S : s \circ a = x) \implies \exists\big(s \mid s \in S : x \leq_{\mathcal{K}} s \circ b\big)\big) \\
\Longleftarrow & \qquad \langle \text{ Monotonic } \exists\text{-Body } \rangle \\
& \forall\big(x \mid: \forall(s \mid s \in S : s \circ a = x \implies x \leq_{\mathcal{K}} s \circ b)\big) \\
\iff & \qquad \langle \text{ Definition of } \implies \rangle \\
& \forall\big(x \mid: \forall(s \mid s \in S : s \circ a \neq x \vee x \leq_{\mathcal{K}} s \circ b)\big) \\
\iff & \qquad \langle \text{ Definition of } \leq_{\mathcal{K}} \rangle \\
& \forall\big(x \mid: \forall(s \mid s \in S : s \circ a \neq x \vee x + (s \circ b) = s \circ b)\big) \\
\Longleftarrow & \qquad \langle \text{ Hypothesis: } a \leq_{\mathcal{K}} b \quad \& \quad \text{Proposition 3(i) } \rangle \\
& \forall\big(x \mid: \forall(s \mid s \in S : \mathsf{true})\big) \\
\iff & \qquad \langle\ \forall\text{-True Body } \rangle \\
& \mathsf{true}
\end{array}$$

(ii) $\mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(a+b)$

$$\begin{array}{ll}
 & \mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(a+b) \\
\Longleftarrow & \langle \text{ Proposition 4(i) } \rangle \\
 & a \leq_{\mathcal{K}} a+b \\
\Longleftrightarrow & \langle \text{ Definition of } \leq_{\mathcal{K}} \quad \& \quad \text{Idempotence of } + \rangle \\
 & a+b = a+b \\
\Longleftrightarrow & \langle \text{ Reflexivity of } = \rangle \\
 & \mathsf{true}
\end{array}$$

(iii) $\mathrm{Orb}((a*b)\,;(c*d)) \lessdot_{\mathcal{K}} \mathrm{Orb}((a\,;c)*(b\,;d))$

$$\begin{array}{ll}
 & \mathrm{Orb}((a*b)\,;(c*d)) \lessdot_{\mathcal{K}} \mathrm{Orb}((a\,;c)*(b\,;d)) \\
\Longleftarrow & \langle \text{ Proposition 4(i) } \rangle \\
 & (a*b)\,;(c*d) \leq_{\mathcal{K}} (a\,;c)*(b\,;d) \\
\Longleftrightarrow & \langle \text{ Proposition 1(ii) } \rangle \\
 & \mathsf{true}
\end{array}$$

(iv) $\mathrm{Orb}(a\,;b) \lessdot_{\mathcal{K}} \mathrm{Orb}(a*b)$

$$\begin{array}{ll}
 & \mathrm{Orb}(a\,;b) \lessdot_{\mathcal{K}} \mathrm{Orb}(a*b) \\
\Longleftarrow & \langle \text{ Proposition 4(i) } \rangle \\
 & a\,;b \leq_{\mathcal{K}} a*b \\
\Longleftrightarrow & \langle \text{ Proposition 1(iii) } \rangle \\
 & \mathsf{true}
\end{array}$$

(v) $\mathrm{Orb}(a\,;b+b\,;a) \lessdot_{\mathcal{K}} \mathrm{Orb}(a*b)$

$$\begin{array}{ll}
 & \mathrm{Orb}(a\,;b+b\,;a) \lessdot_{\mathcal{K}} \mathrm{Orb}(a*b) \\
\Longleftarrow & \langle \text{ Proposition 4(i) } \rangle \\
 & a\,;b+b\,;a \leq_{\mathcal{K}} a*b \\
\Longleftrightarrow & \langle \text{ Proposition 1(i) } \quad \& \quad \text{Proposition 1(iii)} \quad \& \quad \text{Idempotence of } + \rangle \\
 & \mathsf{true}
\end{array}$$

(vi) $\mathrm{Orb}((a * b)\,;c) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * (b\,;c))$

$$\begin{array}{ll}
& \mathrm{Orb}((a * b)\,;c) \lessdot_{\mathcal{K}} \mathrm{Orb}(a * (b\,;c)) \\
\Longleftarrow & \qquad \langle \text{ Proposition 4(i) } \rangle \\
& (a * b)\,;c \leq_{\mathcal{K}} a * (b\,;c) \\
\Longleftrightarrow & \qquad \langle \text{ Proposition 1(iv) } \rangle \\
& \mathsf{true}
\end{array}$$

(vii) $\mathrm{Orb}(a\,;(b * c)) \lessdot_{\mathcal{K}} \mathrm{Orb}((a\,;b) * c)$

$$\begin{array}{ll}
& \mathrm{Orb}(a\,;(b * c)) \lessdot_{\mathcal{K}} \mathrm{Orb}((a\,;b) * c) \\
\Longleftarrow & \qquad \langle \text{ Proposition 4(i) } \rangle \\
& a\,;(b * c) \leq_{\mathcal{K}} (a\,;b) * c \\
\Longleftrightarrow & \qquad \langle \text{ Proposition 1(v) } \rangle \\
& \mathsf{true}
\end{array}$$

(viii) $\mathrm{Orb}(a^{\textcircled{;}}) \lessdot_{\mathcal{K}} \mathrm{Orb}(a^{\circledast})$

$$\begin{array}{ll}
& \mathrm{Orb}(a^{\textcircled{;}}) \lessdot_{\mathcal{K}} \mathrm{Orb}(a^{\circledast}) \\
\Longleftarrow & \qquad \langle \text{ Proposition 4(i) } \rangle \\
& a^{\textcircled{;}} \leq_{\mathcal{K}} a^{\circledast} \\
\Longleftrightarrow & \qquad \langle \text{ Proposition 2 } \rangle \\
& \mathsf{true}
\end{array}$$

(ix) $\mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(c) \;\wedge\; \mathrm{Orb}(b) \lessdot_{\mathcal{K}} \mathrm{Orb}(c) \iff \mathrm{Orb}(a) \cup \mathrm{Orb}(b) \lessdot_{\mathcal{K}} \mathrm{Orb}(c)$

$$\begin{array}{ll}
& \mathrm{Orb}(a) \cup \mathrm{Orb}(b) \lessdot_{\mathcal{K}} \mathrm{Orb}(c) \\
\Longleftrightarrow & \qquad \langle \text{ Definition 7 } \rangle \\
& \forall\big(x \mid x \in \mathrm{Orb}(a) \cup \mathrm{Orb}(b) : \exists(y \mid y \in \mathrm{Orb}(c) : x \leq_{\mathcal{K}} y)\big) \\
\Longleftrightarrow & \qquad \langle \text{ Set Union Axiom } \rangle \\
& \forall\big(x \mid x \in \mathrm{Orb}(a) \vee x \in \mathrm{Orb}(b) : \exists(y \mid y \in \mathrm{Orb}(c) : x \leq_{\mathcal{K}} y)\big) \\
\Longleftrightarrow & \qquad \langle \text{ Range Split } \rangle \\
& \forall\big(x \mid x \in \mathrm{Orb}(a) : \exists(y \mid y \in \mathrm{Orb}(c) : x \leq_{\mathcal{K}} y)\big) \;\wedge \\
& \forall\big(x \mid x \in \mathrm{Orb}(b) : \exists(y \mid y \in \mathrm{Orb}(c) : x \leq_{\mathcal{K}} y)\big) \\
\Longleftrightarrow & \qquad \langle \text{ Definition 7 } \rangle \\
& \mathrm{Orb}(a) \lessdot_{\mathcal{K}} \mathrm{Orb}(c) \;\wedge\; \mathrm{Orb}(b) \lessdot_{\mathcal{K}} \mathrm{Orb}(c)
\end{array}$$

A.4 Detailed Proof of $\mathrm{Stab}(a) \cap \mathrm{Stab}(b) \lessdot_{\mathcal{S}} \mathrm{Stab}(a+b)$

For all $a, b \in K$, $\mathrm{Stab}(a) \cap \mathrm{Stab}(b) \lessdot_{\mathcal{S}} \mathrm{Stab}(a+b)$.

$$
\begin{array}{ll}
& \mathrm{Stab}(a) \cap \mathrm{Stab}(b) \lessdot_{\mathcal{S}} \mathrm{Stab}(a+b) \\
\Longleftrightarrow & \qquad \langle \text{ Definition 7 for } \lessdot_{\mathcal{S}} \rangle \\
& \forall\big(x \mid x \in \mathrm{Stab}(a) \cap \mathrm{Stab}(b) : \exists(y \mid y \in \mathrm{Stab}(a+b) : x \leq_{\mathcal{S}} y)\big) \\
\Longleftrightarrow & \qquad \langle \text{ Trading for } \exists \rangle \\
& \forall\big(x \mid x \in \mathrm{Stab}(a) \cap \mathrm{Stab}(b) : \exists(y \mid: y \in \mathrm{Stab}(a+b) \wedge x \leq_{\mathcal{K}} y)\big) \\
\Longleftarrow & \qquad \langle \text{ } \exists\text{-Introduction } \rangle \\
& \forall\big(x \mid x \in \mathrm{Stab}(a) \cap \mathrm{Stab}(b) : x \in \mathrm{Stab}(a+b) \wedge x \leq_{\mathcal{K}} x\big) \\
\Longleftrightarrow & \qquad \langle \text{ Trading for } \forall \quad \& \quad \text{Reflexivity of } \leq_{\mathcal{K}} \rangle \\
& \forall\big(x \mid: x \in \mathrm{Stab}(a) \cap \mathrm{Stab}(b) \implies x \in \mathrm{Stab}(a+b)\big) \\
\Longleftrightarrow & \qquad \langle \text{ Definition 6(iii) } \rangle \\
& \forall\big(x \mid: x \circ a = a \wedge x \circ b = b \implies x \circ (a+b) = a+b\big) \\
\Longleftrightarrow & \qquad \langle \text{ Definition 3(i) for } \big({}_{\mathcal{S}}K, +\big) \rangle \\
& \forall\big(x \mid: \mathsf{true}\big) \\
\Longleftrightarrow & \qquad \langle \text{ } \forall\text{-True Body } \rangle \\
& \mathsf{true}
\end{array}
$$

A.5 Detailed Proof of Proposition 5

Let $(\mathcal{S}, \mathcal{K})$ be a C^{2}KA and let $a, b \in K$.

(i) 0 is a fixed point w.r.t. $\circ$.

$$
\begin{array}{ll}
& 0 \text{ is a fixed point} \\
\Longleftrightarrow & \qquad \langle \text{ Definition 6(iv) } \rangle \\
& \forall(s \mid s \in S : s \circ 0 = 0) \\
\Longleftrightarrow & \qquad \langle \text{ } \big({}_{\mathcal{S}}K, +\big) \text{ is zero-preserving } (\mathfrak{d} \circ a = 0) \rangle \\
& \forall(s \mid s \in S : s \circ (\mathfrak{d} \circ a) = 0) \\
\Longleftrightarrow & \qquad \langle \text{ Definition 3(iii) } \rangle \\
& \forall(s \mid s \in S : (s \odot \mathfrak{d}) \circ a = 0) \\
\Longleftrightarrow & \qquad \langle \text{ } \mathfrak{d} \text{ is multiplicatively absorbing in } \mathcal{S} \ (s \odot \mathfrak{d} = \mathfrak{d}) \rangle \\
& \forall(s \mid s \in S : \mathfrak{d} \circ a = 0) \\
\Longleftrightarrow & \qquad \langle \text{ } \big({}_{\mathcal{S}}K, +\big) \text{ is zero-preserving } (\mathfrak{d} \circ a = 0) \rangle \\
& \forall(s \mid s \in S : \mathsf{true}) \\
\Longleftrightarrow & \qquad \langle \text{ } \forall\text{-True Body } \rangle \\
& \mathsf{true}
\end{array}
$$

(ii) a and b are fixed points $\implies$ $a+b$ is a fixed point

$$\begin{array}{ll}
& a+b \text{ is a fixed point} \\
\iff & \qquad \langle \text{ Definition 6(iv) } \rangle \\
& \forall(s \mid s \in S\backslash\{\mathfrak{d}\} : s \circ (a+b) = a+b\,) \\
\iff & \qquad \langle \text{ Definition 3(i) for } \big({}_{\mathcal{S}}K, +\big) \rangle \\
& \forall(s \mid s \in S\backslash\{\mathfrak{d}\} : s \circ a + s \circ b = a+b\,) \\
\Longleftarrow & \qquad \langle \text{ Hypothesis: } a \text{ and } b \text{ are fixed points } \rangle \\
& \forall(s \mid s \in S\backslash\{\mathfrak{d}\} : a+b = a+b\,) \\
\iff & \qquad \langle \text{ Reflexivity of } = \rangle \\
& \forall(s \mid s \in S\backslash\{\mathfrak{d}\} : \mathsf{true}\,) \\
\iff & \qquad \langle\ \forall\text{-True Body } \rangle \\
& \mathsf{true}
\end{array}$$

(iii) a and b are fixed points $\implies$ $a\,;b$ is a fixed point

$$\begin{array}{ll}
& a\,;b \text{ is a fixed point} \\
\iff & \qquad \langle \text{ Definition 6(iv) } \rangle \\
& \forall(s \mid s \in S\backslash\{\mathfrak{d}\} : s \circ (a\,;b) = a\,;b\,) \\
\iff & \qquad \langle \text{ Definition 5(i) } \rangle \\
& \forall\big(s \mid s \in S\backslash\{\mathfrak{d}\} : (s \circ a)\,;\big(\lambda(s,a) \circ b\big) = a\,;b\big) \\
\Longleftarrow & \qquad \langle\ \lambda(s,a) \in S \quad \& \quad \text{Hypothesis: } a \text{ and } b \text{ are fixed points } \rangle \\
& \forall(s \mid s \in S\backslash\{\mathfrak{d}\} : a\,;b = a\,;b\,) \\
\iff & \qquad \langle \text{ Reflexivity of } = \rangle \\
& \forall(s \mid s \in S\backslash\{\mathfrak{d}\} : \mathsf{true}\,) \\
\iff & \qquad \langle\ \forall\text{-True Body } \rangle \\
& \mathsf{true}
\end{array}$$

(iv) a is a fixed point $\wedge$ $\big(\mathcal{S},\mathcal{K}\big)$ is without reactivation $\implies$ $a^{\odot}$ is a fixed point

$a^{\odot}$ is a fixed point

$\iff$ $\langle$ Definition 6(iv) $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : s \circ a^{\odot} = a^{\odot})$

$\iff$ $\langle$ Proposition 3(vi) $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : +(n \mid n \geq 0 : s \circ a^{n}) = a^{\odot})$

$\Longleftarrow$ $\langle$ Hypothesis: a is a fixed point $\wedge$ $\big(\mathcal{S},\mathcal{K}\big)$ is without reactivation ($s \circ 1 = 1$) & Proposition 5(iii) $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : +(n \mid n \geq 0 : a^{n}) = a^{\odot})$

$\iff$ $\langle$ Definition of $a^{\odot}$: Equation (1) $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : a^{\odot} = a^{\odot})$

$\iff$ $\langle$ Reflexivity of $=$ $\rangle$

$\forall(s \mid s \in S\backslash\{\mathfrak{d}\} : \mathsf{true})$

$\iff$ $\langle$ $\forall$-True Body $\rangle$

true

A.6 Detailed Proof of Proposition 6

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

(i) a is a fixed point $\implies$ $\forall(b \mid b \in K \wedge b \neq 0 \wedge b \neq a : \neg(a \lhd b))$

$\forall(b \mid b \in K \wedge b \neq 0 \wedge b \neq a : \neg(a \lhd b))$

$\iff$ $\langle$ Definition 8 & Set Membership Axiom $\rangle$

$\forall(b \mid b \in K \wedge b \neq 0 \wedge b \neq a : b \notin \mathrm{Orb}(a))$

$\iff$ $\langle$ Trading for $\forall$ $\rangle$

$\forall(b \mid b \in K : (b \neq 0 \wedge b \neq a) \implies b \notin \mathrm{Orb}(a))$

$\Longleftarrow$ $\langle$ Hypothesis: a is a fixed point $\implies$ $\mathrm{Orb}(a) = \{0, a\}$ $\rangle$

$\forall(b \mid b \in K : (b \neq 0 \wedge b \neq a) \implies b \notin \{0, a\})$

$\iff$ $\langle$ $b \notin \{0, a\} \iff (b \neq 0 \wedge b \neq a)$ $\rangle$

$\forall(b \mid b \in K : (b \neq 0 \wedge b \neq a) \implies (b \neq 0 \wedge b \neq a))$

$\iff$ $\langle$ Reflexivity of $\implies$ $\rangle$

$\forall(b \mid b \in K : \mathsf{true})$

$\iff$ $\langle$ $\forall$-True Body $\rangle$

true

(ii) $a \sim_{\mathcal{K}} b \implies a \lhd b \;\wedge\; b \lhd a$

$$\begin{aligned}
& a \lhd b \;\wedge\; b \lhd a \\
\iff & \qquad \langle \text{ Definition 8 } \quad \& \quad \text{Set Membership Axiom } \rangle \\
& b \in \mathrm{Orb}(a) \;\wedge\; a \in \mathrm{Orb}(b) \\
\Longleftarrow & \qquad \langle \text{ Hypothesis: } a \sim_{\mathcal{K}} b \iff \mathrm{Orb}(a) = \mathrm{Orb}(b) \;\rangle \\
& b \in \mathrm{Orb}(b) \;\wedge\; a \in \mathrm{Orb}(a) \\
\iff & \qquad \langle \text{ Definition of } \mathrm{Orb}(b) \quad \& \quad \text{Definition of } \mathrm{Orb}(a) \;\rangle \\
& b \in \{s \circ b \mid s \in S\} \;\wedge\; a \in \{s \circ a \mid s \in S\} \\
\iff & \qquad \langle \text{ Set Membership Axiom } \rangle \\
& \exists(s \mid s \in S : s \circ b = b) \;\wedge\; \exists(s \mid s \in S : s \circ a = a) \\
\Longleftarrow & \qquad \langle \; \exists\text{-Introduction } \rangle \\
& \mathfrak{n} \circ b = b \;\wedge\; \mathfrak{n} \circ a = a \\
\iff & \qquad \langle \; \big({}_{\mathcal{S}}K, +\big) \text{ is unitary } (\mathfrak{n} \circ a = a) \quad \& \quad \text{Idempotence of } \wedge \;\rangle \\
& \mathsf{true}
\end{aligned}$$

(iii) $a \sim_{\mathcal{K}} b \implies (a \lhd c \iff b \lhd c)$

$$\begin{aligned}
& a \sim_{\mathcal{K}} b \implies (a \lhd c \iff b \lhd c) \\
\iff & \qquad \langle \text{ Mutual Implication } \rangle \\
& a \sim_{\mathcal{K}} b \implies \big[(a \lhd c \implies b \lhd c) \;\wedge\; (b \lhd c \implies a \lhd c)\big] \\
\iff & \qquad \langle \text{ Distributivity of } \implies \text{ over } \wedge \;\rangle \\
& \big[a \sim_{\mathcal{K}} b \implies (a \lhd c \implies b \lhd c)\big] \;\wedge\; \big[a \sim_{\mathcal{K}} b \implies (b \lhd c \implies a \lhd c)\big] \\
\iff & \qquad \langle \text{ Shunting } \rangle \\
& \big[a \sim_{\mathcal{K}} b \;\wedge\; a \lhd c \implies b \lhd c\big] \;\wedge\; \big[a \sim_{\mathcal{K}} b \;\wedge\; b \lhd c \implies a \lhd c\big] \\
\iff & \qquad \langle \text{ Definition of } \sim_{\mathcal{K}} \quad \& \quad \text{Definition 8 } \rangle \\
& \big[\mathrm{Orb}(a) = \mathrm{Orb}(b) \;\wedge\; c \in \mathrm{Orb}(a) \implies c \in \mathrm{Orb}(b)\big] \;\wedge \\
& \big[\mathrm{Orb}(a) = \mathrm{Orb}(b) \;\wedge\; c \in \mathrm{Orb}(b) \implies c \in \mathrm{Orb}(a)\big] \\
\iff & \qquad \langle \; x \in A \iff \{x\} \subseteq A \;\rangle \\
& \big[\mathrm{Orb}(a) = \mathrm{Orb}(b) \;\wedge\; \{c\} \subseteq \mathrm{Orb}(a) \implies \{c\} \subseteq \mathrm{Orb}(b)\big] \;\wedge \\
& \big[\mathrm{Orb}(a) = \mathrm{Orb}(b) \;\wedge\; \{c\} \subseteq \mathrm{Orb}(b) \implies \{c\} \subseteq \mathrm{Orb}(a)\big] \\
\iff & \qquad \langle \text{ Transitivity of } \subseteq \quad \& \quad \text{Idempotence of } \wedge \;\rangle \\
& \mathsf{true}
\end{aligned}$$

B Input File for Prover9

The main purpose of this appendix is to document the extent to which the proofs in this paper can be automated. For this purpose, we have used the Prover9 [21] automated theorem proving system. Below, we give the Prover9 input file for C^{2}KA. Using Prover9, we are able to automatically prove the results given in Proposition 2 and Proposition 3 (except for (vi) and (xii)). We do not provide the proof outputs from Prover9 in this paper.

```
% Saved by Prover9-Mace4 Version 0.5B, March 2008 (Dec 2007 LADR).

set(ignore_option_dependencies). % GUI handles dependencies

if(Prover9). % Options for Prover9
  assign(max_seconds, 60).
end_if.

if(Mace4).   % Options for Mace4
  assign(max_seconds, 60).
end_if.

formulas(assumptions).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONCURRENT KLEENE ALGEBRA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  x,x1,x2,x3,x4 are elements of K
% PK is +   -- choice of agent behaviours
% PC is *   -- parallel composition of agent behaviours
% SC is ;   -- sequential composition of agent behaviours
% PI is (*) -- parallel iteration of agent behaviours
% SI is (;) -- sequential iteration of agent behaviours
% 0k is the inactive agent
% 1k is the idle agent
% RK is the sub-behaviour relation on agent behaviours
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    PK(x1,PK(x2,x3)) = PK(PK(x1,x2),x3)                 # label("associativity of PK").
    PK(x1,x2) = PK(x2,x1)                               # label("commutativity of PK").
    PK(x,x) = x                                         # label("idempotence of PK").
    PK(x,"0k") = x                                      # label("identity of PK").

    PC(x1,PC(x2,x3)) = PC(PC(x1,x2),x3)                 # label("associativity of PC").
    PC(x1,x2) = PC(x2,x1)                               # label("commutativity of PC").
    PC(x,"1k") = x                                      # label("identity of PC").
    PC(x1,PK(x2,x3)) = PK(PC(x1,x2),PC(x1,x3))          # label("distributivity of PC over PK").
    PC(x,"0k") = "0k"                                   # label("annihilator of PC").

    SC(x1,SC(x2,x3)) = SC(SC(x1,x2),x3)                 # label("associativity of SC").
    SC(x,"1s") = x                                      # label("right identity of SC").
    SC("1s",x) = x                                      # label("left identity of SC").
    SC(PK(x1,x2),x3) = PK(SC(x1,x3),SC(x2,x3))          # label("right distributivity of SC over PK").
    SC(x1,PK(x2,x3)) = PK(SC(x1,x2),SC(x1,x3))          # label("left distributivity of SC over PK").
    SC(x,"0s") = "0s"                                   # label("right annihilator of SC").
    SC("0s",x) = "0s"                                   # label("left annihilator of SC").

    RK(SC(PC(x4,x1),PC(x2,x3)),PC(SC(x4,x2),SC(x1,x3)))   # label("exchange axiom").

    PK("1k",PC(x,PI(x))) = PI(x)                        # label("right unfold rule of PI").
    PK("1k",PC(PI(x),x)) = PI(x)                        # label("left unfold rule of PI").
    RK(PK(x3,PC(x1,x2)),x2) -> RK(PC(PI(x1),x3),x2)     # label("left induction rule of PI").
    RK(PK(x3,PC(x2,x1)),x2) -> RK(PC(x3,PI(x1)),x2)     # label("right induction rule of PI").
    PK("1k",SC(x,SI(x))) = SI(x)                        # label("right unfold rule of SI").
    PK("1k",SC(SI(x),x)) = SI(x)                        # label("left unfold rule of SI").
    RK(PK(x3,SC(x1,x2)),x2) -> RK(SC(SI(x1),x3),x2)     # label("left induction rule of SI").
    RK(PK(x3,SC(x2,x1)),x2) -> RK(SC(x3,SI(x1)),x2)     # label("right induction rule of SI").

    RK(x1,x2) <-> PK(x1,x2) = x2                        # label("definition of RK").


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% STIMULUS STRUCTURE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% y,y1,y2,y3 are elements of S
% PS is (+) -- choice of external stimuli
% DS is (.) -- sequential composition of external stimuli
% 0s is the deactivation stimulus
% 1s is the neutral stimulus
% RS is the sub-stimulus relation on external stimuli
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    PS(y1,PS(y2,y3)) = PS(PS(y1,y2),y3)                 # label("associativity of PS").
    PS(y1,y2) = PS(y2,y1)                               # label("commutativity of PS").
    PS(y,y) = y                                         # label("idempotence of PS").
    PS(y,"0s") = y                                      # label("identity of PS").

    DS(y1,DS(y2,y3)) = DS(DS(y1,y2),y3)                 # label("associativity of DS").
    DS(y,"1s") = y                                      # label("right identity of DS").
    DS("1s",y) = y                                      # label("left identity of DS").
    DS(PS(y1,y2),y3) = PS(DS(y1,y3),DS(y2,y3))          # label("right distributivity of DS over PS").
    DS(y1,PS(y2,y3)) = PS(DS(y1,y2),DS(y1,y3))          # label("left distributivity of DS over PS").
    DS(y,"0s")="0s"                                     # label("right annihilator of DS").
    DS("0s",y)="0s"                                     # label("left annihilator of DS").

    RS(y1,y2) <-> PS(y1,y2) = y2                        # label("definition of RS").


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LEFT S-SEMIMODULE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   f: S x K -> K is the next behaviour mapping o
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    f(y,PK(x1,x2)) = PK(f(y,x1),f(y,x2))                # label("distributivity of f over PK").
    f(PS(y1,y2),x) = PK(f(y1,x),f(y2,x))                # label("distributivity of f over PS").
    f(DS(y1,y2),x) = f(y1,f(y2,x))                      # label("sequential application of f").
    f("1s",x) = x                                       # label("unitary f").
    f("0s",x) = "0k"                                    # label("zero-preserving f").


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% RIGHT K-SEMIMODULE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  g: S x K -> S is the next stimulus mapping lambda
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    g(PS(y1,y2),x) = PS(g(y1,x),g(y2,x))                # label("distributivity of g over PS").
    g(y,PK(x1,x2)) = PS(g(y,x1),g(y,x2))                # label("distributivity of g over PK").
    g(y,SC(x1,x2)) = g(g(y,x1),x2)                      # label("sequential application of g").
    g(y,"1k") = y                                       # label("unitary g").
    g(y,"0k") = "0s"                                    # label("zero-preserving g").
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% COMMUNICATING CONCURRENT KLEENE ALGEBRA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    f(y,SC(x1,x2)) = SC(f(y,x1),f(g(y,x1),x2))           # label("cascading axiom").
    RK(x3,x1) | SC(f(y,x1),f(g(y,x3),x2)) = "0k"         # label("cascading output law").
    g(DS(y1,y2),x) = DS(g(y1,f(y2,x)),g(y2,x))           # label("sequential output axiom").


end_of_list.



formulas(goals).


    RK(SI(x),PI(x))                                      # label("Proposition 2").

    f(y,"0k") = "0k"                                     # label("Proposition 3(i)").
    RK(x1,x2) -> RK(f(y,x1),f(y,x2))                     # label("Proposition 3(ii)").
    RS(y1,y2) -> RK(f(y1,x),f(y2,x))                     # label("Proposition 3(iii)").
    RK(x1,x2) & RS(y1,y2) -> RK(f(y1,x1),f(y2,x1))       # label("Proposition 3(iv)").
    RK(f(y,PK(SC(x1,x2),SC(x2,x1))),f(y,PC(x1,x2)))      # label("Proposition 3(v)").
    RK(f(y,SI(x)),f(y,PI(x)))                            # label("Proposition 3(vi)").

    g("0s",x) = "0s"                                     # label("Proposition 3(viii)").
    RS(y1,y2) -> RS(g(y1,x),g(y2,x))                     # label("Proposition 3(ix)").
    RK(x1,x2) -> RS(g(y,x1),g(y,x2))                     # label("Proposition 3(x)").
    RK(x1,x2) & RS(y1,y2) -> RS(g(y1,x1),g(y2,x1))       # label("Proposition 3(xi)").
    RS(g(y,PK(SC(x1,x2),SC(x2,x1))),g(y,PC(x1,x2)))      # label("Proposition 3(xii)").
    RS(g(y,SI(x)),g(y,PI(x)))                            # label("Proposition 3(xiii)").


end_of_list.
```