DyNetKAT: An Algebra of Dynamic Networks

Georgiana Caltais
University of Konstanz, Germany

Hossein Hojjat
Tehran Institute for Advanced Studies, Iran

Mohammad Reza Mousavi
University of Leicester, UK

Hünkar Can Tunç
University of Konstanz, Germany

Abstract

We introduce a formal language for specifying dynamic updates for Software Defined Networks. Our language builds upon Network Kleene Algebra with Tests (NetKAT) and adds constructs for synchronisations and multi-packet behaviour to capture the interaction between the control- and data-plane in dynamic updates. We provide a sound and ground-complete axiomatization of our language. We exploit the equational theory to provide an efficient reasoning method about safety properties for dynamic networks. We implement our equational theory in DyNetiKAT – a tool prototype, based on the Maude Rewriting Logic and the NetKAT tool, and apply it to a case study. We show that we can analyse the case study for networks with hundreds of switches using our initial tool prototype.

Subject Classification Theory of computation → Semantics and reasoning

Keywords and phrases Software Defined Networks, Dynamic Updates, Dynamic Network Reconfiguration, NetKAT, Process Algebra, Equational Reasoning

Funding *Georgiana Caltais*: supported by the DFG project "CRENKAT", proj. no. 398056821
Mohammad Reza Mousavi: supported by the UKRI Trustworthy Autonomous Systems Node in Verifiability, Grant Award Reference EP/V026801/1.
Hünkar Can Tunç: supported by the DFG project "CRENKAT", proj. no. 398056821

1 Introduction

Software defined networking (SDN) has gained immense popularity due to simplicity in network management and offering network programmability. Many programming languages have been designed for programming SDNs [25, 15]. They range from industrial-scale, hardware-oriented and low-level programming languages such as OpenFlow [18] to domain-specific, high-level and programmer-centric languages such as Frenetic [10]. In recent years, there has been a growing interest in analysable languages based on mathematical foundations which provide a solid reasoning framework to prove correctness properties in SDNs (e.g., safety).

There is a spectrum of mathematically inspired network programming languages that varies between those with a small number of language constructs and those with expressive language design which allow them to support more networking features. On the more expressive side of the spectrum, Flowlog [20] is an example of a language that uses a powerful formalism (first-order Horn clause logic) to program a Software Defined Network (SDN). In order to keep the language decidable, Flowlog disallows recursion in the clauses. For the purpose of formal analysis of a Flowlog program, the authors of [20] provide a translator to the Alloy tool. As another example of an expressive language, Kinetic [14] is a language based on finite state machines that is mostly geared towards dynamic feature of SDNs. Model checking is used to formally analyse the Kinetic programs. NetKAT [2, 9] is an example of a minimalist language based on Kleene algebra with tests that has a sound and complete equational theory. While the core of the language is very simple with a few number of operators, the language has been extended in various ways to support different aspects of networking such as congestion control [8], history-based routing [5] and higher-order functions [26].

Our starting point is NetKAT, because it provides a clean and analyseable framework for specifying SDNs. The minimalist design of NetKAT does not cater for some common (failure) patterns in SDNs, particularly those arising from dynamic reconfiguration and the interaction between the data- and control-plane flows. In [16], the authors have proposed an extension to NetKAT to support stateful network updates. The extension embraces the notion of mutable state in the language which is in contrast to its pure functional nature. The purpose of this paper is to propose an extension to NetKAT to support dynamic and stateful behaviours. To this end, we pledge to keep the minimalist design of NetKAT with adding only a few number of new operators. Furthermore, our extension does not contradict the nature of the language.

A number of concurrent extensions of NetKAT have been introduced to date [22, 27, 13]. These extensions followed different design decisions than the present paper and a comparison of their approaches with ours is provided in Section 2; however, the most important difference lies in the fact that inspired by earlier abstractions in this domain [21], we were committed to create different layers for data-plane flows and dynamic updates such that every data-plane packet observes a single set of flow tables through its flight through the network. This allowed us, unlike the earlier approaches, to build a layer on top of NetKAT without modifying its semantics.

1.1 Running Examples

Throughout the paper, we focus on modelling with DyNetKAT two examples that involve dynamically updating the network configuration. In the first example, stateful firewall, the data-plane initiates the update by allowing a disallowed path in the network as a result of requests received from the trusted intranet. In the second, distributed controller, the control-plane initiates the update by modifying the forwarding route of a packet in a multi-controller setting.

▶ **Example 1.** A firewall is supposed to protect the intranet of an organization from unauthorised access from the Internet. However, due to certain requests from the intranet, it should be able to open up connections from the Internet to intranet. An example is when a user within the intranet requests a secure connection to a node on the Internet; in that case, the response from the node should be allowed to enter the intranet. The behaviour of updating the flow tables with respects to some events in the network such as receiving a specific packet is a challenging phenomenon for languages such as NetKAT.

Figure 1 shows a simplified version of the stateful firewall network. In this version, the *Switch* does not allow any packet from the port *ext* to *int* at the beginning. When the *Host* sends a request to the *Switch* it opens up the connection.

▶ **Example 2.** Another running example concerns a well-known challenge in SDNs, namely, race conditions resulting from dynamic updates of flow-tables and in-flight packets [17, 24]. Below we specify a typical scenario for such race conditions; similar scenarios concerning actual bugs are abundant in the literature [24, 11, 12].

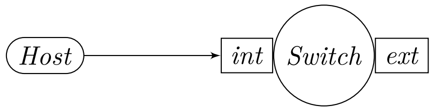


Figure 1 Stateful Firewall

Consider the network topology depicted in Figure 2. The controller $C1$ controls the top part of the network (switches $S1$, $S3$ and $S5$) and the controller $C2$ is responsible for the bottom part. Initially, the packets from $H1$, which enter the network through switch $S1$ (port 2), should be routed through switches $S5$ (through ports 6 and 7), $S6$ (through ports 8 and 10) and finally port 12 of switch $S2$, to reach $H2$. Due to an event, the controllers have to take down the previous route, and to install a new route in the network that routes packets from $H3$ through $S3$ (ports 1 to 3), $S5$ (through ports 5 to 7), $S6$ (ports 8 to 9) to switch $S4$ (port 11) to finally reach $H4$. It is an important security property that the traffic in these two routes should not mix. In particular, it will be serious breach if packets from $H1$ arrive at $H4$ or vice versa, packets from $H3$ arrive at $H2$.

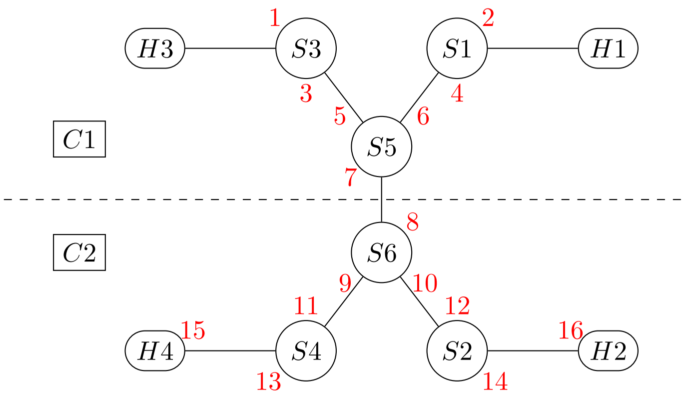


Figure 2 Race Condition in a Distributed Controller

1.2 Our Contributions

The contributions of this paper are summarized as follows:

- we define the syntax and operational semantics of a dynamic extension of NetKAT that allows for modelling and reasoning about control-plane updates and their interaction with data-plane flows;
- we give a sound and ground-complete axiomatization of our languages; and
- we devise analysis methods for reasoning about flow properties using our axiomatization, apply them on examples from the domain and gather and analyze evidence of applicability and efficiency for our approach.

1.3 Structure of Paper

In Section 2, we provide a brief overview of NetKAT, review our design decision and introduce the syntax and operational semantics of DyNetKAT. In Section 3, we investigate some semantic properties of DyNetKAT by defining a notion of behavioural equivalence and providing a sound and ground-complete axiomatization. We exploit this axiomatization in Section 4 in an analysis method. We implement and apply our analysis method in Section 5 on a case study and report about its scalability on large examples with hundreds of switches. We conclude the paper and present some avenues for future work in Section 6.

2 Language Design

In what follows, we provide a brief overview of the NetKAT syntax and semantics [2]. Then, we motivate our language design decisions, we introduce the syntax of DyNetKAT and its underlying semantics, and provide the corresponding encoding of our running examples presented in Section 1.1.

2.1 Brief Overview of NetKAT

We proceed by first introducing some basic notions that are used throughout the paper.

▶ **Definition 1** (Network Packets.). *Let $F = \{f_1, \ldots, f_n\}$ be a set of field names f_i with $i \in \{1, \ldots n\}$. We call* network packet *a function in $F \to \mathbb{N}$ that maps field names in F to values in $\mathbb{N}$. We use σ, σ' to range over network packets. We write, for instance, $\sigma(f_i) = v_i$ to denote a test checking whether the value of f_i in σ is v_i. Furthermore, we write $\sigma[f_i := n_i]$ to denote the assignment of f_i to v_i in σ.*

A (possibly empty) list of packets *is formally defined as a function from natural numbers to packets, where the natural number in the domain denotes the position of the packet in the list such that the domain of the function forms an interval starting from* 0*.*

The empty list *is denoted by $\langle\rangle$ and is formally defined as the empty function (the function with the empty set as its domain). Let σ be a packet and l be a list, then $\sigma :: l$ is the list l' in which σ is at position 0 in l', i.e., $l'(0) = \sigma$, and $l'(i+1) = l(i)$, for all i in the domain of l.*

In Figure 3, we recall the NetKAT syntax and semantics [2].

NetKAT Syntax:

$$Pr ::= \mathbf{0} \mid \mathbf{1} \mid Pr + Pr \mid Pr \cdot Pr \mid \neg Pr$$
$$N ::= Pr \mid f \leftarrow n \mid N + N \mid N \cdot N \mid N^* \mid \mathbf{dup}$$

NetKAT Semantics:

$$\begin{aligned}
[\![\mathbf{1}]\!](h) &\triangleq \{h\} \\
[\![\mathbf{0}]\!](h) &\triangleq \{\} \\
[\![f = n]\!]\ (\sigma{::}h) &\triangleq \begin{cases} \{\sigma{::}h\} & \text{if } \sigma(f) = n \\ \{\} & \text{otherwise} \end{cases} \\
[\![\neg a]\!]\ (h) &\triangleq \{h\} \setminus [\![a]\!]\ (h) \\
[\![f \leftarrow n]\!]\ (\sigma{::}h) &\triangleq \{\sigma[f := n]{::}h\} \\
[\![p + q]\!]\ (h) &\triangleq [\![p]\!]\ (h) \cup [\![q]\!]\ (h)
\end{aligned}
\qquad
\begin{aligned}
[\![p \cdot q]\!]\ (h) &\triangleq ([\![p]\!] \bullet [\![q]\!])\ (h) \\
[\![p^*]\!]\ (h) &\triangleq \textstyle\bigcup_{i \in N} F^i\ (h) \\
F^0\ (h) &\triangleq \{h\} \\
F^{i+1}\ (h) &\triangleq ([\![p]\!] \bullet F^i)\ (h) \\
(f \bullet g)(x) &\triangleq \textstyle\bigcup \{g(y) \mid y \in f(x)\} \\
[\![\mathbf{dup}]\!]\ (\sigma{::}h) &\triangleq \{\sigma{::}(\sigma{::}h)\}
\end{aligned}$$

Figure 3 NetKAT: Syntax and Semantics [2]

The predicate for dropping a packet is denoted by **0**, while passing on a packet (without any modification) is denoted by **1**. The predicate checking whether the field f of a packet has value n is denoted by $(f = n)$; if the predicate fails on the current packet it results on dropping the packet, otherwise it will pass the packet on. Disjunction and conjunction between predicates are denoted by $Pr + Pr$ and $Pr \cdot Pr$, respectively. Negation is denoted by $\neg Pr$. Predicates are the basic building blocks of NetKAT policies and hence, a predicate is a policy by definition. The policy that modifies the field f of the current packet to take value n is denoted by $(f \leftarrow n)$. A multicast behaviour of policies is denoted by $N + N$, while sequencing policies (to be applied on the same packet) are denoted by $N \cdot N$. The repeated application of a policy is encoded as N^*. The construct **dup** simply makes a copy of the current network packet.

In [2], lists of packets are referred to as *histories*. Let H stand for the set of packet histories, and $\mathcal{P}(H)$ denote the powerset of H. More formally, the denotational semantics of NetKAT policies is inductively defined via the semantic map $[\![-]\!] : N \to (H \to \mathcal{P}(H))$ in Figure 3, where N stands for the set of NetKAT policies, $h \in H$ is a packet history, $a \in Pr$ denotes a NetKAT predicate and $\sigma \in F \to \mathbb{N}$ is a network packet.

For a reminder, the equational axioms of NetKAT, denoted by E_{NK}, are provided in Figure 4. E_{NK} includes the Kleene Algebra axioms (KA-...), Boolean Algebra axioms (BA-...) and Packet Algebra axioms (PA-...). The novelty is the set of PA-axioms. In short, PA-MOD-MOD-COMM states that the order in which two different packet fields are assigned does not matter. PA-MOD-FILTER-COMM encodes a similar property, for the case of a field assignment followed by a test of a different field's value. PA-MOD-FILTER ignores the test of a field preceded by an assignment of the same value to the field. Orthogonaly, PA-FILTER-MOD ignores a field assignment preceded by a test against the assigned value. PA-MOD-MOD states that a sequence of assignments to the same field only takes into consideration the last assignment. PA-CONTRA encodes the fact that a field cannot have two different values at the same point. PA-MATCH-ALL identifies the policy accepting all the packets with the sum of all possible tests of a field's value. Intuitively, PA-DUP-FILTER-COMM states that adding the current packet to the history is independent of tests.

$p + (q + r) \equiv (p + q) + r$	KA-PLUS-ASSOC	$a + (b \cdot c) \equiv (a + b) \cdot (a + c)$	BA-PLUS-DIST
$p + q \equiv q + p$	KA-PLUS-COMM	$a + 1 \equiv 1$	BA-PLUS-ONE
$p + 0 \equiv p$	KA-PLUS-ZERO	$a + \neg a \equiv 1$	BA-EXCL-MID
$p + p \equiv p$	KA-PLUS-IDEM	$a \cdot b \equiv b \cdot a$	BA-SEQ-COMM
$p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$	KA-SEQ-ASSOC	$a \cdot \neg a \equiv 0$	BA-CONTRA
$1 \cdot p \equiv p$	KA-ONE-SEQ	$a \cdot a \equiv a$	BA-SEQ-IDEM
$p \cdot 1 \equiv p$	KA-SEQ-ONE		
$p \cdot (q + r) \equiv p \cdot q + p \cdot r$	KA-SEQ-DIST-L	$f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n, \text{if } f \neq f'$	PA-MOD-MOD-COMM
$(p + q) \cdot r \equiv p \cdot r + q \cdot r$	KA-SEQ-DIST-R	$f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n, \text{if } f \neq f'$	PA-MOD-FILTER-COMM
$0 \cdot p \equiv 0$	KA-ZERO-SEQ	$\mathbf{dup} \cdot f = n \equiv f = n \cdot \mathbf{dup}$	PA-DUP-FILTER-COMM
$p \cdot 0 \equiv 0$	KA-ZERO-SEQ	$f \leftarrow n \cdot f = n \equiv f \leftarrow n$	PA-MOD-FILTER
$1 + p \cdot p^* \equiv p^*$	KA-UNROLL-L	$f = n \cdot f \leftarrow n \equiv f = n$	PA-FILTER-MOD
$1 + p^* \cdot p \equiv p^*$	KA-UNROLL-R	$f \leftarrow n \cdot f \leftarrow n' \equiv f \leftarrow n'$	PA-MOD-MOD
$q + p \cdot r \leq r \Rightarrow p^* \cdot q \leq r$	KA-LFP-L	$f = n \cdot f = n' \equiv 0, \text{if } n \neq n'$	PA-CONTRA
$p + q \cdot r \leq q \Rightarrow p \cdot r^* \leq q$	KA-LFP-R	$\Sigma_i f = i \equiv 1$	PA-MATCH-ALL

Figure 4 E_{NK}: NetKAT Equational Axioms [2]

2.2 Design Decisions

Our main motivation behind DyNetKAT was to have a *minimalistic* language that can model *control-plane* and *data-plane* network traffic and their interaction. Our choice for a minimal language is motivated by our desire to use our language as a basis for scalable analysis. We would like to be able to compile major practical languages into ours. Our minimal design helps us reuse much of the well-known scalable analysis techniques. Regarding its modelling capabilities, we are interested in modelling the stateful and dynamic behaviour of networks emerging from these interactions. We would like to be able to model control messages, connections between controllers and switches, data packets, links among switches, and model and analyse their interaction in a seamless manner.

Based on these motivations, we started off with NetKAT as a fundamental and minimal network programming language, which allows us to model the basic policies governing the network traffic. The choice of NetKAT, in addition to its minimalist nature, is motivated by its rigorous semantics and equational theory, and the existing techniques and tools for its analysis. This motivated our next design constraint, namely, to build upon NetKAT in a hierarchical manner and without redefining its semantics. This constraint should not be taken lightly as the challenges in the recent concurrent extensions of NetKAT demonstrated [22, 27, 13]. We will elaborate on this point, in the presentation of our syntax and semantics. We could achieve this thanks to the abstractions introduced in the domain [21] that allowed for a neat layering of data-plane and control-plan flows such that every data-plane flow sees one set of flow-tables in its flight through the network.

We then introduced few extensions and modifications to cater for the phenomena we desired to model in our extension regarding control-plane and dynamic and stateful behaviour:

- Synchronization: we introduced a basic mechanism of handshake synchronization with the possibility of communicating a network program (a flow table). This construct allows for capturing the dynamicity and interaction between the control and data planes.
- Guarded recursion: we introduced the concept of recursion to model the (persistent) dynamic changes that result from control messages and stateful behaviour; in other words, recursion is used to model the new state of the flow tables. An alternative modelling construct could have been using "global" variables and guards, but we preferred recursion due to its neat algebraic representation. We restricted the use of recursion to guarded recursion, that is a policy should be applied before changing state to a new recursive definition, in order to remain within a decidable and analyse-able realm. A natural extension of our framework could introduce formal parameters and parameterised recursive variables; this future extension is orthogonal to our existing extensions and in this paper, we go for a minimal extension in which the parameters are coded in variable names.
- Multi-packet semantics: we introduce the semantics of treating a list of packets, which is essential for studying the interaction between control- and data plane packets. This is in contrast with NetKAT where a single-packet semantics is introduced. The introduction of multi-packet semantics also called for a new operator to denote the end of applying a flow-table to the current packet and proceeding with the next packet (possibly with the modified flow-table in place). This is our new sequential composition operator, denoted by ";".

2.3 DyNetKAT Syntax

As already mentioned, NetKAT provides the possibility of recording the individual "hops" that packets take as they go through the network by using the so-called **dup** construct. The latter keeps track of the state of the packet at each intermediate hop. As a brief reminder of the approach in [2]: assume a NetKAT switch policy p and a topology t, together with an ingress *in* and an egress *out*. Checking whether *out* is reachable from *in* reduces to checking: $in \cdot \mathbf{dup} \cdot (p \cdot t \cdot \mathbf{dup})^* \cdot out \not\equiv \mathbf{0}$ (see Definition 2 and Theorem 4 in [2]). Furthermore, as shown in [9], **dup** plays a crucial role in devising the NetKAT language semantics in a coalgebraic fashion, via Brzozowski-like derivatives on top of NetKAT coalgebras (or NetKAT automata) corresponding to NetKAT expressions.

We decided to depart from NetKAT in this respect, due to our important constraint not to redefine the NetKAT semantics: the **dup** expression allows for observable intermediate steps that result from incomplete application of flow-tables and in concurrency scenarios, the same data packet may become subject to more than one flow table due to the concurrent interactions with the control plain. For this semantics to be compositional, one needs to define a small step operational semantics in such a way that the small steps in predicate evaluation also become visible (see our past work on compositionality of SOS with data on such constraints [19]). This will first break our constrain in building upon NetKAT semantics and secondly, due to the huge number of possible interleavings, make the resulting state-space intractable for analysis.

In addition to the argumentation above, note that similarly to the approach in [2], we work with packet fields ranging over finite domains. Consequently, our analyses can be formulated in terms of reachability properties, further verifiable by means of **dup**-free expressions of shape: $in \cdot (p \cdot t)^* \cdot out \not\equiv \mathbf{0}$. Hence, we chose to define DyNetKAT synchronization, guarded recursion and multi-packet semantics on top of the **dup**-free fragment of NetKAT, denoted by NetKAT$^{-\mathbf{dup}}$.

The syntax of DyNetKAT is defined on top of the **dup**-free fragment of NetKAT as:

$$\begin{array}{lll} N & ::= & \text{NetKAT}^{-\mathbf{dup}} \\ D & ::= & \bot \mid N \,; D \mid x?N \,; D \mid x!N \,; D \mid D \,||\, D \mid D \oplus D \mid X \\ & & X \triangleq D \end{array} \tag{1}$$

We sometimes write $p \in$ NetKAT, $p \in$ NetKAT$^{-\mathbf{dup}}$ or, respectively, $p \in$ DyNetKAT in order to refer to a NetKAT, NetKAT$^{-\mathbf{dup}}$ or, respectively, DyNetKAT policy p.

The DyNetKAT-specific constructs are as follows. By $\bot$ we denote a dummy policy without behaviour. Our new sequential composition operator, denoted by $N \,; D$, specifies when the NetKAT$^{-\mathbf{dup}}$ policy N applicable to the current packet has come to a successful end and, thus, the packet can be transmitted further and the next packet can be fetched for processing according to the rest of the policy D.

Communication in DyNetKAT, encoded via $x!N; D$ and $x?N; D$, consists of two steps. In the first place, sending and receiving NetKAT$^{-\mathbf{dup}}$ policies through channel x are denoted by $x!N$, and $x?N$. Intuitively, these correspond to updating the current network configuration according to N. Secondly, as soon as the sending or receiving messages are successfully communicated, a new packet is fetched and processed according to D. The parallel composition of two DyNetKAT policies (to enable synchronization) is denoted by $D \,||\, D$.

As it will become clearer in Section 2.4 (semantics), communication in DyNetKAT guarantees preservation of well-defined behaviours when transitioning between network configurations. This corresponds to the so-called per-packet consistency in [21], and it guarantees that every packet traversing the network is processed according to exactly one NetKAT$^{-\mathbf{dup}}$ policy.

Non-deterministic choice of DyNetKAT policies is denoted by $D \oplus D$. For a non-determinstic choice over a finite domain P, we use the syntactic sugar $\oplus_{p \in P} P'$, where p appears as "bound variable" in P'; this is interpreted as a sum of finite summand by replacing the variable p with all its possible values in P.

Finally, one can use recursive variables X in the specification of DyNetKAT policies, where each recursive variable should have a unique defining equation $X \triangleq D$.

For the simplicity of notation, we do not explicitly specify the trailing "$;\bot$" in our policy specifications, whenever clear from the context.

In Figure 5 we provide the DyNetKAT formalization of the firewall in Example 1. In the DyNetKAT encoding, we use the message channel *secConReq* to open up the connection and *secConEnd* to close it. We model the behavior of the switch using the two programs *Switch* and *Switch'*.

$$
\begin{array}{lcl}
\mathit{Host} & \triangleq & \mathit{secConReq}!\mathbf{1} \,;\, \mathit{Host} \oplus \\
 & & \mathit{secConEnd}!\mathbf{1} \,;\, \mathit{Host} \\
\\
\mathit{Switch} & \triangleq & \big((\mathit{port} = \mathit{int}) \cdot (\mathit{port} \leftarrow \mathit{ext})\big) \,;\, \mathit{Switch} \oplus \\
 & & \big((\mathit{port} = \mathit{ext}) \cdot \mathbf{0}\big) \,;\, \mathit{Switch} \oplus \\
 & & \mathit{secConReq}?\mathbf{1} \,;\, \mathit{Switch}' \\
\\
\mathit{Switch}' & \triangleq & \big((\mathit{port} = \mathit{int}) \cdot (\mathit{port} \leftarrow \mathit{ext})\big) \,;\, \mathit{Switch}' \oplus \\
 & & \big((\mathit{port} = \mathit{ext}) \cdot (\mathit{port} \leftarrow \mathit{int})\big) \,;\, \mathit{Switch}' \oplus \\
 & & \mathit{secConEnd}?\mathbf{1} \,;\, \mathit{Switch} \\
\\
\mathit{Init} & \triangleq & \mathit{Host} \,||\, \mathit{Switch}
\end{array}
$$

Figure 5 Stateful Firewall in DyNetKAT

In Figure 6 we provide the DyNetKAT formalization of the distributed controllers in Example 2. In the code in Figure 6 the controllers work independently to update the network (which can lead to security breach). The specification $\mathit{SwitchX}_{ft}$ is a generic specification for the behaviour of all switches in this example; the domain of P in this example is the set of all 5 policies that are being communicated, such as $\mathbf{0}$, $((\mathit{port} = 11) \cdot (\mathit{port} \leftarrow 13))$, and $((\mathit{port} = 5) \cdot (\mathit{port} \leftarrow 7))$.

However, in the code in Figure 7 the controllers synchronise before updating the rest of the switches.

2.4 DyNetKAT Semantics

The operational semantics of DyNetKAT in Figure 8 is provided over configurations of shape (d, H, H'), where d stands for the current DyNetKAT policy, H is the list of packets to be processed by the network according to d and H' is the list of packets handled successfully by the network. The rule labels γ range over pairs of packets (σ, σ') or communication/reconfiguration-like actions of shape $x!q$, $x?q$ or $\mathbf{rcfg(x, q)}$, depending on the context.

Note that the DyNetKAT semantics is devised in a "layered" fashion. Rule $(\mathbf{cpol}^{\checkmark}_{\_;})$ in Figure 8 is the base rule that makes the transition between the NetKAT denotations and DyNetKAT operations. More precisely, whenever σ' is a packet resulted from the successful evaluation of a NetKAT policy p on σ, a (σ, σ')-labelled step is observed at the level of DyNetKAT. This transition applies whenever the current configuration encapsulates a DyNetKAT policy of shape $p; q$ and a list of packets to be processed starting with σ. The resulting configuration continues with evaluating q on the next packet in the list, while σ' is marked as successfully handled by the network.

$$
\begin{array}{lll}
L & \triangleq & (((port = 3) \cdot (port \leftarrow 5)) + \\
& & ((port = 4) \cdot (port \leftarrow 6)) + \\
& & ((port = 7) \cdot (port \leftarrow 8)) + \\
& & ((port = 9) \cdot (port \leftarrow 11)) + \\
& & ((port = 10) \cdot (port \leftarrow 12)) + \\
& & ((port = 13) \cdot (port \leftarrow 15)) + \\
& & ((port = 14) \cdot (port \leftarrow 16))) \\
\\
S_1 & \triangleq & (port = 2) \cdot (port \leftarrow 4) \\
S_2 & \triangleq & (port = 12) \cdot (port \leftarrow 14) \\
S_3 & \triangleq & \mathbf{0} \\
S_4 & \triangleq & \mathbf{0} \\
S_5 & \triangleq & (port = 6) \cdot (port \leftarrow 7) \\
S_6 & \triangleq & (port = 8) \cdot (port \leftarrow 10) \\
SDN_{X_1,\ldots,X_6} & \triangleq & ((X_1 + \ldots + X_6) \cdot L)^* \,; SDN_{X_1,\ldots,X_6} \oplus \\
& & \sum_{X_i' \in FT} upSi?X_i' \,; SDN_{X_1,\ldots,X_i',\ldots,X_6} \\
\\
ft3 & \triangleq & (port = 1) \cdot (port \leftarrow 3) \\
ft4 & \triangleq & (port = 11) \cdot (port \leftarrow 13) \\
ft5 & \triangleq & (port = 5) \cdot (port \leftarrow 7) \\
ft6 & \triangleq & (port = 8) \cdot (port \leftarrow 9) \\
FT & = & \{\mathbf{0}, ft3, ft4, ft5, ft6\} \\
\\
SDN & \triangleq & SDN_{S_1,\ldots,S_6} \,||\, C_1 \,||\, C_2 \\
\\
C_1 & \triangleq & upS1!\mathbf{0} \,||\, upS3!ft3 \,||\, upS5!ft5 \\
\\
C_2 & \triangleq & upS2!\mathbf{0} \,||\, upS4!ft4 \,||\, upS6!ft6
\end{array}
$$

Figure 6 Distributed Controller in DyNetKAT: Independent Controllers

$$
\begin{array}{lll}
C_1 & \triangleq & upS1!\mathbf{0}; \\
& & syn!\mathbf{1}; \\
& & upS3!((port = 1) \cdot (port \leftarrow 3)); \\
& & upS5!((port = 5) \cdot (port \leftarrow 7))
\end{array}
\qquad
\begin{array}{lll}
C_2 & \triangleq & upS2!\mathbf{0}; \\
& & syn?\mathbf{1}; \\
& & upS4!((port = 11) \cdot (port \leftarrow 13)); \\
& & upS6!((port = 8) \cdot (port \leftarrow 9))
\end{array}
$$

Figure 7 Distributed Controller in DyNetKAT: Synchronizing Controllers

The remaining rules in Figure 8 define non-deterministic choice, synchronization and recursion in the standard fashion.

Rules ($\mathbf{cpol}_{\_\oplus}$) and ($\mathbf{cpol}_{\oplus\_}$) define non-deterministic behaviours. Assume H_0 is the list of packets to be processed by the network according to p (respectively, q) and H_0' is the list of packets handled successfully by the network. Whenever p (respectively, q) determines a γ-labelled transition into (p', H_1, H_1') (respectively, (q', H_1, H_1')), the policy $p \oplus q$ is able to mimic the same behaviour. Rules ($\mathbf{cpol}_{\_||}$) and ($\mathbf{cpol}_{||\_}$) follow a similar pattern; the only difference is that the "inactive" operand is preserved by the target of the semantic rule.

Mere sending ($\mathbf{cpol}_{!}$) and receiving ($\mathbf{cpol}_{?}$) entail transitions labelled accordingly, and continue with the DyNetKAT policy following the ; operator. Note that the list of packets to be processed by the network and the list of packets handled successfully by the network remain unchanged.

DyNetKAT synchronization is defined by ($\mathbf{cpol}_{!?}$) and ($\mathbf{cpol}_{?!}$). Intuitively, when both operands q and, respectively, s "agree" on sending/receiving a policy p on channel x in the context of the same packet lists H and H', and behave like q', respectively, s' afterwards, then a $\mathbf{rcfg}(\mathbf{x}, \mathbf{p})$ step can be observed. The system proceeds with the continuation behaviour $q' \,||\, s'$.

As denoted by ($\mathbf{cpol}_{\mathbf{X}}$), a recursive variable defined as $X \triangleq p$ behaves according to p.

In Figure 9 we depict a labelled transition system (LTS) encoding a possible behaviour of the stateful firewall in Example 1. We assume the list of network packets to be processed consists of a "safe" packet σ_i travelling from *int* to *ext* (i.e., $\sigma_i(port) = int$) followed by a potentially "dangerous" packet σ_e travelling from *ext* to *int* (i.e., $\sigma_e(port) = ext$). For the simplicity of notation, in Figure 9 we write H for *Host*, S for *Switch*, S' for *Switch'*, *SCR* for *secConReq* and *SCE* for *secConEnd*. Note that σ_e can enter the network only if a secure connection request was received. More precisely, the transition labelled (σ_e, σ_i) is preceded by a transition labelled $SCR?1$ or $\mathbf{rcfg}(SCR, \mathbf{1})$: $n_2 \xrightarrow{SCR?1,\ \mathbf{rcfg}(SCR,\mathbf{1})} n_3 \xrightarrow{(\sigma_e,\sigma_i)} n_4$.

In Figure 10 we depict an excerpt of the LTS corresponding to the distributed independent controllers in Example 2, given a network packet denoted by σ_2. In Figure 10 we write σ_i to denote a network packet such that $\sigma_i(port) = i$. For instance, transitions of shape $n_0 \xrightarrow{(\sigma_2,\sigma_i)} \overline{n}_i$ encode forwarding of the current packet from port 2 to port i based on the subsequent unfoldings of the Kleene-star expression in the definition of $SDN_{X_1,\ldots,X_6}$. The transition $n_2 \xrightarrow{(\sigma_2,\sigma_{15})} \overline{\overline{n}}_{15}$ reveals a breach in the network corresponding to the possibility of forwarding the current packet from H_1 to H_4. This is possible due to two consecutive reconfigurations of the flow tables of switches S_6 and S_4, respectively, enabling traffic from port 8 to 9, and from port 11 to 13.

3 Semantic Results

In this section we define bisimilarity of DyNetKAT policies, introduce some necessary definitions and terminology, and provide a corresponding sound and complete axiomatization.

3.1 An Axiom System for DyNetKAT Bisimilarity

Bisimilarity of DyNetKAT terms is defined in the standard fashion:

▶ **Definition 2** (Bisimilarity ($\sim$)). *A symmetric relation R over* DyNetKAT *policies is a* bisimulation *whenever for $(p, q) \in R$ the following holds:*
If $(p, H_0, H_1) \xrightarrow{\gamma} (p', H_0', H_1')$ then exists q' s.t. $(q, H_0, H_1) \xrightarrow{\gamma} (q', H_0', H_1')$ and $(p', q') \in R$, with $\gamma ::= (\sigma, \sigma') \mid x?r \mid x!r \mid \mathbf{rcfg}(\mathbf{x}, \mathbf{r})$.

$$(\mathbf{cpol}^{\checkmark}_{\_;})\,\frac{\sigma' \in [\![p]\!](\sigma::\langle\rangle)}{(p;q,\sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (q, H, \sigma' :: H')} \qquad (\mathbf{cpol_X})\,\frac{(p, H_0, H_1) \xrightarrow{\gamma} (p', H_0', H_1')}{(X, H_0, H_1) \xrightarrow{\gamma} (p', H_0', H_1')}\, X \triangleq p$$

$$(\mathbf{cpol}_{\_\oplus})\,\frac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(p \oplus q, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')} \qquad (\mathbf{cpol}_{\oplus\_})\,\frac{(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}{(p \oplus q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}$$

$$(\mathbf{cpol}_{\_||})\,\frac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(p||q, H_0, H_0') \xrightarrow{\gamma} (p'||q, H_1, H_1')} \qquad (\mathbf{cpol}_{||\_})\,\frac{(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}{(p||q, H_0, H_0') \xrightarrow{\gamma} (p||q', H_1, H_1')}$$

$$(\mathbf{cpol}_{?})\,\frac{}{(x?p;q, H, H') \xrightarrow{x?p} (q, H, H')} \qquad (\mathbf{cpol}_{!})\,\frac{}{(x!p;q, H, H') \xrightarrow{x!p} (q, H, H')}$$

$$(\mathbf{cpol}_{!?})\,\frac{(q, H, H') \xrightarrow{x!p} (q', H, H') \quad (s, H, H') \xrightarrow{x?p} (s', H, H')}{(q||s, H, H') \xrightarrow{\mathbf{rcfg(x,p)}} (q'||s', H, H')}$$

$$(\mathbf{cpol}_{?!})\,\frac{(q, H, H') \xrightarrow{x?p} (q', H, H') \quad (s, H, H') \xrightarrow{x!p} (s', H, H')}{(q||s, H, H') \xrightarrow{\mathbf{rcfg(x,p)}} (q'||s', H, H')}$$

$\gamma ::= (\sigma, \sigma') \mid x!q \mid x?q \mid \mathbf{rcfg(x, q)}$

Figure 8 DyNetKAT: Operational Semantics

We call bisimilarity *the largest bisimulation relation.*

Two policies p and q are bisimilar *($p \sim q$) if and only if there is a bisimulation relation R such that $(p, q) \in R$.*

Semantic equivalence of NetKAT$^{-\mathbf{dup}}$ policies is preserved by DyNetKAT bisimilarity.

▶ **Proposition 3** (Semantic Layering). *Let p and q be two* NetKAT$^{-\mathbf{dup}}$ *policies. The following holds:*

$$[\![p]\!] = [\![q]\!] \text{ iff } (p; d) \sim (q; d)$$

for any DyNetKAT *policy d.*

Proof. The result follows directly according to the definition of bisimilarity and $(\mathbf{cpol}^{\checkmark}_{\_;})$ in Figure 8. ◀

Next, we introduce the restriction operator $\delta_{\mathcal{L}}(-)$ [1, 3], with $\mathcal{L}$ a set of forbidden actions ranging over $x?z$ and $x!z$ as in (1). The semantics of $\delta_{\mathcal{L}}(-)$ is:

$$(\delta)\,\frac{(p, H_0, H_1) \xrightarrow{\gamma} (p', H_0', H_1')}{(\delta_{\mathcal{L}}(p), H_0, H_1) \xrightarrow{\gamma} (\delta_{\mathcal{L}}(p'), H_0', H_1')}\, \gamma \notin \mathcal{L} \tag{2}$$

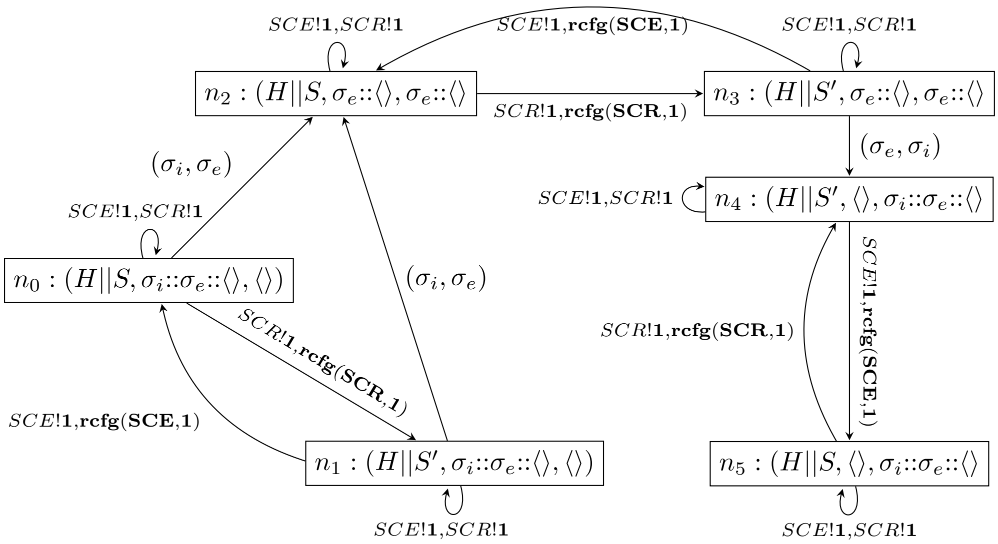


Figure 9 Stateful Firewall LTS

In practice, we use the restriction operator to force synchronous communication. For an example, consider the synchronising controllers in Figure 7. Let $\mathcal{L}$ be the set of restricted actions ranging over elements of shape $upS_i?X$, $upS_i!X$, $syn?1$ and $syn!1$. The restricted system $\delta_{\mathcal{L}}(SDN_{S1,\ldots,S6} \,||\, C_1 \,||\, C_2)$ ensures that: (1) traffic through $S2$ and $S1$ is first disabled via reconfigurations $\mathbf{rcfg}(\mathbf{upS2}, \mathbf{0})$ and $\mathbf{rcfg}(\mathbf{upS1}, \mathbf{0})$ and (2) the controllers acknowledge this deactivation via a synchronization step $\mathbf{rcfg}(\mathbf{syn}, \mathbf{1})$ before installing further flow tables for $S4$ and $S6$.

In the style of [3], we define a projection operator $\pi_n(-)$ that, intuitively, captures the first n steps of a DyNetKAT policy. Its formal semantics is:

$$(\pi)\,\frac{(p, H_0, H_1) \xrightarrow{\gamma} (p', H_0', H_1')}{(\pi_{n+1}(p), H_0, H_1) \xrightarrow{\gamma} (\pi_n(p'), H_0', H_1')} \tag{3}$$

As we shall later see, $\pi_n(-)$ is crucial for defining the so-called "Approximation Induction Principle" that enables reasoning about equivalence of recursive DyNetKAT specifications.

We further provide some additional ingredients needed to introduce the DyNetKAT axiomatization in Figure 11. First, note that our notion of bisimilarity identifies synchronization steps as in $(\mathbf{cpol}_{!?})$ and $(\mathbf{cpol}_{?!})$. At the axiomatization level, this requires introducing corresponding constants $\mathbf{rcfg}_{x,z}$ defined as:

$$(\mathbf{rcfg_{x,z}})\,\frac{}{(\mathbf{rcfg}_{x,z}; p, H_0, H_1) \xrightarrow{\mathbf{rcfg(x,z)}} (p, H_0, H_1)} \tag{4}$$

Last, but not least, we introduce the left-merge operator ($\lfloor\!\lfloor$) and the communication-merge operator ($|$) utilised for axiomatizing parallel composition. Intuitively, a process of shape $p \lfloor\!\lfloor q$ behaves like p as a first step, and then continues as the parallel composition between the remaining behaviour of p and q. A process of shape $p \mid q$ forces the synchronous communication between p and q in a first step, and then continues as the parallel composition

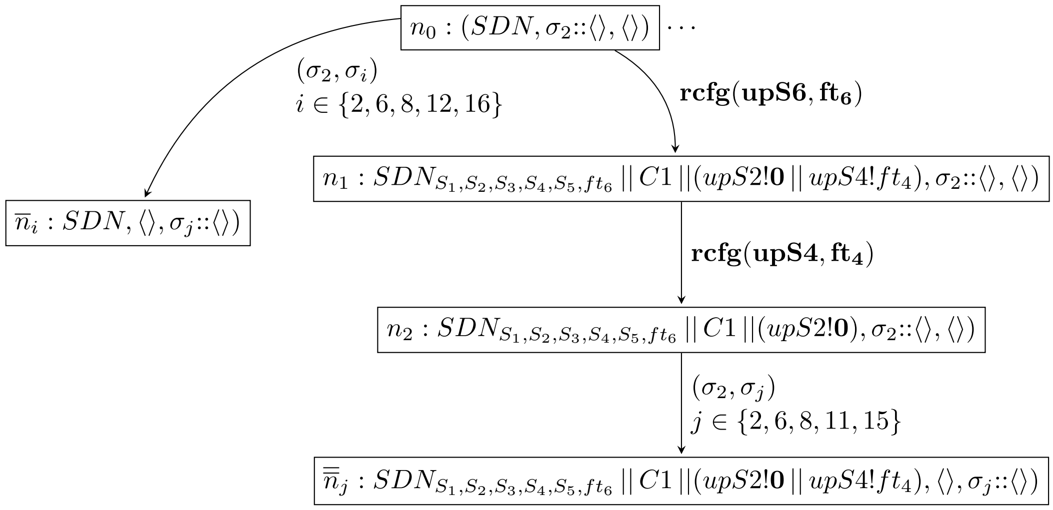


Figure 10 Independent Controllers LTS (excerpt)

between the remaining behaviours of p and q. The corresponding semantic rules are:

$$
\begin{array}{l}
(\lfloor\!\lfloor)\dfrac{(p, H_0, H_1) \xrightarrow{\gamma} (p', H_0', H_1')}{(p \lfloor\!\lfloor q, H_0, H_1)) \xrightarrow{\gamma} (p' \,||\, q, H_0', H_1')} \qquad \gamma ::= (\sigma, \sigma') \mid x!p \mid x?p \mid \mathbf{rcfg(x, p)} \\
(|^{?!})\dfrac{(p, H, H') \xrightarrow{x?r} (p', H, H') \qquad (q, H, H') \xrightarrow{x!r} (q', H, H')}{(p \mid q, H, H') \xrightarrow{\mathbf{rcfg(x,p)}} (p' \,||\, q', H, H')} \\
(|^{!?})\dfrac{(p, H, H') \xrightarrow{x!r} (p', H, H') \qquad (q, H, H') \xrightarrow{x?r} (q', H, H')}{(p \mid q, H, H') \xrightarrow{\mathbf{rcfg(x,p)}} (p' \,||\, q', H, H')}
\end{array}
\tag{5}
$$

From this point onward, we denote by DyNetKAT the extension with the operators in (2), (3) and (4):

$$
\begin{array}{lll}
N & ::= & \text{NetKAT}^{-\mathbf{dup}} \\
D_e & ::= & \bot \mid N \,;\, D \mid x?N \,;\, D_e \mid x!N \,;\, D_e \mid \mathbf{rcfg}_{x,N} \,;\, D_e \mid \\
 & & D_e \,||\, D_e \mid D_e \oplus D_e \mid \delta_{\mathcal{L}}(D_e) \mid \pi_n(D_e) \mid D_e \lfloor\!\lfloor D_e \mid D_e | D_e \mid X \\
 & & X \triangleq D_e,\ n \in \mathbb{N},\ \mathcal{L} = \{c \mid c ::= x?N \mid x!N\}
\end{array}
\tag{6}
$$

Bisimilarity is defined for DyNetKAT terms as in (6) in the natural fashion, according to the operational semantics of the new operators in (2), (3) and (4).

▶ **Lemma 3.** DyNetKAT *bisimilarity is a* congruence*.*

Proof. The result follows from the fact that the semantic rules defined in this paper comply to the congruence formats proposed in [19]. ◀

▶ **Definition 4** (Complete Tests & Assignments [2])**.** *Let $F = \{f_1, \ldots, f_n\}$ be a set of fields names with values in V_i, for $i \in \{1, \ldots, n\}$. We call* complete test *(typically denoted by α) an expression $f_1 = v_1 \cdot \ldots \cdot f_n = v_n$, with $v_i \in V_i$, for $i \in \{1, \ldots, n\}$. We call* complete assignment *(typically denoted by π) an expression $f_1 \leftarrow v_1 \cdot \ldots \cdot f_n \leftarrow v_n$, with $v_i \in V_i$, for $i \in \{1, \ldots, n\}$. We sometimes write α_π in order to denote the complete test derived from* for $p, q, r \in$ DyNetKAT and $z, y \in$ NetKAT$^{-\mathbf{dup}}$

for $a ::= z \mid x?z \mid x!z \mid \mathbf{rcfg}_{x,z}$

$$\mathbf{0}\,; p \equiv \bot \quad (A0)$$
$$(z + y)\,; p \equiv z\,; p \oplus y\,; p \quad (A1)$$
$$p \oplus q \equiv q \oplus p \quad (A2)$$
$$(p \oplus q) \oplus r \equiv p \oplus (q \oplus r) \quad (A3)$$
$$p \oplus p \equiv p \quad (A4)$$
$$p \oplus \bot \equiv p \quad (A5)$$
$$p \,||\, q \equiv q \,||\, p \quad (A6)$$
$$p \,||\, \bot \equiv p \quad (A7)$$
$$p \,||\, q \equiv p \llfloor q \oplus q \llfloor p \oplus p \mid q \quad (A8)$$
$$\bot \llfloor p \equiv \bot \quad (A9)$$
$$(a\,; p) \llfloor q \equiv a\,; (p \,||\, q) \quad (A10)$$
$$(p \oplus q) \llfloor r \equiv (p \llfloor r) \oplus (q \llfloor r) \quad (A11)$$
$$(x?z\,; p) \mid (x!z\,; q) \equiv \mathbf{rcfg}_{x,z}\,; (p \,||\, q) \quad (A12)$$
$$(p \oplus q) \mid r \equiv (p \mid r) \oplus (q \mid r) \quad (A13)$$
$$p \mid q \equiv q \mid p \quad (A14)$$
$$p \mid q \equiv \bot \ [owise] \quad (A15)$$

for $at ::= \alpha \cdot \pi \mid x?z \mid x!z \mid \mathbf{rcfg}_{x,z}$:

$$\delta_{\mathcal{L}}(\bot) \equiv \bot \quad (\delta_\bot)$$
$$\delta_{\mathcal{L}}(at\,; p) \equiv at\,; \delta_{\mathcal{L}}(p) \text{ if } at \notin \mathcal{L} \quad (\delta_;)$$
$$\delta_{\mathcal{L}}(at\,; p) \equiv \bot \text{ if } at \in \mathcal{L} \quad (\delta_;^\bot)$$
$$\delta_{\mathcal{L}}(p \oplus q) \equiv \delta_{\mathcal{L}}(p) \oplus \delta_{\mathcal{L}}(q) \quad (\delta_\oplus)$$

for $n \in \mathbb{N}$:

$$\pi_0(p) \equiv \bot \quad (\Pi_0)$$
$$\pi_n(\bot) \equiv \bot \quad (\Pi_\bot)$$
$$\pi_{n+1}(at\,; p) \equiv at\,; \pi_n(p) \quad (\Pi_;)$$
$$\pi_n(p \oplus q) \equiv \pi_n(p) \oplus \pi_n(q) \quad (\Pi_\oplus)$$

$$p \equiv q \text{ if } \forall n \in \mathbb{N} : \pi_n(p) \equiv \pi_n(q) \quad (AIP)$$

E_{NK}

Figure 11 The axiom system E_{DNK} (including E_{NK})

the complete assignment π by replacing all $f_i \leftarrow v_i$ in π with $f_i = v_i$; symmetrically for π_α. Additionally, we sometimes write σ_α to denote the network packet whose fields are assigned the corresponding values in α; symmetrically for σ_π.

In Figure 11, we introduce E_{DNK} – the axiom system of DyNetKAT, including the NetKAT axiomatization E_{NK}. Most of the axioms in Figure 11 comply to the standard axioms of parallel and communicating processes [3], where, intuitively, $\oplus$ plays the role of non-deterministic choice, ; resembles sequential composition and $\bot$ is a process that deadlocks.

For instance, axioms $(A2) - (A5)$ encode the ACI properties of $\oplus$ together with the fact that $\bot$ is the neutral element.

Axioms $(A8) - (A15)$ define parallel composition $(||)$ in terms of left-merge $(\llfloor)$ and communication-merge $(\,|\,)$ in the standard fashion. Additionally, $(A12)$ "pin-points" a communication step via the newly introduced constants of form $\mathbf{rcfg}_{x,z}$. An interesting axiom is $(A7)$: $p \,||\, \bot \equiv p$ which, intuitively, states that if one network component fails, then the whole system continues with the behaviour of the remaining components. This is a departure from the approach in [13], where recovery is not possible in case of a component's failure; i.e., $e \,||\, 0 \equiv 0$.

Axiom $(A0)$ states that if the current packet is dropped as a result of the unsuccessful evaluation of a NetKAT policy, then the continuation is deadlocked. $(A1)$ enables mapping the non-deterministic choice at the level of NetKAT to the setting of DyNetKAT.

The axioms encoding the restriction operator $\delta_{\mathcal{L}}(-)$ and the projection operator $\pi_n(-)$ are defined in the standard fashion, on top of DyNetKAT normal forms later defined in this section. Intuitively, normal forms are defined inductively, as sums of complete tests and complete assignments $\alpha \cdot \pi$, or communication steps $x?q$, $x!q$ and $\mathbf{rcfg}_{x,q}$, followed by arbitrary DyNetKAT policies.

Last, but not least, (AIP) corresponds to the so-called "Approximation Induction Principle", and it provides a mechanism for reasoning on the equivalence of recursive behaviours, up to a certain limit denoted by n.

3.1.1 Soundness and Completeness

In what follows, we show that the axiom system E_{DNK} is sound and ground-complete with respect to DyNetKAT bisimilarity.

We proceed by first defining a notion of normal forms of DyNetKAT terms, together with a notion of guardedness and a statement about the branching finiteness of guarded DyNetKAT processes.

▶ **Lemma 5** (NetKAT$^{-\mathbf{dup}}$ Normal Forms)**.** *We call a* NetKAT$^{-\mathbf{dup}}$ *policy q in* normal form *(n.f.) whenever q is of shape $\Sigma_{\alpha\cdot\pi\in\mathcal{A}}\alpha\cdot\pi$ with $\mathcal{A} = \{\alpha_i \cdot \pi_i \mid i \in I\}$. For every* NetKAT$^{-\mathbf{dup}}$ *policy p there exists a* NetKAT$^{-\mathbf{dup}}$ *policy q in n.f. such that $E_{NK} \vdash p \equiv q$.*

Proof. The result follows by Lemma 4 in [2], stating that:

$$[\![p]\!] = \bigcup_{x\in G(p)} [\![x]\!] \tag{7}$$

where $G(p)$ defines the language model of NetKAT terms. Let A be the set of all complete tests, and Π be the set of all complete assignments. Similarly to [2], we consider network packets with values in finite domains. Consequently, A and Π are finite. In [2], $G(p)$ is defined as a set with elements in $A\cdot(\Pi\cdot\mathbf{dup})^*\cdot\Pi$. Recall that, in our setting, we work with the **dup**-free fragment of NetKAT. Hence, $G(p)$ is a finite set of shape $G = \{\alpha_i \cdot \pi_i \mid i \in I, \alpha_i \in A, \pi_i \in \Pi\}$. Based on the definition of $[\![-]\!]$ and (7) it follows that:

$$[\![p]\!] = [\![\Sigma_{\alpha\cdot\pi\in G}\alpha\cdot\pi]\!] \tag{8}$$

Therefore, by the completeness of NetKAT, it holds that: $E_{NK} \vdash p \equiv \Sigma_{\alpha\cdot\pi\in G}\alpha\cdot\pi$. In other words, p can be reduced to a term in n.f. ◀

▶ **Definition 6** (DyNetKAT Normal Forms)**.** *We call a* DyNetKAT *policy in* normal form *(n.f.) if it is of shape*

$$\Sigma^{\oplus}_{i\in I}(\alpha_i \cdot \pi_i); d_i \oplus \Sigma^{\oplus}_{j\in J} c_j; d_j \ (\oplus\bot)$$

where d_i, d_j range over DyNetKAT *policies and $c_j ::= x?q \mid x!q \mid \mathbf{rcfg}_{x,q}$ with q denoting terms in* NetKAT$^{-\mathbf{dup}}$*.*

▶ **Definition 7** (Guardedness)**.** *A* DyNetKAT *policy p is* guarded *if and only if all occurrences of all variables X in p are* guarded*. An occurrence of a variable X in a policy p is* guarded *if and only if (i) p has a subterm of shape $p'; t$ such that either p' is variable-free, or all the occurrences of variables Y in p' are guarded, and X occurs in t, or (ii) if p is of shape $y?X; t$, $y!X; t$ or $\mathbf{rcfg}_{X,t}$.*

▶ **Lemma 8** (Branching Finiteness)**.** *All guarded* DyNetKAT *policies are finitely branching.*

▶ **Lemma 9** (DyNetKAT Normalization)**.** *E_{DNK} is normalising for* DyNetKAT*. In other words, for every guarded* DyNetKAT *policy p there exists a* DyNetKAT *policy q in n.f. such that $E_{DNK} \vdash p \equiv q$.*

Proof. The proof follows from Lemma 5 and $(A1) : (z + y); p \equiv z; p \oplus y; p$ in a standard fashion, by structural induction.
Base cases.

- $p \triangleq \bot$ trivially holds

- $p \triangleq q; d$ with q a NetKAT$^{-\mathbf{dup}}$ term holds by Lemma 5 and $(A1)$
- $p \triangleq c; d$ with $c ::= x?q \mid x!q \mid \mathbf{rcfg}_{x,q}$ trivially holds

Induction step.

- $p \triangleq p_1 \oplus p_2$ $\quad$ $p \triangleq X$ - case discarded, as p is not guarded
- $p \triangleq p_1 \lfloor\lfloor p_2$ $\quad$ $p \triangleq \pi_n(')$
- $p \triangleq p_1 \mid p_2$ $\quad$ $p \triangleq \delta_{\mathcal{L}}(p')$
- $p \triangleq p_1 \,||\, p_2$

All items above follow by the axiom system E_{DNK} and the induction hypothesis, under the assumption that p_1, p_2 and p' are guarded. ◀

For simplicity, in what follows, we assume that DyNetKAT policies are guarded.

▶ **Lemma 10** (Soundness of $E_{\text{DyNetKAT}\backslash AIP}$)**.** *Let $E_{\text{DyNetKAT}\backslash AIP}$ stand for the axiom system E_{DNK} in Figure 11, without the axiom (AIP). $E_{\text{DyNetKAT}\backslash AIP}$ is sound for* DyNetKAT *bisimilarity.*

Proof. The proof reduces to showing that for all p, q DyNetKAT policies, the following holds: If $E_{\text{DyNetKAT}\backslash AIP} \vdash p \equiv q$ then $p \sim q$. This is proven in a standard fashion, by case analysis on transitions of shape

$$(p, H_0, H_0') \xrightarrow{\gamma} (q, H_1, H_1')$$

with $\gamma ::= (\sigma, \sigma') \mid x?n \mid x!n \mid \mathbf{rcfg(x, n)}$, according to the semantic rules in Figure 8, (2), (3), (4) and (5).

For an example, consider $(A1)$ and $(A12)$ in Figure 11; the proof of soundness for these axioms are given in the following. The soundness proofs for the rest of the axioms are provided in Appendix A.

- Axiom under consideration:

$$(z + y) \,; p \equiv z \,; p \oplus y \,; p \quad (A1) \tag{9}$$

for $z, y \in$ NetKAT$^{-\mathbf{dup}}$ and $p \in$ DyNetKAT. Consider an arbitrary but fixed network packet σ, let $S_z \triangleq [\![z]\!](\sigma{::}\langle\rangle)$, $S_y \triangleq [\![y]\!](\sigma{::}\langle\rangle)$ and $S_{zy} \triangleq [\![z + y]\!](\sigma{::}\langle\rangle)$. According to the semantic rules of DyNetKAT, the derivations of the term $(z + y) \,; p$ are as follows:

(a)

$$\text{For all } \sigma' \in S_{zy}: \quad (\mathbf{cpol}^{\checkmark}_{\_;})\,\frac{}{((z + y); p, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p, H, \sigma' :: H')}$$

Accordingly, the derivations of the term $z \,; p \oplus y \,; p$ are as follows:

(b)

$$\text{For all } \sigma' \in S_z: \quad (\mathbf{cpol}^{\checkmark}_{\_;})\,\frac{}{(z; p, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p, H, \sigma' :: H')}$$
$$(\mathbf{cpol}_{\_\oplus})\,\frac{}{(z; p \oplus y; p, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p, H, \sigma' :: H')}$$

(c)

$$\text{For all } \sigma' \in S_y: \quad (\mathbf{cpol}^{\checkmark}_{\_;})\,\frac{}{(y; p, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p, H, \sigma' :: H')}$$
$$(\mathbf{cpol}_{\oplus\_})\,\frac{}{(z; p \oplus y; p, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p, H, \sigma' :: H')}$$

As demonstrated in (a) and (b), (c), both of the terms $(z + y)\,;p$ and $z\,;p \oplus y\,;p$ initially only afford a transition of shape (σ, σ') and they converge into the same expression after taking that transition:

$$((z + y); p, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p, H, \sigma' :: H') \tag{10}$$

$$(z\,; p \oplus y\,; p, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p, H, \sigma' :: H') \tag{11}$$

In the case of the term $(z + y)\,;p$, the possible values for the σ' ranges over S_{zy}. Whereas for the term $(z + y)\,;p$, the possible values for the σ' ranges over $S_z \cup S_y$. However, observe that S_{zy} is equal to $S_z \cup S_y$:

$$S_{zy} = [\![z + y]\!](\sigma{::}\langle\rangle) \qquad \text{(Definition of } S_{zy}) \tag{12}$$

$$= [\![z]\!](\sigma{::}\langle\rangle) \cup [\![y]\!](\sigma{::}\langle\rangle) \qquad \text{(Definition of } +) \tag{13}$$

$$= S_z \cup S_y \qquad \text{(Definition of } S_z \text{ and } S_y) \tag{14}$$

Hence, it is straightforward to conclude that the following holds:

$$((z + y)\,; p) \sim (z\,; p \oplus y\,; p) \tag{15}$$

- Axiom under consideration:

$$(x?z\,; p) \mid (x!z\,; q) \equiv \mathbf{rcfg}_{x,z}\,;(p \,||\, q) \quad (A12) \tag{16}$$

for $p, q \in$ DyNetKAT. The derivations of the term $(x?z\,; p) \mid (x!z\,; q)$ are as follows:

(a)

$$\dfrac{(\mathbf{cpol}_?)\dfrac{}{(x?z; p, H, H') \xrightarrow{x?z} (p, H, H')} \qquad (\mathbf{cpol}_!)\dfrac{}{(x!z; q, H, H') \xrightarrow{x!z} (q, H, H')}}{((x?z\,; p) \mid (x!z\,; q), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p \,||\, q, H, H')}\,(|^{?!})$$

The derivations of the term $\mathbf{rcfg}_{x,z}\,;(p \,||\, q)$ are as follows:

(b)

$$(\mathbf{rcfg_{x,z}})\dfrac{}{(\mathbf{rcfg}_{x,z}\,;(p \,||\, q), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p \,||\, q, H, H')}$$

As demonstrated in (a) and (b), both of the terms $(x?z\,; p) \mid (x!z\,; q)$ and $\mathbf{rcfg}_{x,z}\,;(p \,||\, q)$ initially only afford the transition $\mathbf{rcfg(x,z)}$ and they converge into the same expression after taking that transition:

$$((x?z\,; p) \mid (x!z\,; q), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p \,||\, q, H, H') \tag{17}$$

$$(\mathbf{rcfg}_{x,z}\,;(p \,||\, q), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p \,||\, q, H, H') \tag{18}$$

Hence, it is straightforward to conclude that the following holds:

$$((x?z\,; p) \mid (x!z\,; q)) \sim (\mathbf{rcfg}_{x,z}\,;(p \,||\, q)). \tag{19}$$

◀

▶ **Lemma 11** (Soundness of AIP). *The Approximation Induction Principle (AIP) is sound for* DyNetKAT *bisimilarity.*

Proof. The proof is close to the one of Theorem 2.5.8 in [3] and uses the branching finiteness property of DyNetKAT policies in Lemma 8. Assume two DyNetKAT policies p, p' such that

$$\forall n \in \mathbb{N} : \pi_n(p) \equiv \pi_n(p') \tag{20}$$

By Lemma 10 it follows that

$$\forall n \in \mathbb{N} : \pi_n(p) \sim \pi_n(p') \tag{21}$$

We want to prove that $p \sim p'$. The idea is to build a bisimulation relation R such that $(p, p') \in R$. We define R as follows:

$$R = \{(t, t') \mid \forall n \in \mathbb{N} : \pi_n(t) \sim \pi_n(t')\} \tag{22}$$

Without loss of generality, assume that p and p' are in n.f. Assume $(p, p') \in R$ and

$$(p, H_0, H_0') \xrightarrow{\gamma} (p_1, H_1, H_1') \tag{23}$$

Next, for all $n > 0$, define

$$S_n = \{p_1' \mid (p', H_0, H_0') \xrightarrow{\gamma} (p_1', H_1, H_1') \text{ and } \pi_n(p_1) \sim \pi_n(p_1')\} \tag{24}$$

The following hold:

1. $S_1 \supseteq S_2 \ldots$ as if $\pi_{n+1}(p) \sim \pi_{n+1}(p')$ then $\pi_n(p_1) \sim \pi_n(p_1')$. The latter is a straightforward result derived according to the definition of $\sim$ and the semantics of $\pi_n(-)$, under the assumption that p, p' are in n.f.
2. $S_n \neq \emptyset$ for all $n \geq 1$ since $\pi_{n+1}(p) \sim \pi_{n+1}(p')$ by (21) and $(p, H_0, H_0') \xrightarrow{\gamma} (p_1, H_1, H_1')$ according to (23)
3. S_n is finite, for all $n \in \mathbb{N}$, as p' is finitely branching according to Lemma 8

Hence, the sequence $S_1, S_2, \ldots$ remains constant from some n onward and $\cap_{n \geq 0} S_n \neq \emptyset$. Let $p_1' \in \cap_{n \geq 0} S_n$. It holds that:

- $(p', H_0, H_0') \xrightarrow{\gamma} (p_1', H_1, H_1')$
- $(p_1, p_1') \in R$ by the definition of R and S_n

Symmetrically to (23), assume $(p, p') \in R$ and $(p', H_0, H_0') \xrightarrow{\gamma} (p_1', H_1, H_1')$. By following a similar reasoning, we can show that:

- $(p, H_0, H_0') \xrightarrow{\gamma} (p_1, H_1, H_1')$
- $(p_1, p_1') \in R$ by the definition of R and S_n

Hence, R is a bisimulation relation and $p \sim p'$. ◀

▶ **Theorem 4** (Soundness & Completeness). *E_{DNK} is sound and ground-complete for* DyNetKAT *bisimilarity.*

Proof. Soundness: if $E_{DNK} \vdash p \equiv q$ then $p \sim q$, follows from Lemma 10 and Lemma 11.

Completeness: if $p \sim q$ then $E_{DNK} \vdash p \equiv q$, is shown as follows. Without loss of generality, assume p and q are in n.f., according to Lemma 9. We want to show that:

$$\begin{array}{rcl} p & \equiv & q \oplus p \\ q & \equiv & p \oplus q \end{array} \tag{25}$$

which, by ACI of $\oplus$ implies $p \equiv q$. This reduces to showing that every summand of p is a summand of q and vice-versa. We first argue that every summand of p is a summand of q. The reasoning is by structural induction.

Base case.

- $p \triangleq \bot$. It holds by the hypothesis $p \sim q$ that $q \triangleq \bot$.

Induction step.

- $p \triangleq ((\alpha \cdot \pi); p') \oplus p''$. Then, $(p, \sigma_\alpha :: H, H') \xrightarrow{(\sigma_\alpha, \sigma_\pi)} (p', H, \sigma_\pi :: H')$ implies by the hypothesis $p \sim q$ that $(q, \sigma_\alpha :: H, H') \xrightarrow{(\sigma_\alpha, \sigma_\pi)} (q', H, \sigma_\pi :: H')$ and $p' \sim q'$. Recall that q is in n.f.; hence, by the shape of the semantic rules in Figure 8 it holds that $q \triangleq ((\alpha \cdot \pi); q') \oplus q''$. By the induction hypothesis, it holds that $p' \equiv q'$ hence, $(\alpha \cdot \pi); p'$ is a summand of q as well.
- Cases $p \triangleq (c; p') \oplus p''$ with $c ::= x?n \mid x!n \mid \mathbf{rcfg}_{x,n}$ follow in a similar fashion.

Hence, $p \equiv q \oplus p$ holds. The symmetric case $q \equiv p \oplus q$ follows the same reasoning. ◀

4 A Framework for Safety

In this section we provide a language for specifying safety properties of DyNetKAT networks, together with a procedure for reasoning about safety in an equational fashion. Intuitively, safety properties enable specifying undesired network behaviours.

▶ **Definition 12** (Safety Properties - Syntax). *Let $\mathcal{A}$ be an alphabet over letters of shape $\alpha \cdot \pi$ and $\mathbf{rcfg}(\mathbf{x}, \mathbf{p})$, with α and π ranging over complete tests and assignments as in Definition 4, and $\mathbf{rcfg}(\mathbf{x}, \mathbf{p})$ ranging over reconfiguration actions. A* safety property *prop is defined as:*

$$\begin{array}{rcll} act & ::= & \alpha \cdot \pi \mid \mathbf{rcfg}_{x,p} & (\text{where } \alpha \cdot \pi,\ \mathbf{rcfg}_{x,p} \in \mathcal{A}) \\ regexp & ::= & act \mid regexp + regexp \mid regexp \cdot regexp & \\ prop & ::= & [regexp]false & \end{array}$$

The intuition behind Definition 12 is as follows. A safety property specification *prop* is satisfied whenever the behaviour encoded by *regexp* cannot be observed within the network. Regular expressions *regexp* are defined with respect to actions *act*: a flow of shape $\alpha \cdot \pi$ is the observable behaviour of a (NetKAT$^{-\mathbf{dup}}$) policy transforming a packet encoded by α into α_π, whereas $\mathbf{rcfg}_{x,p}$ corresponds to a reconfiguration step in a network. Recursively, a sum of regular expressions $regexp_1 + regexp_2$ encodes the union of the two behaviours, a concatenation of regular expressions $regexp_1 \cdot regexp_2$ encodes the behaviour of $regexp_1$ followed by the behaviour of $regexp_2$.

▶ **Definition 5** (Head Normal Forms for Safety). *Let $\mathcal{A}$ be an alphabet over letters of shape $\alpha \cdot \pi$ and $\mathbf{rcfg}(\mathbf{x}, \mathbf{p})$, with α and π ranging over complete tests and assignments as in Definition 4, and $\mathbf{rcfg}(\mathbf{x}, \mathbf{p})$ ranging over reconfiguration actions. We write w, w' for (non-empty) words* *with letters in $\mathcal{A}$ (i.e., $w, w' \in \mathcal{A}^*$) and $\mid w \mid$ for the length of w. We write $w' \preceq w$ whenever w' is a prefix of w (including w).*

Let r be a regular expression (regexp) as in Definition 12. We call head normal form *of r, denoted by $hnf(r)$, the sum of words obtained by distributing $\cdot$ over $+$ in r, in the standard fashion:*

$$\begin{array}{rcl} hnf(a) & \triangleq & a \quad (a \in \mathcal{A}) \\ hnf(w) & \triangleq & w \quad (w \in \mathcal{A}^*) \\ hnf(r_1 + r_2) & \triangleq & hnf(r_1) + hnf(r_2) \\ hnf(r_1 \cdot (r_2 + r_3)) & \triangleq & hnf(r_1 \cdot r_2) + hnf(r_1 \cdot r_3) \\ hnf((r_1 + r_2) \cdot r_3) & \triangleq & hnf(r_1 \cdot r_3) + hnf(r_1 \cdot r_3) \\ hnf(r' \cdot (r_1 + r_2) \cdot r'') & \triangleq & hnf(r' \cdot r_1 \cdot r'') + hnf(r' \cdot r_2 \cdot r'') \end{array}$$

Next, we give the formal semantics of safety properties.

▶ **Definition 13** (Safety Properties - Semantics). *Let Prop stand for the set of all properties as in Definition 12. The semantic map $[\![-]\!] : Prop \to \text{DyNetKAT}$ associates to each safety property in Prop a* DyNetKAT *expression as follows.*

Let Θ be the DyNetKAT *policy (in normal form) encoding all possible behaviours over $\mathcal{A}$:*

$$\Theta \triangleq \Sigma^{\oplus}_{\alpha\cdot\pi\in\mathcal{A}}(\alpha \cdot \pi; \bot \oplus \alpha \cdot \pi; \Theta) \;\oplus\; \Sigma^{\oplus}_{\mathbf{rcfg}_{x,p}\in\mathcal{A}}(\mathbf{rcfg}_{x,p}; \bot \oplus \mathbf{rcfg}_{x,p}; \Theta)$$

Then:

$$(c_1) \quad [\![\, [\Sigma_{\substack{i \in I \\ w_i \in A^*}} w_i] false \,]\!] \;\triangleq\; \Sigma^{\oplus}_{\substack{w \in \mathcal{A}^* \\ \mid w \mid < M \\ \forall i \in I : w_i \not\preceq w}} \overline{w}; \bot \;\;\oplus\;\; \Sigma^{\oplus}_{\substack{w \in \mathcal{A}^* \\ \mid w \mid = M \\ \forall i \in I : w_i \not\preceq w}} (\overline{w}; \bot \oplus \overline{w}; \Theta)$$

$$(c_2) \qquad [\![[r] false]\!] \;\triangleq\; [\![[hnf(r)] false]\!] \qquad \text{[otherwise]}$$

such that M is the length of the longest word w_i, with $i \in I$, and $\overline{w}$ is a DyNetKAT*-compatible term obtained from w where all letters have been separated by* ; *and inductively defined in the obvious way:*

$$\begin{array}{rcl} \overline{a} & \triangleq & a \quad (a \in \mathcal{A}) \\ \overline{a \cdot w} & \triangleq & a; \overline{w} \quad (a \in \mathcal{A}, w \in \mathcal{A}^*) \end{array}$$

The semantic map $[\![-]\!] : Prop \to \text{DyNetKAT}$ is defined in accordance with the intuition provided in the beginning of this section. For instance, as shown in (c_1), if none of the sequences of steps w_i can be observed in the system, then the associated DyNetKAT term prevents the immediate execution of all w_i. Typically, safety analysis is reduced to reachability analysis. Intuitively, in our context, a safety property is violated whenever the network system under analysis displays a (finite) execution that is not in the behaviour of the property. Thus, the semantic map in Definition 13 is based on traces (or words in $\mathcal{A}^*$) and is not sensitive to branching; see the use of head normal forms in (c_2).

With these ingredients at hand, we can reason about the satisfiability of safety properties in an equational fashion.

▶ **Definition 14** (E^{tr}_{DNK}). *Let E^{tr}_{DNK} stand for the equational axioms in Figure 11, including the additional axiom that enables switching from the context of bisimilarity to trace equivalence of* DyNetKAT *policies, namely:*

$$p; (q \oplus r) \equiv p; q \oplus p; r \qquad (A_{16}) \tag{26}$$

▶ **Definition 15** (Safe Network Systems)**.** *Assume a specification given as the safety formula s and a network system implemented as the* DyNetKAT *policy i. We say that the network is* safe *whenever the following holds:*

$$E^{tr}_{DNK} \vdash [\![s]\!] \oplus i \equiv [\![s]\!] \tag{27}$$

In words: checking whether i satisfies s reduces to checking whether the trace behaviour of i is included into that of s.

4.1 Sugars for Safety

In this section we introduce a version of safety properties extended with negated actions ($\neg(\alpha{\cdot}\pi)$ and, respectively, $\neg\mathbf{rcfg}_{x,p}$), the *true* construct and repetitions (r^n), equally expressive but enabling more concise property specifications.

▶ **Definition 16** (Safety Properties - Extended Syntax)**.** *Let $\mathcal{A}$ be an alphabet over letters of shape $\alpha \cdot \pi$ and $\mathbf{rcfg}_{x,p}$, with α and π ranging over complete tests and assignments as in Definition 4, and $\mathbf{rcfg}_{x,p}$ ranging over reconfiguration actions. Safety properties are extended in the following fashion:*

$$\begin{array}{rcl} act_e & ::= & \alpha \cdot \pi \mid \mathbf{rcfg}_{x,p} \mid \neg act_e \qquad (\text{with } \alpha \cdot \pi,\, \mathbf{rcfg}_{x,p} \in \mathcal{A}) \\ regexp_e & ::= & true \mid act_e \mid regexp_e + regexp_e \mid regexp_e \cdot regexp_e \mid (regexp_e)^n \quad (\text{with } n \geq 1) \\ prop_e & ::= & [regexp_e]false \end{array}$$

Intuitively, a property of shape $[\neg a]false$, with $a \in \mathcal{A}$, states that the system cannot do anything apart from a as a first step. The property $[true]false$ states that no action can be observed in the network, whereas $[r^n]false$ encodes the repeated application of r for n times.

Let *Reg* and, respectively, Reg_e denote the set of regular expressions *regexp* in Definition 12 and, respectively, the set of regular expressions $regexp_e$ in Definition 16. The "desugaring" function defining the regular equivalent of the extended safety properties is defined as follows:

$$\begin{array}{rcl} & & ds : Reg_e \to Reg \\ ds(true) & \triangleq & \Sigma_{a \in \mathcal{A}} a \\ ds(\neg(\alpha \cdot \pi)) & \triangleq & \displaystyle\sum_{\substack{\alpha_i \cdot \pi_i \in \mathcal{A} \\ \alpha_i \neq \alpha \\ or \\ \pi_i \neq \pi}} \alpha_i \cdot \pi_i \\ ds(\neg\mathbf{rcfg}_{x,p}) & \triangleq & \displaystyle\sum_{\substack{\mathbf{rcfg}_{y,q} \in \mathcal{A} \\ \mathbf{rcfg}_{y,q} \neq \mathbf{rcfg}_{x,p}}} \mathbf{rcfg}_{y,q} \\ ds(r^n) & \triangleq & ds(\underbrace{r \cdot r \cdot \ldots \cdot r}_{n \text{ times}}) \\ ds(r_1 \cdot r_2) & \triangleq & ds(r_1) \cdot ds(r_2) \quad \text{if } r_1 \cdot r_2 \notin Reg \\ ds(r_1 + r_2) & \triangleq & ds(r_1) + ds(r_2) \quad \text{if } r_1 + r_2 \notin Reg \\ ds(r) & \triangleq & r \quad [\text{otherwise}] \end{array}$$

The (overloaded) semantic map $[\![-]\!] : Prop_e \to$ DyNetKAT is defined as expected:

$$[\![\, [r_e]false \,]\!] \triangleq [\![\, [ds(r_e)]false \,]\!]$$

For an example, consider the distributed controllers in Figure 2 and the corresponding encoding in Figure 6. Recall that reaching $H4$ from $S2$ is considered a breach in the system. This entails the safety formulae s_n defined as $[(true)^n \cdot (\alpha \cdot \pi)]false$, for $n \in \mathbb{N}$, $\alpha \triangleq (port = 2)$ and $\pi \triangleq (port \leftarrow 15)$. In words: no matter what sequence of events (of length n) is executed,

$\alpha \cdot \pi$ cannot happen as the next step. Therefore, checking whether the network is safe reduces to checking, for all $n \in \mathbb{N}$:

$$E^{tr}_{DNK} \vdash [\![s_n]\!] \oplus SDN \equiv [\![s_n]\!] \tag{28}$$

Note that, for a fixed n, the verification procedure resembles bounded model checking [4].

5 Implementation

In Section 4 we introduced a notion of safety for DyNetKAT and provided a mechanism for reasoning about safety in an equational fashion, by exploiting DyNetKAT trace semantics. To this end, we search for traces that violate the safety property, i.e., we turn the equational reasoning about safety into checking for reachability properties of shape $s \triangleq \langle regexp \rangle true$; for an implementation i, this is achieved by checking the following equation using our axiomatization: $E^{tr}_{DNK} \vdash i \oplus [\![s]\!] \equiv i$.

We developed a prototype tool, called DyNetiKAT, based on Maude [7] and Python [23], for checking the aforementioned equation. We build upon the reachability checking method in NetKAT [2]. For a reminder: we state that *out* is reachable from *in*, in the context of a switch policy p and topology t, whenever the following property is satisfied: $in \cdot (p \cdot t)^* \cdot out \not\equiv \mathbf{0}$ (and vice-versa). The inputs to our tool are a DyNetKAT program p, a list of input predicates in, a list of output predicates out, and the equivalences that describe the desired properties. For an example, consider the stateful firewall in Figure 1 and the corresponding encoding in Figure 5. Consider that we have the input predicates $in \triangleq [port = int, port = ext]$. We would like to check if packets at port int can arrive at port ext before and after reconfiguration events, and packets at port ext can arrive at port int only after a proper reconfiguration. This is achieved by analysing the step by step behaviour of DyNetKAT terms in normal form via a set of operators $head(D)$, and $tail(D, R)$, where R is a set of terms of shape $\mathbf{rcfg}_{X,N}$. Intuitively, the operator $head(D)$ returns a NetKAT policy which represents the current configuration in the input D, and the operator $tail(D, R)$ returns a DyNetKAT policy which represents the configurations in the input D that appear after the events in R.

For the firewall example, the analysis reduces to defining the output predicates $out \triangleq [port = ext, port = int]$, and the following properties:

$$in(0) \cdot head(p) \cdot out(0) \not\equiv \mathbf{0} \tag{29}$$

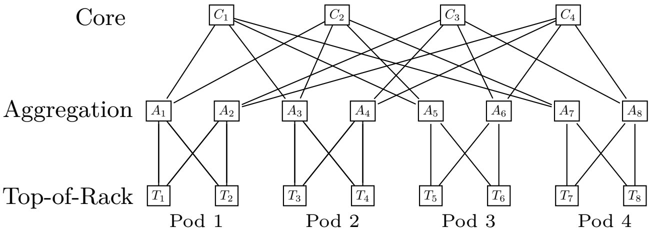
$$in(0) \cdot head(tail(p, \{\mathbf{rcfg}_{secConReq,\mathbf{1}}\})) \cdot out(0) \not\equiv \mathbf{0} \tag{30}$$

$$in(1) \cdot head(p) \cdot out(1) \equiv \mathbf{0} \tag{31}$$

$$in(1) \cdot head(tail(p, \{\mathbf{rcfg}_{secConReq,\mathbf{1}}\})) \cdot out(1) \not\equiv \mathbf{0} \tag{32}$$

Intuitively, the equivalences in (29) and (30) express that packets at port int are able to reach to port ext in the current configuration and in the configuration after the synchronization on the channel $secConReq$. The equivalence in (31) expresses that packets at port ext are not able to reach to port int in the initial configuration and (32) expresses that the configuration after the synchronization on the channel $secConReq$ allows this flow.

We performed experiments on the FatTree topologies, which are most commonly used in data centers, to evaluate the performance of our implementation. A FatTree is a hierarchical tree which typically consists of 3 layers: core, aggregation and top-of-rack (ToR). The switches at each level contain a number of redundant links to the switches at the next upper level. The groups of ToR switches and their corresponding aggregation switches are called pods. In Figure 12 (left) we illustrate a FatTree topology with 4 pods. For analyzing scalability,



# of Switches	Time (s)
7	0.12
36	1.45
99	19.87
208	154.55
375	690.23
612	3085.90

Figure 12 A FatTree Topology and Results of FatTree Experiments

we generated 6 FatTree topologies that grow in size and achieve a maximum size of 612 switches. We checked two properties on these topologies and assessed the time performance of our tool. We first computed a shortest path forwarding policy between all pairs of ToR switches in the networks and in these forwarding policies we enforced that for certain two ToR switches T_a and T_b that reside in different pods, T_a is initially not able to communicate with T_b. Accordingly, the first property that we considered is to check if T_b is reachable from T_a in the initial configuration of the network. Then, in order to check a dynamic property we considered a scenario where in an updated configuration of the network, T_b becomes accessible to T_a. In accordance with this scenario the second property that we considered is to check if T_b is reachable from T_a after a proper reconfiguration. The experiments were conducted on a computer running Ubuntu 18.04 LTS with 8 core 3.7GHz AMD Ryzen 7 2700x processors and 32 GB RAM. The results of these experiments are displayed in Figure 12 (right). The results indicate that for relatively small networks with less than 100 switches, a result is obtained in less than 20 seconds. For larger networks with sizes up to 375 switches, a result is obtained in less than 12 minutes. The experiment which contained 612 switches took the longest time with approximately 51 minutes.

In order to be able to compare our technique with another verification method, we also aimed to perform an analysis based on explicit state model checking. For this purpose, we devised an operational semantics for NetKAT and implemented it in Maude along with the operational semantics of DyNetKAT. However, this method immediately failed at scaling even for small networks, hence, we did not perform further analysis by using this method.

DyNetiKAT is available for download at: `https://github.com/hcantunc/DyNetiKAT`.

6 Conclusions

We developed a language, called DyNetKAT for modelling and reasoning about dynamic reconfigurations in Software Defined Networks. Our language builds upon the concepts, syntax, and semantics of NetKAT and hence, provides a modular extension and makes it possible to reuse the theory and tools of NetKAT. We define a formal semantics for our language and provide a sound and ground-complete axiomatization. We exploit our axiomatization to analyse reachability properties of dynamic networks and show that our approach is indeed scalable to networks with hundreds of switches.

Our language builds upon the assumption that control plane updates interleave with data plane packet processing in such a way that each data plane packet sees one set of flow tables throughout their flight in the network. This assumption is inspired by the framework put forward by Reitblatt et al. [21] and is motivated by the requirement to design a modular extension on top of NetKAT. However, we have experimented with a much smaller-stepped semantics in which the control plane updates can have a finer interleaving with in-flight packet moves. This alternative language breaks the hierarchy with NetKAT and a naive treatment of this alternative semantics will lead to much larger state-spaces. We would like to investigate this small-step semantics and efficient analysis techniques for it further.

References

1 Luca Aceto, Bard Bloom, and Frits W. Vaandrager. Turning SOS rules into equations. *Inf. Comput.*, 111(1):1–52, 1994. doi:10.1006/inco.1994.1040.

2 Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. NetKAT: semantic foundations for networks. In Suresh Jagannathan and Peter Sewell, editors, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 113–126. ACM, 2014. doi:10.1145/2535838.2535862.

3 Jos C. M. Baeten and W. P. Weijland. *Process algebra*, volume 18 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 1990.

4 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

5 Ryan Beckett, Michael Greenberg, and David Walker. Temporal netkat. In Chandra Krintz and Emery Berger, editors, *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2016, Santa Barbara, CA, USA, June 13-17, 2016*, pages 386–401. ACM, 2016. doi:10.1145/2908080.2908108.

6 Georgiana Caltais, Hossein Hojjat, Mohammad Mousavi, and Hünkar Can Tunç. DyNetKAT: An Algebra of Dynamic Networks. URL: https://www.cs.le.ac.uk/people/mm789/pub/icalp2021-ext.pdf.

7 Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn L. Talcott. Full Maude: Extending Core Maude. In Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn L. Talcott, editors, *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, volume 4350 of *Lecture Notes in Computer Science*, pages 559–597. Springer, 2007. doi:10.1007/978-3-540-71999-1_18.

8 Nate Foster, Dexter Kozen, Konstantinos Mamouras, Mark Reitblatt, and Alexandra Silva. Probabilistic NetKAT. In Peter Thiemann, editor, *Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9632 of *Lecture Notes in Computer Science*, pages 282–309. Springer, 2016. doi:10.1007/978-3-662-49498-1_12.

9 Nate Foster, Dexter Kozen, Matthew Milano, Alexandra Silva, and Laure Thompson. A Coalgebraic Decision Procedure for NetKAT. In Sriram K. Rajamani and David Walker, editors, *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pages 343–355. ACM, 2015. doi:10.1145/2676726.2677011.

10 Nate Foster, Rob Harrison, Michael J. Freedman, Christopher Monsanto, Jennifer Rexford, Alec Story, and David Walker. Frenetic: a network programming language. In *Proceeding of the 16th ACM SIGPLAN international conference on Functional Programming (ICFP 2011)*. pages 279–291, ACM, 2011. .

11 Ahmed El-Hassany, Ahmed Miserez, Pavol Bielik, Laurent Vanbever, and Martin T. Vechev. SDNRacer: concurrency analysis for software-defined networks. In Chandra Krintz and Emery Berger, Eds. , *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2016)*, 402–415, ACM, 2016. .

12 Maciej Kuzniar, Peter Peresíni, and Dejan Kostic. Providing Reliable FIB Update Acknowledgments in SDN. In Aruna Seneviratne, Christophe Diot, Jim Kurose, Augustin Chaintreau, and Luigi Rizzo, Eds. *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies (CoNEXT 2014)*, 415–422, ACM, 2014. .

13 Tobias Kappé, Paul Brunet, Alexandra Silva, Jana Wagemaker, and Fabio Zanasi. Concurrent Kleene Algebra with Observations: from Hypotheses to Completeness. *CoRR*, abs/2002.09682, 2020. URL: `https://arxiv.org/abs/2002.09682`, `arXiv:2002.09682`.

14 Hyojoon Kim, Joshua Reich, Arpit Gupta, Muhammad Shahbaz, Nick Feamster, and Russell J. Clark. Kinetic: Verifiable dynamic network control. In *12th USENIX Symposium on Networked Systems Design and Implementation, NSDI 15, Oakland, CA, USA, May 4-6, 2015*, pages 59–72. USENIX Association, 2015. URL: `https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/kim`.

15 Zohaib Latif, Kashif Sharif, Fan Li , Md. Monjurul Karim, Sujit Biswas, and Yu Wang. A comprehensive survey of interface protocols for software defined networks. J. Netw. Comput. Appl. 156:102563, 2020. .

16 Jedidiah McClurg, Hossein Hojjat, Nate Foster, and Pavol Cerný. Event-driven network programming. In Chandra Krintz and Emery Berger, editors, *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2016, Santa Barbara, CA, USA, June 13-17, 2016*, pages 369–385. ACM, 2016. `doi:10.1145/2908080.2908097`.

17 Jedidiah McClurg, Hossein Hojjat, and Pavol Cerný. Synchronization Synthesis for Network Programs. In *Proceedings of the 29th International Conference on Computer Aided Verification (CAV 2017).* volume 10427 of Lecture Notes in Computer Science, pages 301–321, Springer, 2017. .

18 Nick McKeown, Thomas E. Anderson, Hari Balakrishnan, Guru M. Parulkar, Larry L. Peterson, Jennifer Rexford, Scott Shenker, and Jonathan S. Turner. OpenFlow: enabling innovation in campus networks. Computer Communication Review, 38(2):69–74, 2008. .

19 Mohammad Reza Mousavi, Michel A. Reniers, and Jan Friso Groote. Notions of bisimulation and congruence formats for SOS with data. *Information and Computation*, 200(1):107 – 147, 2005. `doi:https://doi.org/10.1016/j.ic.2005.03.002`.

20 Tim Nelson, Andrew D. Ferguson, Michael J. G. Scheer, and Shriram Krishnamurthi. Tierless programming and reasoning for software-defined networks. In Ratul Mahajan and Ion Stoica, editors, *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2014, Seattle, WA, USA, April 2-4, 2014*, pages 519–531. USENIX Association, 2014. URL: `https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/nelson`.

21 Mark Reitblatt, Nate Foster, Jennifer Rexford, Cole Schlesinger, and David Walker. Abstractions for network update. In Lars Eggert, Jörg Ott, Venkata N. Padmanabhan, and George Varghese, editors, *ACM SIGCOMM 2012 Conference, SIGCOMM '12, Helsinki, Finland - August 13 - 17, 2012*, pages 323–334. ACM, 2012. `doi:10.1145/2342356.2342427`.

22 Alexandra Silva. Models of Concurrent Kleene Algebra. In Elvira Albert and Laura Kovács, editors, *LPAR 2020: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Alicante, Spain, May 22-27, 2020*, volume 73 of *EPiC Series in Computing*, page 516. EasyChair, 2020. URL: `https://easychair.org/publications/paper/6C8R`.

23 Guido van Rossum. Python programming language. In Jeff Chase and Srinivasan Seshan, editors, *Proceedings of the 2007 USENIX Annual Technical Conference, Santa Clara, CA, USA, June 17-22, 2007.* USENIX, 2007.

24 Teemu Koponen, Keith Amidon, Peter Balland, Martín Casado, Anupam Chanda, Bryan Fulton, Igor Ganichev, Jesse Gross, Paul Ingram, Ethan J. Jackson, Andrew Lambeth, Romain Lenglet, Shih-Hao Li, Amar Padmanabhan, Justin Pettit, Ben Pfaff, Rajiv Ramanathan, Scott Shenker, Alan Shieh, Jeremy Stribling, Pankaj Thakkar, Dan Wendlandt, Alexander Yip, and Ronghua Zhang. Network Virtualization in Multi-tenant Datacenters. In Ratul Mahajan and Ion Stoica, Eds., *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2014)*, 203–216, USENIX Association, 2014.

25 Celio Trois, Marcos Didonet Del Fabro, Luis Carlos Erpen De Bona, and Magnos Martinello. A Survey on SDN Programming Languages: Toward a Taxonomy. IEEE Commun. Surv. Tutorials 18(4): 2687–2712, 2016. .
26 Alexander Vandenbroucke and Tom Schrijvers. P$\lambda\omega$nk: functional probabilistic netkat. *Proc. ACM Program. Lang.*, 4(POPL):39:1–39:27, 2020. `doi:10.1145/3371107`.
27 Jana Wagemaker, Paul Brunet, Simon Docherty, Tobias Kappé, Jurriaan Rot, and Alexandra Silva. Partially Observable Concurrent Kleene Algebra. In Igor Konnov and Laura Kovács, editors, *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference)*, volume 171 of *LIPIcs*, pages 20:1–20:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.CONCUR.2020.20`.

A Soundness Proofs

- Axiom under consideration:

$$\mathbf{0}\,;p \equiv \bot \quad (A0) \tag{33}$$

for $p \in$ DyNetKAT. According to the semantic rules of DyNetKAT, the derivations of the term $\mathbf{0}\,;p$ are as follows:

(a)

$$\text{For all } \sigma' \in [\![\mathbf{0}]\!](\sigma{::}\langle\rangle) : \ (\mathbf{cpol}^{\checkmark}_{\_;})\frac{}{(\mathbf{0}\,;p, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p, H, \sigma' :: H')}$$

However, observe that $[\![\mathbf{0}]\!](\sigma{::}\langle\rangle)$ is equal to empty set:

$$[\![\mathbf{0}]\!](\sigma{::}\langle\rangle) = \{\} \quad (\text{Definition of } \mathbf{0}) \tag{34}$$

Hence, the term $\mathbf{0}\,;p$ does not afford any transition. Similarly, observe that according to the semantic rules of DyNetKAT, the term $\bot$ does not afford any transition. Hence, the following trivially holds:

$$(\mathbf{0}\,;p) \sim \bot \tag{35}$$

- Axiom under consideration:

$$p \oplus q \equiv q \oplus p \quad (A2) \tag{36}$$

for $p, q \in$ DyNetKAT. According to the semantic rules of DyNetKAT, the following are the possible transitions that can initially occur in the terms $p \oplus q$ and $q \oplus p$:

$$\begin{cases} (1)\ (p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1') \\ (2)\ (q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1') \end{cases}$$

$\gamma ::= (\sigma, \sigma') \mid x!z \mid x?z \mid \mathbf{rcfg(x, z)}$

Case (1): $(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$

The derivations of $p \oplus q$ are as follows:

(a)

$$(\mathbf{cpol}_{\_\oplus})\frac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(p \oplus q, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}$$

The derivations of $q \oplus p$ are as follows:

(b)

$$(\mathbf{cpol}_{\oplus\_})\frac{(p, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1)}{(q \oplus p, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1)}$$

As demonstrated in (a) and (b), if $(p, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1)$ holds then both of the terms $p \oplus q$ and $q \oplus p$ converge to the same expression with the γ transition:

$$\begin{aligned} &(p \oplus q, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1) \\ &(q \oplus p, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1) \end{aligned} \tag{37}$$

Case (2): $(q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)$

The derivations of $p \oplus q$ are as follows:

(c)

$$(\mathbf{cpol}_{\oplus\_})\frac{(q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)}{(p \oplus q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)}$$

The derivations of $q \oplus p$ are as follows:

(d)

$$(\mathbf{cpol}_{\_\oplus})\frac{(q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)}{(q \oplus p, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)}$$

As demonstrated in (c) and (d), if $(q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)$ holds then both of the terms $p \oplus q$ and $q \oplus p$ converge to the same expression with the γ transition:

$$\begin{aligned} &(p \oplus q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1) \\ &(q \oplus p, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1) \end{aligned} \tag{38}$$

Therefore, by (37) and (38) it is straightforward to conclude that the following holds:

$$(p \oplus q) \sim (q \oplus p) \tag{39}$$

- Axiom under consideration:

$$(p \oplus q) \oplus r \equiv p \oplus (q \oplus r) \quad (A3) \tag{40}$$

for $p, q, r \in$ DyNetKAT. According to the semantic rules of DyNetKAT, the following are the possible transitions that can initially occur in the terms $(p \oplus q) \oplus r$ and $p \oplus (q \oplus r)$:

$$\begin{cases} (1)\ (p, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1) \\ (2)\ (q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1) \\ (3)\ (r, H_0, H'_0) \xrightarrow{\gamma} (r', H_1, H'_1) \end{cases}$$

$\gamma ::= (\sigma, \sigma') \mid x!z \mid x?z \mid \mathbf{rcfg(x, z)}$

Case (1): $(p, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1)$

The derivations of $(p \oplus q) \oplus r$ are as follows:

(a)

$$(\mathbf{cpol}_{\_\oplus})\dfrac{(\mathbf{cpol}_{\_\oplus})\dfrac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(p \oplus q, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}}{((p \oplus q) \oplus r, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}$$

The derivations of $p \oplus (q \oplus r)$ are as follows:

(b)

$$(\mathbf{cpol}_{\_\oplus})\dfrac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(p \oplus (q \oplus r), H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}$$

As demonstrated in (a) and (b), if $(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$ holds then both of the terms $(p \oplus q) \oplus r$ and $p \oplus (q \oplus r)$ converge to the same expression with the γ transition:

$$\begin{aligned} &((p \oplus q) \oplus r, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1') \\ &(p \oplus (q \oplus r), H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1') \end{aligned} \tag{41}$$

Case (2): $(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')$

The derivations of $(p \oplus q) \oplus r$ are as follows:

(c)

$$(\mathbf{cpol}_{\_\oplus})\dfrac{(\mathbf{cpol}_{\oplus\_})\dfrac{(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}{(p \oplus q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}}{((p \oplus q) \oplus r, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}$$

The derivations of $p \oplus (q \oplus r)$ are as follows:

(d)

$$(\mathbf{cpol}_{\oplus\_})\dfrac{(\mathbf{cpol}_{\_\oplus})\dfrac{(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}{(q \oplus r, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}}{(p \oplus (q \oplus r), H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}$$

As demonstrated in (c) and (d), if $(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')$ holds then both of the terms $(p \oplus q) \oplus r$ and $p \oplus (q \oplus r)$ converge to the same expression with the γ transition:

$$\begin{aligned} &((p \oplus q) \oplus r, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1') \\ &(p \oplus (q \oplus r), H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1') \end{aligned} \tag{42}$$

Case (3): $(r, H_0, H_0') \xrightarrow{\gamma} (r', H_1, H_1')$

The derivations of $(p \oplus q) \oplus r$ are as follows:

(e)

$$(\mathbf{cpol}_{\oplus\_})\dfrac{(r, H_0, H_0') \xrightarrow{\gamma} (r', H_1, H_1')}{((p \oplus q) \oplus r, H_0, H_0') \xrightarrow{\gamma} (r', H_1, H_1')}$$

The derivations of $p \oplus (q \oplus r)$ are as follows:

(f)

$$(\mathbf{cpol}_{\oplus\_})\dfrac{(\mathbf{cpol}_{\oplus\_})\dfrac{(r, H_0, H_0') \xrightarrow{\gamma} (r', H_1, H_1')}{(q \oplus r, H_0, H_0') \xrightarrow{\gamma} (r', H_1, H_1')}}{(p \oplus (q \oplus r), H_0, H_0') \xrightarrow{\gamma} (r', H_1, H_1')}$$

As demonstrated in (e) and (f), if $(r, H_0, H_0') \xrightarrow{\gamma} (r', H_1, H_1')$ holds then both of the terms $(p \oplus q) \oplus r$ and $p \oplus (q \oplus r)$ converge to the same expression with the γ transition:

$$\begin{aligned} &((p \oplus q) \oplus r, H_0, H_0') \xrightarrow{\gamma} (r', H_1, H_1') \\ &(p \oplus (q \oplus r), H_0, H_0') \xrightarrow{\gamma} (r', H_1, H_1') \end{aligned} \tag{43}$$

Therefore, by (41), (42) and (43) it is straightforward to conclude that the following holds:

$$((p \oplus q) \oplus r) \sim (p \oplus (q \oplus r)) \tag{44}$$

- Axiom under consideration:

$$p \oplus p \equiv p \quad (A4) \tag{45}$$

for $p \in$ DyNetKAT. According to the semantic rules of DyNetKAT, the following are the possible transitions that can initially occur in the terms $p \oplus p$ and p:

$$\Big\{(1)\ (p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$$

$\gamma ::= (\sigma, \sigma') \mid x!z \mid x?z \mid \mathbf{rcfg(x, z)}$

Case (1): $(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$

The derivations of $p \oplus p$ are as follows:

(a)

$$(\mathbf{cpol}_{\_\oplus})\dfrac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(p \oplus p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}$$

(b)

$$(\mathbf{cpol}_{\oplus\_})\dfrac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(p \oplus p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}$$

As demonstrated in (a) and (b), if $(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$ holds then it is also the case that the term $p \oplus p$ evolves into the same expression with the γ transition:

$$(p \oplus p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1') \tag{46}$$

Hence, it is straightforward to conclude that the following holds:

$$(p \oplus p) \sim p \tag{47}$$

- Axiom under consideration:

$$p \oplus \bot \equiv p \quad (A5) \tag{48}$$

for $p \in$ DyNetKAT. According to the semantic rules of DyNetKAT, the following are the possible transitions that can initially occur in the terms $p \oplus \bot$ and p:

$$\Big\{(1)\ (p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$$

$\gamma ::= (\sigma, \sigma') \mid x!z \mid x?z \mid \mathbf{rcfg}(\mathbf{x}, \mathbf{z})$

Case (1): $(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$

The derivations of $p \oplus \bot$ are as follows:

(a)

$$(\mathbf{cpol\_}_{\oplus})\frac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(p \oplus \bot, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}$$

As demonstrated in (a), if $(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$ holds then it is also the case that the term $p \oplus \bot$ evolves into the same expression with the γ transition:

$$(p \oplus \bot, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1') \tag{49}$$

Hence, it is straightforward to conclude that the following holds:

$$(p \oplus \bot) \sim p \tag{50}$$

- Axiom under consideration:

$$p \,||\, q \equiv q \,||\, p \quad (A6) \tag{51}$$

for $p, q \in$ DyNetKAT. The soundness proof of the axiom $(A6)$ follows by induction on the size of the syntactic tree associated to $p \,||\, q$. Without loss of generality, assume p and q are in n.f. The size of $p \,||\, q$ is then defined as follows:

$$size(\bot) = 1 \tag{52}$$
$$size(\alpha \cdot \pi \,; t) = 2 + size(t) \tag{53}$$
$$size(x?z \,; t) = 2 + size(t) \tag{54}$$
$$size(x!z \,; t) = 2 + size(t) \tag{55}$$
$$size(\mathbf{rcfg}_{x,z} \,; t) = 2 + size(t) \tag{56}$$
$$size(p \oplus q) = 1 + size(p) + size(q) \tag{57}$$
$$size(p \,||\, q) = 1 + size(p) + size(q) \tag{58}$$

Base case.

- $size(p \,||\, q) = 3$. It follows that $p \triangleq \bot$ and $q \triangleq \bot$. Therefore, the soundness of $(A6)$ trivially holds.

Induction step. Assume $(A6)$ is sound for all p, q such that $size(p \,||\, q) \leq M$, with $M \in \mathbb{N}$. We want to show that $(A6)$ is sound for all p, q such that $size(p \,||\, q) > M$.

(i) $p \triangleq \bot$. Then, it is straightforward to observe that both $\bot \,||\, q$ and $q \,||\, \bot$ evolve according to the semantic rules corresponding to q. Hence, we can safely conclude that $(\bot \,||\, q) \sim (q \,||\, \bot)$ holds.

(ii) $p \triangleq \alpha \cdot \pi; p'$. Consider an arbitrary but fixed network packet σ, let $S_{\alpha\pi} \triangleq [\![\alpha \cdot \pi]\!](\sigma::\langle\rangle)$. The first step derivations entailed by p in a context $p \,||\, t$ are as follows:

(a)

$$\text{For all } \sigma' \in S_{\alpha\pi}: \quad (\mathbf{cpol}^{\checkmark}_{\_;})\frac{}{(\alpha \cdot \pi \,;p', \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p', H, \sigma' :: H')}$$
$$(\mathbf{cpol}_{\_||})\frac{}{((\alpha \cdot \pi \,;p') \,||\, t, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p' \,||\, t, H, \sigma' :: H')}$$

The first step derivations entailed by p in a context $t \,||\, p$ are as follows:

(b)

$$\text{For all } \sigma' \in S_{\alpha\pi}: \quad (\mathbf{cpol}^{\checkmark}_{\_;})\frac{}{(\alpha \cdot \pi \,;p', \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p', H, \sigma' :: H')}$$
$$(\mathbf{cpol}_{||\_})\frac{}{(t \,||(\alpha \cdot \pi \,;p'), \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (t \,||\, p', H, \sigma' :: H')}$$

Hence, given that q in n.f. always evolves into a term t with simpler structure (according to the DyNetKAT semantic rules), and based on the induction hypothesis, it is safe to conclude that $(p \,||\, q) \sim (q \,||\, p)$.

(iii) $p \triangleq \mathbf{rcfg}_{x,z}\,; p'$. The reasoning is similar to (ii) above.

(iv) $p \triangleq x?z; p'$. The first step of asynchronous derivations entailed by p in a context $p \,||\, t$ are as follows:

(a)

$$(\mathbf{cpol}_{?})\frac{}{(x?z\,;p', H, H') \xrightarrow{x?z} (p', H, H')}$$
$$(\mathbf{cpol}_{\_||})\frac{}{((x?z\,;p') \,||\, t, H, H') \xrightarrow{x?z} (p' \,||\, t, H, H')}$$

The first step of asynchronous derivations entailed by p in a context $t \,||\, p$ are as follows:

(b)

$$(\mathbf{cpol}_{?})\frac{}{(x?z\,;p', H, H') \xrightarrow{x?z} (p', H, H')}$$
$$(\mathbf{cpol}_{||\_})\frac{}{(t \,||(x?z\,;p'), H, H') \xrightarrow{x?z} (t \,||\, p', H, H')}$$

Furthermore, if q has a summand of shape $x!z\,; q'$, then:
The first step synchronous derivations of $p \,||\, q$ are as follows:

(c)

$$(\mathbf{cpol}_{?!})\frac{(\mathbf{cpol}_{?})\frac{}{(x?z; p', H, H') \xrightarrow{x?z} (p', H, H')} \quad (\mathbf{cpol}_{!})\frac{}{(x!z\,;q', H, H') \xrightarrow{x!z} (q', H, H')}}{((x?z\,;p') \,||(x!z\,;q'), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H, H')}$$

The first step synchronous derivations of $q \,||\, p$ are as follows:

(d)

$$(\mathbf{cpol}_{!})\frac{}{(x!z\,;q', H, H') \xrightarrow{x!z} (q', H, H')} \quad (\mathbf{cpol}_{?})\frac{}{(x?z;p', H, H') \xrightarrow{x?z} (p', H, H')}$$
$$(\mathbf{cpol}_{!?})\frac{}{((x!z\,;q')\,||(x?z\,;p'), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (q'\,||\,p', H, H')}$$

In connection with (iv)(a) and (iv)(b) above, note that q in n.f. always evolves into a term t with simpler structure (according to the DyNetKAT semantic rules). This, together with the observations in (iv)(c) and (iv)(d), and based on the induction hypothesis, enable us to safely to conclude that $(p\,||\,q) \sim (q\,||\,p)$.

(v) $p \triangleq x!z\,;p'$. The reasoning is similar to (iv) above.

(vi) $p \triangleq p_1 \oplus p_2$ where p_1 and p_2 are in n.f. Without loss of generality, assume $p_1 ::= \alpha \cdot \pi\,;p_1' \mid \mathbf{rcfg}_{x,z}\,;p_1' \mid x?z\,;p_1' \mid x!z\,;p_1'$ and assume $(p_1, H_0, H_0') \xrightarrow{\gamma} (p_1', H_1, H_1')$. The derivation entailed by p_1 in p is as follows:

$$(\mathbf{cpol\_}_{\oplus})\frac{(p_1, H_0, H_0') \xrightarrow{\gamma} (p_1', H_1, H_1')}{(p_1 \oplus p_2, H_0, H_0') \xrightarrow{\gamma} (p_1', H_1, H_1')}$$

From this point onward, showing that the first step derivations entailed by p_1 in a context $p\,||\,t$ correspond to the first step derivations entailed by p_1 in a context $t\,||\,p$ follows the reasoning in (ii)−(v), with p_1 ranging over terms of shape $(\alpha \cdot \pi\,;p_1'), (\mathbf{rcfg}_{x,z}\,;p_1'), (x!z\,;p_1')$ and $(x?z\,;p_1')$, respectively. Hence, given that q in n.f. always evolves into a term t with simpler structure (according to the DyNetKAT semantic rules), and based on the induction hypothesis, it is safe to conclude that $(p\,||\,q) \sim (q\,||\,p)$.

- Axiom under consideration:

$$p\,||\,\bot \equiv p \quad (A7) \tag{59}$$

for $p \in$ DyNetKAT. According to the semantic rules of DyNetKAT, observe that both $p\,||\,\bot$ and p evolve according to the semantic rules corresponding to p. Hence, it is straightforward to conclude that the following holds:

$$(p\,||\,\bot) \sim p \tag{60}$$

- Axiom under consideration:

$$p\,||\,q \equiv p\llfloor q \oplus q\llfloor p \oplus p \mid q \quad (A8) \tag{61}$$

for $p, q \in$ DyNetKAT. According to the semantic rules of DyNetKAT, the following are the possible transitions that can initially occur in the terms $p\,||\,q$ and $p\llfloor q \oplus q\llfloor p \oplus p \mid q$:

$$\begin{cases} (1)\ (p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1') \\ (2)\ (q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1') \\ (3)\ (p, H_0, H_0') \xrightarrow{x!z} (p', H_1, H_1') \quad (q, H_0, H_0') \xrightarrow{x?z} (q', H_1, H_1') \\ (4)\ (p, H_0, H_0') \xrightarrow{x?z} (p', H_1, H_1') \quad (q, H_0, H_0') \xrightarrow{x!z} (q', H_1, H_1') \end{cases}$$

$\gamma ::= (\sigma, \sigma') \mid x!z \mid x?z \mid \mathbf{rcfg(x, z)}$

Case (1): $(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$

The derivations of $p \,||\, q$ are as follows:

(a)

$$(\mathbf{cpol}\_{||})\dfrac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(p \,||\, q, H_0, H_0') \xrightarrow{\gamma} (p' \,||\, q, H_1, H_1')}$$

The derivations of $p \lfloor\!\lfloor q \oplus q \lfloor\!\lfloor p \oplus p \mid q$ are as follows:

(b)

$$(\mathbf{cpol}\_{\oplus})\dfrac{(\lfloor\!\lfloor)\dfrac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(p \lfloor\!\lfloor q, H_0, H_0') \xrightarrow{\gamma} (p' \,||\, q, H_1, H_1')}}{(p \lfloor\!\lfloor q \oplus q \lfloor\!\lfloor p \oplus p \mid q, H_0, H_0') \xrightarrow{\gamma} (p' \,||\, q, H_1, H_1')}$$

As demonstrated in (a) and (b), if $(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$ holds then both of the terms $p \,||\, q$ and $p \lfloor\!\lfloor q \oplus q \lfloor\!\lfloor p \oplus p \mid q$ converge to the same expression with the γ transition:

$$\begin{aligned} (p \,||\, q, H_0, H_0') &\xrightarrow{\gamma} (p' \,||\, q, H_1, H_1') \\ (p \lfloor\!\lfloor q \oplus q \lfloor\!\lfloor p \oplus p \mid q, H_0, H_0') &\xrightarrow{\gamma} (p' \,||\, q, H_1, H_1') \end{aligned} \tag{62}$$

Case (2): $(q, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$

The derivations of $p \,||\, q$ are as follows:

(c)

$$(\mathbf{cpol}_{||}\_)\dfrac{(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}{(p \,||\, q, H_0, H_0') \xrightarrow{\gamma} (p \,||\, q', H_1, H_1')}$$

The derivations of $p \lfloor\!\lfloor q \oplus q \lfloor\!\lfloor p \oplus p \mid q$ are as follows:

(d)

$$(\mathbf{cpol}\_{\oplus})\dfrac{(\mathbf{cpol}_{\oplus}\_)\dfrac{(\lfloor\!\lfloor)\dfrac{(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}{(q \lfloor\!\lfloor p, H_0, H_0') \xrightarrow{\gamma} (q' \,||\, p, H_1, H_1')}}{(p \lfloor\!\lfloor q \oplus q \lfloor\!\lfloor p, H_0, H_0') \xrightarrow{\gamma} (q' \,||\, p, H_1, H_1')}}{(p \lfloor\!\lfloor q \oplus q \lfloor\!\lfloor p \oplus p \mid q, H_0, H_0') \xrightarrow{\gamma} (q' \,||\, p, H_1, H_1')}$$

As demonstrated in (c) and (d), if $(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$ holds then both of the terms $p \,||\, q$ and $p \lfloor\!\lfloor q \oplus q \lfloor\!\lfloor p \oplus p \mid q$ are able to perform the γ transition:

$$\begin{aligned} (p \,||\, q, H_0, H_0') &\xrightarrow{\gamma} (p \,||\, q', H_1, H_1') \\ (p \lfloor\!\lfloor q \oplus q \lfloor\!\lfloor p \oplus p \mid q, H_0, H_0') &\xrightarrow{\gamma} (q' \,||\, p, H_1, H_1') \end{aligned} \tag{63}$$

Observe that the terms evolve into different expressions, however, according to the axiom $A6$ the "$||$" operator is commutative. Hence, the following holds:

$$(p \,||\, q') \sim (q' \,||\, p) \tag{64}$$

Case (3): $(p, H_0, H'_0) \xrightarrow{x!z} (p', H_1, H'_1) \quad (q, H_0, H'_0) \xrightarrow{x?z} (q', H_1, H'_1)$

The derivations of $p \,||\, q$ are as follows:

(e)

$$(\mathbf{cpol}_{!?})\frac{(p, H_0, H'_0) \xrightarrow{x!z} (p', H_1, H'_1) \quad (q, H_0, H'_0) \xrightarrow{x?z} (q', H_1, H'_1)}{(p \,||\, q, H_0, H'_0) \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H_1, H'_1)}$$

The derivations of $p \llfloor q \oplus q \llfloor p \oplus p \mid q$ are as follows:

(f)

$$(\mathbf{cpol}_{\oplus\_})\frac{(\mathbf{cpol}_{!?})\dfrac{(p, H_0, H'_0) \xrightarrow{x!z} (p', H_1, H'_1) \quad (q, H_0, H'_0) \xrightarrow{x?z} (q', H_1, H'_1)}{(p \mid q, H_0, H'_0) \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H_1, H'_1)}}{(p \llfloor q \oplus q \llfloor p \oplus p \mid q, H_0, H'_0) \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H_1, H'_1)}$$

As demonstrated in (e) and (f), if $(p, H_0, H'_0) \xrightarrow{x!z} (p', H_1, H'_1)$ and $(q, H_0, H'_0) \xrightarrow{x?z} (q', H_1, H'_1)$ hold then both of the terms $p \,||\, q$ and $p \llfloor q \oplus q \llfloor \oplus p \mid q$ converge to the same expression with the $\mathbf{rcfg(x, z)}$ transition:

$$\begin{aligned} (p \,||\, q, H_0, H'_0) &\xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H_1, H'_1) \\ (p \llfloor q \oplus q \llfloor p \oplus p \mid q, H_0, H'_0) &\xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H_1, H'_1) \end{aligned} \tag{65}$$

Case (4): $(p, H_0, H'_0) \xrightarrow{x?z} (p', H_1, H'_1) \quad (q, H_0, H'_0) \xrightarrow{x!z} (q', H_1, H'_1)$

The derivations of $p \,||\, q$ are as follows:

(g)

$$(\mathbf{cpol}_{?!})\frac{(p, H_0, H'_0) \xrightarrow{x?z} (p', H_1, H'_1) \quad (q, H_0, H'_0) \xrightarrow{x!z} (q', H_1, H'_1)}{(p \,||\, q, H_0, H'_0) \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H_1, H'_1)}$$

The derivations of $p \llfloor q \oplus q \llfloor p \oplus p \mid q$ are as follows:

(h)

$$(\mathbf{cpol}_{\oplus\_})\frac{(\mathbf{cpol}_{?!})\dfrac{(p, H_0, H'_0) \xrightarrow{x?z} (p', H_1, H'_1) \quad (q, H_0, H'_0) \xrightarrow{x!z} (q', H_1, H'_1)}{(p \mid q, H_0, H'_0) \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H_1, H'_1)}}{(p \llfloor q \oplus q \llfloor p \oplus p \mid q, H_0, H'_0) \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H_1, H'_1)}$$

As demonstrated in (g) and (h), if $(p, H_0, H'_0) \xrightarrow{x?z} (p', H_1, H'_1)$ and $(q, H_0, H'_0) \xrightarrow{x!z} (q', H_1, H'_1)$ hold then both of the terms $p \,||\, q$ and $p \llfloor q \oplus q \llfloor \oplus p \mid q$ converge to the same expression with the $\mathbf{rcfg(x, z)}$ transition:

$$\begin{aligned} (p \,||\, q, H_0, H'_0) &\xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H_1, H'_1) \\ (p \llfloor q \oplus q \llfloor p \oplus p \mid q, H_0, H'_0) &\xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H_1, H'_1) \end{aligned} \tag{66}$$

Therefore, by (62), (63), (64), (65) and (66) it is straightforward to conclude that the following holds:

$$(p \,||\, q) \sim (p \llfloor q \oplus q \llfloor p \oplus p \mid q) \tag{67}$$

- Axiom under consideration:

$$\bot \llfloor p \equiv \bot \quad (A9) \tag{68}$$

for $p \in$ DyNetKAT. Observe that according to the semantic rules of DyNetKAT, the terms $\bot \llfloor p$ and $\bot$ do not afford any transition. Hence, the following trivially holds:

$$(\bot \llfloor p) \sim \bot \tag{69}$$

- Axiom under consideration:

$$(a\,; p) \llfloor q \equiv a\,;(p \,||\, q) \quad (A10) \tag{70}$$

for $a \in \{z, x?z, x!z, \mathbf{rcfg}_{x,z}\}$, $z \in \text{NetKAT}^{-\mathbf{dup}}$ and $p, q \in$ DyNetKAT. In the following, we make a case analysis on the shape of a and show that the terms $(a\,; p) \llfloor q$ and $a\,;(p \,||\, q)$ are bisimilar.

Case (1): $a \triangleq z$

Consider an arbitrary but fixed network packet σ, let $S_z \triangleq [\![z]\!](\sigma{::}\langle\rangle)$. The derivations of $(z\,; p) \llfloor q$ are as follows:

(a)

$$\text{For all } \sigma' \in S_z : \quad (\llfloor)\dfrac{(\mathbf{cpol}^{\checkmark}_{\_;})\dfrac{}{(z\,; p, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p, H, \sigma' :: H')}}{((z\,; p) \llfloor q, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p \,||\, q, H, \sigma' :: H)}$$

The derivations of $z\,;(p \,||\, q)$ are as follows:

(b)

$$\text{For all } \sigma' \in S_z : \quad (\mathbf{cpol}^{\checkmark}_{\_;})\dfrac{}{(z\,;(p \,||\, q), \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p \,||\, q, H, \sigma' :: H')}$$

As demonstrated in (a) and (b), both of the terms $(z; p) \llfloor q$ and $z\,;(p \,||\, q)$ initially afford the same set of transitions of shape (σ, σ') and they converge to the same expression after taking these transitions:

$$\begin{aligned} &((z; p) \llfloor q, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p \,||\, q, H, \sigma' :: H') \\ &(z\,;(p \,||\, q), \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p \,||\, q, H, \sigma' :: H') \end{aligned} \tag{71}$$

Case (2): $a \triangleq x?z$

The derivations of $(x?z\,; p) \llfloor q$ are as follows:

(c)

$$\dfrac{(\mathbf{cpol}_?)\dfrac{}{(x?z\,;p,H,H') \xrightarrow{x?z} (p,H,H')}}{(\lfloor\!\lfloor)\ ((x?z\,;p)\lfloor\!\lfloor q,H,H') \xrightarrow{x?z} (p\,||\,q,H,H')}$$

The derivations of $x?z\,;(p\,||\,q)$ are as follows:

(d)

$$(\mathbf{cpol}_?)\dfrac{}{(x?z\,;(p\,||\,q),H,H') \xrightarrow{x?z} (p\,||\,q,H,H')}$$

As demonstrated in (c) and (d), both of the terms $(x?z;p)\lfloor\!\lfloor q$ and $x?z\,;(p\,||\,q)$ initially only afford the $x?z$ transition and they converge to the same expression after taking this transition:

$$\begin{aligned} ((x?z;p)\lfloor\!\lfloor q,H,H') &\xrightarrow{x?z} (p\,||\,q,H,H') \\ (x?z\,;(p\,||\,q),H,H') &\xrightarrow{x?z} (p\,||\,q,H,H') \end{aligned} \tag{72}$$

Case (3): $a \triangleq x!z$

The derivations of $(x!z\,;p)\lfloor\!\lfloor q$ are as follows:

(e)

$$\dfrac{(\mathbf{cpol}_!)\dfrac{}{(x!z\,;p,H,H') \xrightarrow{x!z} (p,H,H')}}{(\lfloor\!\lfloor)\ ((x!z\,;p)\lfloor\!\lfloor q,H,H') \xrightarrow{x!z} (p\,||\,q,H,H')}$$

The derivations of $x!z\,;(p\,||\,q)$ are as follows:

(f)

$$(\mathbf{cpol}_!)\dfrac{}{(x!z\,;(p\,||\,q),H,H') \xrightarrow{x!z} (p\,||\,q,H,H')}$$

As demonstrated in (e) and (f), both of the terms $(x!z;p)\lfloor\!\lfloor q$ and $x!z\,;(p\,||\,q)$ initially only afford the $x!z$ transition and they converge to the same expression after taking this transition:

$$\begin{aligned} ((x!z;p)\lfloor\!\lfloor q,H,H') &\xrightarrow{x!z} (p\,||\,q,H,H') \\ (x!z\,;(p\,||\,q),H,H') &\xrightarrow{x!z} (p\,||\,q,H,H') \end{aligned} \tag{73}$$

Case (4): $a \triangleq \mathbf{rcfg}_{x,z}$

The derivations of $(\mathbf{rcfg}_{x,z}\,;p)\lfloor\!\lfloor q$ are as follows:

(g)

$$\dfrac{(\mathbf{rcfg_{x,z}})\dfrac{}{(\mathbf{rcfg}_{x,z}\,;p,H,H') \xrightarrow{\mathbf{rcfg(x,z)}} (p,H,H')}}{(\lfloor\!\lfloor)\ ((\mathbf{rcfg}_{x,z}\,;p)\lfloor\!\lfloor q,H,H') \xrightarrow{\mathbf{rcfg(x,z)}} (p\,||\,q,H,H')}$$

The derivations of $\mathbf{rcfg}_{x,z}\,;(p\,||\,q)$ are as follows:

(h)

$$(\mathbf{rcfg_{x,z}})\frac{}{(\mathbf{rcfg}_{x,z}\,;(p\,||\,q), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p\,||\,q, H, H')}$$

As demonstrated in (g) and (h), both of the terms $(\mathbf{rcfg}_{x,z}; p)\lfloor\!\lfloor q$ and $\mathbf{rcfg}_{x,z}\,;(p\,||\,q)$ initially only afford the $\mathbf{rcfg(x,z)}$ transition and they converge to the same expression after taking this transition:

$$\begin{aligned}((\mathbf{rcfg}_{x,z}; p)\lfloor\!\lfloor q, H, H') &\xrightarrow{\mathbf{rcfg(x,z)}} (p\,||\,q, H, H')\\ (\mathbf{rcfg}_{x,z}\,;(p\,||\,q), H, H') &\xrightarrow{\mathbf{rcfg(x,z)}} (p\,||\,q, H, H')\end{aligned} \tag{74}$$

Therefore, by (71), (72), (73) and (74) it is straightforward to conclude that the following holds:

$$((a; p)\lfloor\!\lfloor q) \sim (a\,;(p\,||\,q)) \tag{75}$$

- Axiom under consideration:

$$(p \oplus q)\lfloor\!\lfloor r \equiv (p\lfloor\!\lfloor r) \oplus (q\lfloor\!\lfloor r) \quad (A11) \tag{76}$$

for $p, q, r \in$ DyNetKAT. According to the semantic rules of DyNetKAT, the following are the possible transitions that can initially occur in the terms $(p \oplus q)\lfloor\!\lfloor r$ and $(p\lfloor\!\lfloor r) \oplus (q\lfloor\!\lfloor r)$:

$$\begin{cases}(1)\ (p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')\\ (2)\ (q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')\end{cases}$$

$\gamma ::= (\sigma, \sigma') \mid x!z \mid x?z \mid \mathbf{rcfg(x,z)}$

Case (1): $(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$

The derivations of $(p \oplus q)\lfloor\!\lfloor r$ are as follows:

(a)

$$(\lfloor\!\lfloor)\frac{(\mathbf{cpol\_}_{\oplus})\dfrac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(p \oplus q, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}}{((p \oplus q)\lfloor\!\lfloor r, H_0, H_0') \xrightarrow{\gamma} (p'\,||\,r, H_1, H_1')}$$

The derivations of $(p\lfloor\!\lfloor r) \oplus (q\lfloor\!\lfloor r)$ are as follows:

(b)

$$(\mathbf{cpol\_}_{\oplus})\frac{(\lfloor\!\lfloor)\dfrac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(p\lfloor\!\lfloor r, H_0, H_0') \xrightarrow{\gamma} (p'\,||\,r, H_1, H_1')}}{((p\lfloor\!\lfloor r) \oplus (q\lfloor\!\lfloor r), H_0, H_0') \xrightarrow{\gamma} (p'\,||\,r, H_1, H_1')}$$

As demonstrated in (a) and (b), if $(p, H_0, H'_0) \xrightarrow{\gamma} (p', H_1, H'_1)$ holds then both of the terms $(p \oplus q)\llfloor r$ and $(p\llfloor r) \oplus (q\llfloor r)$ converge to the same expression with the γ transition:

$$\begin{aligned} ((p \oplus q)\llfloor r, H_0, H'_0) &\xrightarrow{\gamma} (p' \,||\, r, H_1, H'_1) \\ ((p\llfloor r) \oplus (q\llfloor r), H_0, H'_0) &\xrightarrow{\gamma} (p' \,||\, r, H_1, H'_1) \end{aligned} \tag{77}$$

Case (2): $(q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)$

The derivations of $(p \oplus q)\llfloor r$ are as follows:

(c)

$$(\llfloor)\dfrac{(\mathbf{cpol}_{\oplus\_})\dfrac{(q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)}{(p \oplus q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)}}{((p \oplus q)\llfloor r, H_0, H'_0) \xrightarrow{\gamma} (q' \,||\, r, H_1, H'_1)}$$

The derivations of $(p\llfloor r) \oplus (q\llfloor r)$ are as follows:

(d)

$$(\mathbf{cpol}_{\oplus\_})\dfrac{(\llfloor)\dfrac{(q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)}{(q\llfloor r, H_0, H'_0) \xrightarrow{\gamma} (q' \,||\, r, H_1, H'_1)}}{((p\llfloor r) \oplus (q\llfloor r), H_0, H'_0) \xrightarrow{\gamma} (q' \,||\, r, H_1, H'_1)}$$

As demonstrated in (c) and (d), if $(q, H_0, H'_0) \xrightarrow{\gamma} (q', H_1, H'_1)$ holds then both of the terms $(p \oplus q)\llfloor r$ and $(p\llfloor r) \oplus (q\llfloor r)$ converge to the same expression with the γ transition:

$$\begin{aligned} ((p \oplus q)\llfloor r, H_0, H'_0) &\xrightarrow{\gamma} (q' \,||\, r, H_1, H'_1) \\ ((p\llfloor r) \oplus (q\llfloor r), H_0, H'_0) &\xrightarrow{\gamma} (q' \,||\, r, H_1, H'_1) \end{aligned} \tag{78}$$

Therefore, by (77) and (78) it is straightforward to conclude that the following holds:

$$((p \oplus q)\llfloor r) \sim ((p\llfloor r) \oplus (q\llfloor r)) \tag{79}$$

- Axiom under consideration:

$$(p \oplus q) \mid r \equiv (p \mid r) \oplus (q \mid r) \quad (A13) \tag{80}$$

for $p, q, r \in$ DyNetKAT. According to the semantic rules of DyNetKAT, the following are the possible transitions that can initially occur in the terms $(p \oplus q) \mid r$ and $(p \mid r) \oplus (q \mid r)$:

$$\begin{cases} (1)\ (p, H, H') \xrightarrow{x!z} (p', H, H') \quad (r, H, H') \xrightarrow{x?z} (r', H, H') \\ (2)\ (p, H, H') \xrightarrow{x?z} (p', H, H') \quad (r, H, H') \xrightarrow{x!z} (r', H, H') \\ (3)\ (q, H, H') \xrightarrow{x!z} (q', H, H') \quad (r, H, H') \xrightarrow{x?z} (r', H, H') \\ (4)\ (q, H, H') \xrightarrow{x?z} (q', H, H') \quad (r, H, H') \xrightarrow{x!z} (r', H, H') \end{cases}$$

Case (1): $(p, H, H') \xrightarrow{x!z} (p', H, H') \quad (r, H, H') \xrightarrow{x?z} (r', H, H')$

The derivations of $(p \oplus q) \mid r$ are as follows:

(a)

$$(|^{!?})\dfrac{(\mathbf{cpol}\_{\oplus})\dfrac{(p,H,H') \xrightarrow{x!z} (p',H,H')}{(p \oplus q, H, H') \xrightarrow{x!z} (p',H,H')} \qquad \dfrac{}{(r,H,H') \xrightarrow{x?z} (r',H,H')}}{(p \oplus q) \mid r, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, r', H, H')}$$

The derivations of $(p \mid r) \oplus (q \mid r)$ are as follows:

(b)

$$(\mathbf{cpol}\_{\oplus})\dfrac{(|^{!?})\dfrac{(p,H,H') \xrightarrow{x!z} (p',H,H') \qquad (r,H,H') \xrightarrow{x?z} (r',H,H')}{(p \mid r, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, r', H, H')}}{((p \mid r) \oplus (q \mid r), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, r', H, H')}$$

As demonstrated in (a) and (b), if $(p,H,H') \xrightarrow{x!z} (p',H,H')$ and $(r,H,H') \xrightarrow{x!z} (r', H,H')$ hold then both of the terms $(p \oplus q) \mid r$ and $(p \mid r) \oplus (q \mid r)$ converge to the same expression with the $\mathbf{rcfg(x,z)}$ transition:

$$\begin{aligned} ((p \oplus q) \mid r, H, H') &\xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, r', H, H') \\ ((p \mid r) \oplus (q \mid r), H, H') &\xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, r', H, H') \end{aligned} \tag{81}$$

Case (2): $(p,H,H') \xrightarrow{x?z} (p',H,H') \quad (r,H,H') \xrightarrow{x!z} (r',H,H')$

The derivations of $(p \oplus q) \mid r$ are as follows:

(c)

$$(|^{?!})\dfrac{(\mathbf{cpol}\_{\oplus})\dfrac{(p,H,H') \xrightarrow{x?z} (p',H,H')}{(p \oplus q, H, H') \xrightarrow{x?z} (p',H,H')} \qquad \dfrac{}{(r,H,H') \xrightarrow{x!z} (r',H,H')}}{(p \oplus q) \mid r, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, r', H, H')}$$

The derivations of $(p \mid r) \oplus (q \mid r)$ are as follows:

(d)

$$(\mathbf{cpol}\_{\oplus})\dfrac{(|^{?!})\dfrac{(p,H,H') \xrightarrow{x?z} (p',H,H') \qquad (r,H,H') \xrightarrow{x!z} (r',H,H')}{(p \mid r, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, r', H, H')}}{((p \mid r) \oplus (q \mid r), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, r', H, H')}$$

As demonstrated in (c) and (d), if $(p,H,H') \xrightarrow{x!z} (p',H,H')$ and $(r,H,H') \xrightarrow{x?z} (r', H,H')$ hold then both of the terms $(p \oplus q) \mid r$ and $(p \mid r) \oplus (q \mid r)$ converge to the same expression with the $\mathbf{rcfg(x,z)}$ transition:

$$\begin{aligned} ((p \oplus q) \mid r, H, H') &\xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, r', H, H') \\ ((p \mid r) \oplus (q \mid r), H, H') &\xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, r', H, H') \end{aligned} \tag{82}$$

Case (3): $(q,H,H') \xrightarrow{x!z} (q',H,H') \quad (r,H,H') \xrightarrow{x?z} (r',H,H')$

The derivations of $(p \oplus q) \mid r$ are as follows:

(e)

$$
(|^{!?})\,\frac{(\mathbf{cpol}_{\oplus\_})\,\dfrac{(q, H, H') \xrightarrow{x!z} (q', H, H')}{(p \oplus q, H, H') \xrightarrow{x!z} (q', H, H')} \qquad \dfrac{}{(r, H, H') \xrightarrow{x?z} (r', H, H')}}{(p \oplus q) \mid r, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (q' \,||\, r', H, H')}
$$

The derivations of $(p \mid r) \oplus (q \mid r)$ are as follows:

(f)

$$
(\mathbf{cpol}_{\oplus\_})\,\frac{(|^{!?})\,\dfrac{(q, H, H') \xrightarrow{x!z} (q', H, H') \qquad (r, H, H') \xrightarrow{x?z} (r', H, H')}{(q \mid r, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (q' \,||\, r', H, H')}}{((p \mid r) \oplus (q \mid r), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (q' \,||\, r', H, H')}
$$

As demonstrated in (e) and (f), if $(q, H, H') \xrightarrow{x!z} (q', H, H')$ and $(r, H, H') \xrightarrow{x!z} (r', H, H')$ hold then both of the terms $(p \oplus q) \mid r$ and $(p \mid r) \oplus (q \mid r)$ converge to the same expression with the $\mathbf{rcfg(x, z)}$ transition:

$$
\begin{aligned}
((p \oplus q) \mid r, H, H') &\xrightarrow{\mathbf{rcfg(x,z)}} (q' \,||\, r', H, H') \\
((p \mid r) \oplus (q \mid r), H, H') &\xrightarrow{\mathbf{rcfg(x,z)}} (q' \,||\, r', H, H')
\end{aligned} \tag{83}
$$

Case (4): $(q, H, H') \xrightarrow{x?z} (q', H, H') \quad (r, H, H') \xrightarrow{x!z} (r', H, H')$

The derivations of $(p \oplus q) \mid r$ are as follows:

(g)

$$
(|^{?!})\,\frac{(\mathbf{cpol}_{\oplus\_})\,\dfrac{(q, H, H') \xrightarrow{x?z} (q', H, H')}{(p \oplus q, H, H') \xrightarrow{x?z} (q', H, H')} \qquad \dfrac{}{(r, H, H') \xrightarrow{x!z} (r', H, H')}}{(p \oplus q) \mid r, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, r', H, H')}
$$

The derivations of $(p \mid r) \oplus (q \mid r)$ are as follows:

(h)

$$
(\mathbf{cpol}_{\oplus\_})\,\frac{(|^{?!})\,\dfrac{(q, H, H') \xrightarrow{x?z} (q', H, H') \qquad (r, H, H') \xrightarrow{x!z} (r', H, H')}{(q \mid r, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (q' \,||\, r', H, H')}}{((p \mid r) \oplus (q \mid r), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (q' \,||\, r', H, H')}
$$

As demonstrated in (g) and (h), if $(q, H, H') \xrightarrow{x?z} (q', H, H')$ and $(r, H, H') \xrightarrow{x!z} (r', H, H')$ hold then both of the terms $(p \oplus q) \mid r$ and $(p \mid r) \oplus (q \mid r)$ converge to the same expression with the $\mathbf{rcfg(x, z)}$ transition:

$$
\begin{aligned}
((p \oplus q) \mid r, H, H') &\xrightarrow{\mathbf{rcfg(x,z)}} (q' \,||\, r', H, H') \\
((p \mid r) \oplus (q \mid r), H, H') &\xrightarrow{\mathbf{rcfg(x,z)}} (q' \,||\, r', H, H')
\end{aligned} \tag{84}
$$

Therefore, by (81), (82), (83) and (84) it is straightforward to conclude that the following holds:

$$((p \oplus q) \mid r) \sim ((p \mid r) \oplus (q \mid r)) \tag{85}$$

- Axiom under consideration:

$$p \mid q \equiv q \mid p \quad (A14) \tag{86}$$

for $p, q \in$ DyNetKAT. According to the semantic rules of DyNetKAT, the following are the possible transitions that can initially occur in the terms $p \mid q$ and $q \mid p$:

$$\begin{cases} (1)\ (p, H, H') \xrightarrow{x!z} (p', H, H') \quad (q, H, H') \xrightarrow{x?z} (q', H, H') \\ (2)\ (p, H, H') \xrightarrow{x?z} (p', H, H') \quad (q, H, H') \xrightarrow{x!z} (q', H, H') \end{cases}$$

Case (1): $(p, H, H') \xrightarrow{x!z} (p', H, H') \quad (q, H, H') \xrightarrow{x?z} (q', H, H')$

The derivations of $p \mid q$ are as follows:

(a)

$$(\mid^{!?})\frac{(p, H, H') \xrightarrow{x!z} (p', H, H') \qquad (q, H, H') \xrightarrow{x?z} (q', H, H')}{(p \mid q, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H, H')}$$

The derivations of $q \mid p$ are as follows:

(b)

$$(\mid^{?!})\frac{(q, H, H') \xrightarrow{x?z} (q', H, H') \qquad (p, H, H') \xrightarrow{x!z} (p', H, H')}{(q \mid p, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (q' \,||\, p', H, H')}$$

As demonstrated in (a) and (b), if $(p, H, H') \xrightarrow{x!z} (p', H, H')$ and $(q, H, H') \xrightarrow{x?z} (q', H, H')$ hold then both of the terms $p \mid q$ and $q \mid p$ are able to perform the $\mathbf{rcfg(x, z)}$ transition:

$$\begin{aligned} &(p \mid q, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H, H') \\ &(q \mid p, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (q' \,||\, p', H, H') \end{aligned} \tag{87}$$

Observe that the terms evolve into different expressions and we would now need to check if these terms are bisimilar. According to the axiom $(A6)$, the "||" operator is commutative. Hence, the following holds:

$$(p' \,||\, q') \sim (q' \,||\, p') \tag{88}$$

Case (2): $(p, H, H') \xrightarrow{x?z} (p', H, H') \quad (q, H, H') \xrightarrow{x!z} (q', H, H')$

The derivations of $p \mid q$ are as follows:

(c)

$$(\mid^{?!})\frac{(p, H, H') \xrightarrow{x?z} (p', H, H') \qquad (q, H, H') \xrightarrow{x!z} (q', H, H')}{(p \mid q, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H, H')}$$

The derivations of $q \mid p$ are as follows:

(d)

$$(|^{?!})\frac{(q, H, H') \xrightarrow{x!z} (q', H, H') \qquad (p, H, H') \xrightarrow{x?z} (p', H, H')}{(q \mid p, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (q' \,||\, p', H, H')}$$

As demonstrated in (c) and (d), if $(p, H, H') \xrightarrow{x!z} (p', H, H')$ and $(q, H, H') \xrightarrow{x?z} (q', H, H')$ hold then both of the terms $p \mid q$ and $q \mid p$ are able to perform the $\mathbf{rcfg(x,z)}$ transition:

$$\begin{aligned} &(p \mid q, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p' \,||\, q', H, H') \\ &(q \mid p, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (q' \,||\, p', H, H') \end{aligned} \tag{89}$$

Observe that the terms evolve into different expressions and we would now need to check if these terms are bisimilar. According to the axiom $(A6)$, the "$||$" operator is commutative. Hence, the following holds:

$$(p' \,||\, q') \sim (q' \,||\, p') \tag{90}$$

Therefore, by (87), (88), (89) and (90) it is straightforward to conclude that the following holds:

$$(p \mid q) \sim (q \mid p) \tag{91}$$

- Axiom under consideration:

$$p \mid q \equiv \bot \; [owise] \quad (A15) \tag{92}$$

for $p, q \in$ DyNetKAT. Observe that the $[owise]$ condition implies that p cannot be of shape $x?z \,;r$ when q is of shape $x!z\,;r'$, as otherwise the axiom $(A12)$ would become applicable (or vice versa due to commutativity of $|$). Furthermore, note that if p or q contains operators other than sequential composition (;), that is the operators "$\oplus$", "$\lfloor\!\lfloor$" and "$||$", then the axioms such as $(A8)$, $(A10)$ and $(A13)$ would become applicable and hence the $[owise]$ condition would not be met. The axiom $(A15)$ can be written explicitly as follows:

$$(z\,;p) \mid q \equiv \bot \tag{93}$$

$$(x?z\,;p) \mid (x'?z'\,;q) \equiv \bot \tag{94}$$

$$(x!z\,;p) \mid (x'!z'\,;q) \equiv \bot \tag{95}$$

$$(x?z\,;p) \mid (x'!z'\,;q) \equiv \bot \text{ if } x \neq x' \text{ or } z \neq z' \tag{96}$$

$$(\mathbf{rcfg}_{x,z}\,;p) \mid q \equiv \bot \tag{97}$$

for $z, z' \in \text{NetKAT}^{-\mathbf{dup}}$. Observe that the term $\bot$ does not afford any transition and none of the terms on the left hand side of the equivalences above afford any transition as well. Therefore, the following holds if the $[owise]$ condition is met:

$$(p \mid q) \sim \bot \tag{98}$$

- Axiom under consideration:

$$\delta_{\mathcal{L}}(\bot) \equiv \bot \quad (\delta_{\bot}) \tag{99}$$

Observe that according to the semantic rules of DyNetKAT, the terms $\delta_{\mathcal{L}}(\bot)$ and $\bot$ do not afford any transition. Hence, the following trivially holds:

$$(\delta_{\mathcal{L}}(\bot)) \sim \bot \tag{100}$$

- Axiom under consideration:

$$\delta_{\mathcal{L}}(at\,;p) \equiv at\,;\delta_{\mathcal{L}}(p) \text{ if } at \notin \mathcal{L} \quad (\delta_{;}) \tag{101}$$

for $at \in \{\alpha\cdot\pi, x?z, x!z, \mathbf{rcfg}_{x,z}\}$, $z \in \text{NetKAT}^{-\mathbf{dup}}$ and $p \in \text{DyNetKAT}$. In the following, we make a case analysis on the shape of at and show that the terms $\delta_{\mathcal{L}}(at\,;p)$ and $at\,;\delta_{\mathcal{L}}(p)$ are bisimilar. In our analysis we always assume that the condition $at \notin \mathcal{L}$ is satisfied, as otherwise this axiom is not applicable.

Case (1): $at \triangleq \alpha\cdot\pi$

Consider an arbitrary but fixed network packet σ, let $S_{\alpha\pi} \triangleq [\![\alpha\cdot\pi]\!](\sigma::\langle\rangle)$. The derivations of $\delta_{\mathcal{L}}((\alpha\cdot\pi)\,;p)$ are as follows:

(a)

$$\text{For all } \sigma' \in S_{\alpha\pi}: \quad (\delta)\,\dfrac{(\mathbf{cpol}^{\checkmark}_{\_;})\,\dfrac{}{((\alpha\cdot\pi)\,;p, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p, H, \sigma' :: H')}}{(\delta_{\mathcal{L}}((\alpha\cdot\pi)\,;p), \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (\delta_{\mathcal{L}}(p), H, \sigma' :: H)}$$

The derivations of $(\alpha\cdot\pi)\,;\delta_{\mathcal{L}}(p)$ are as follows:

(b)

$$\text{For all } \sigma' \in S_{\alpha\pi}: \quad (\mathbf{cpol}^{\checkmark}_{\_;})\,\dfrac{}{((\alpha\cdot\pi)\,;\delta_{\mathcal{L}}(p), \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (\delta_{\mathcal{L}}(p), H, \sigma' :: H')}$$

As demonstrated in (a) and (b), both of the terms $\delta_{\mathcal{L}}((\alpha\cdot\pi)\,;p)$ and $(\alpha\cdot\pi)\,;\delta_{\mathcal{L}}(p)$ initially afford the same set of transitions of shape (σ,σ') and they converge to the same expression after taking these transitions:

$$\begin{aligned}&(\delta_{\mathcal{L}}((\alpha\cdot\pi)\,;p), \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (\delta_{\mathcal{L}}(p), H, \sigma' :: H')\\&((\alpha\cdot\pi)\,;\delta_{\mathcal{L}}(p), \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (\delta_{\mathcal{L}}(p), H, \sigma' :: H')\end{aligned} \tag{102}$$

Case (2): $at \triangleq x?z$

The derivations of $\delta_{\mathcal{L}}(x?z\,;p)$ are as follows:

(c)

$$(\delta)\,\dfrac{(\mathbf{cpol}_{?})\,\dfrac{}{(x?z\,;p, H, H') \xrightarrow{x?z} (p, H, H')}}{(\delta_{\mathcal{L}}(x?z\,;p), H, H') \xrightarrow{x?z} (\delta_{\mathcal{L}}(p), H, H')}$$

The derivations of $x?z\,;\delta_{\mathcal{L}}(p)$ are as follows:

(d)

$$(\mathbf{cpol}_{?})\,\frac{}{(x?z\,;\delta_{\mathcal{L}}(p), H, H') \xrightarrow{x?z} (\delta_{\mathcal{L}}(p), H, H')}$$

As demonstrated in (c) and (d), both of the terms $\delta_{\mathcal{L}}(x?z\,;p)$ and $x?z\,;\delta_{\mathcal{L}}(p)$ initially only afford the $x?z$ transition and they converge to the same expression after taking this transition:

$$\begin{aligned}(\delta_{\mathcal{L}}(x?z\,;p), H, H') &\xrightarrow{x?z} (\delta_{\mathcal{L}}(p), H, H')\\ (x?z\,;\delta_{\mathcal{L}}(p), H, H') &\xrightarrow{x?z} (\delta_{\mathcal{L}}(p), H, H')\end{aligned} \tag{103}$$

Case (3): $at \triangleq x!z$

The derivations of $\delta_{\mathcal{L}}(x!z\,;p)$ are as follows:

(e)

$$(\delta)\,\frac{(\mathbf{cpol}_{!})\,\frac{}{(x!z\,;p, H, H') \xrightarrow{x!z} (p, H, H')}}{(\delta_{\mathcal{L}}(x!z\,;p), H, H') \xrightarrow{x!z} (\delta_{\mathcal{L}}(p), H, H')}$$

The derivations of $x!z\,;\delta_{\mathcal{L}}(p)$ are as follows:

(f)

$$(\mathbf{cpol}_{!})\,\frac{}{(x!z\,;\delta_{\mathcal{L}}(p), H, H') \xrightarrow{x!z} (\delta_{\mathcal{L}}(p), H, H')}$$

As demonstrated in (e) and (f), both of the terms $\delta_{\mathcal{L}}(x!z\,;p)$ and $x!z\,;\delta_{\mathcal{L}}(p)$ initially only afford the $x!z$ transition and they converge to the same expression after taking this transition:

$$\begin{aligned}(\delta_{\mathcal{L}}(x!z\,;p), H, H') &\xrightarrow{x!z} (\delta_{\mathcal{L}}(p), H, H')\\ (x!z\,;\delta_{\mathcal{L}}(p), \sigma :: H, H') &\xrightarrow{x!z} (\delta_{\mathcal{L}}(p), H, H')\end{aligned} \tag{104}$$

Case (4): $at \triangleq \mathbf{rcfg}_{x,z}$

The derivations of $\delta_{\mathcal{L}}(\mathbf{rcfg}_{x,z}\,;p)$ are as follows:

(g)

$$(\delta)\,\frac{(\mathbf{rcfg_{x,z}})\,\frac{}{(\mathbf{rcfg}_{x,z}\,;p, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p, H, H')}}{(\delta_{\mathcal{L}}(\mathbf{rcfg}_{x,z}\,;p), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (\delta_{\mathcal{L}}(p), H, H')}$$

The derivations of $\mathbf{rcfg}_{x,z}\,;\delta_{\mathcal{L}}(p)$ are as follows:

(h)

$$(\mathbf{rcfg_{x,z}})\,\frac{}{(\mathbf{rcfg}_{x,z}\,;\delta_{\mathcal{L}}(p), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (\delta_{\mathcal{L}}(p), H, H')}$$

As demonstrated in (g) and (h), both of the terms $\delta_{\mathcal{L}}(\mathbf{rcfg}_{x,z}\,;p)$ and $\mathbf{rcfg}_{x,z}\,;\delta_{\mathcal{L}}(p)$ initially only afford the $\mathbf{rcfg(x,z)}$ transition and they converge to the same expression after taking this transition:

$$\begin{aligned}(\delta_{\mathcal{L}}(\mathbf{rcfg}_{x,z}\,;p),H,H') &\xrightarrow{\mathbf{rcfg(x,z)}} (\delta_{\mathcal{L}}(p),H,H')\\ (\mathbf{rcfg}_{x,z}\,;\delta_{\mathcal{L}}(p),H,H') &\xrightarrow{\mathbf{rcfg(x,z)}} (\delta_{\mathcal{L}}(p),H,H')\end{aligned} \tag{105}$$

Therefore, if $at \notin \mathcal{L}$, by (102), (103), (104) and (105) it is straightforward to conclude that the following holds:

$$(\delta_{\mathcal{L}}(at\,;p)) \sim (at\,;\delta_{\mathcal{L}}(p)) \tag{106}$$

- Axiom under consideration:

$$\delta_{\mathcal{L}}(at\,;p) \equiv \bot \text{ if } at \in \mathcal{L} \quad (\delta_{;}^{\bot}) \tag{107}$$

Observe that according to the semantic rules of DyNetKAT, the term $\bot$ do not afford any transition. Furthermore, if the condition $at \in \mathcal{L}$ is satisfied, then the term $\delta_{\mathcal{L}}(at\,;p)$ also does not afford any transition. Therefore, if $at \in \mathcal{L}$, the following trivially holds:

$$\delta_{\mathcal{L}}(at\,;p) \sim \bot \tag{108}$$

- Axiom under consideration:

$$\delta_{\mathcal{L}}(p \oplus q) \equiv \delta_{\mathcal{L}}(p) \oplus \delta_{\mathcal{L}}(q) \quad (\delta_{\oplus}) \tag{109}$$

for $p, q \in$ DyNetKAT. According to the semantic rules of DyNetKAT, the following are the possible transitions that can initially occur in the terms $\delta_{\mathcal{L}}(p \oplus q)$ and $\delta_{\mathcal{L}}(p) \oplus \delta_{\mathcal{L}}(q)$:

$$\begin{cases}(1)\ (p,H_0,H_0') \xrightarrow{\gamma} (p',H_1,H_1')\\ (2)\ (q,H_0,H_0') \xrightarrow{\gamma} (q',H_1,H_1')\end{cases}$$

$\gamma ::= (\sigma,\sigma') \mid x!z \mid x?z \mid \mathbf{rcfg(x,z)}$

Case (1): $(p,H_0,H_0') \xrightarrow{\gamma} (p',H_1,H_1')$

The derivations of $\delta_{\mathcal{L}}(p \oplus q)$ are as follows:

(a)

$$(\delta)\dfrac{(\mathbf{cpol\_}_{\oplus})\dfrac{(p,H_0,H_0') \xrightarrow{\gamma} (p',H_1,H_1')}{(p \oplus q,H_0,H_0') \xrightarrow{\gamma} (p',H_1,H_1')}}{(\delta_{\mathcal{L}}(p \oplus q),H_0,H_0') \xrightarrow{\gamma} (\delta_{\mathcal{L}}(p'),H_1,H_1')}$$

The derivations of $\delta_{\mathcal{L}}(p) \oplus \delta_{\mathcal{L}}(q)$ are as follows:

(b)

$$(\mathbf{cpol\_}_{\oplus})\dfrac{(\delta)\dfrac{(p,H_0,H_0') \xrightarrow{\gamma} (p',H_1,H_1')}{(\delta_{\mathcal{L}}(p),H_0,H_0') \xrightarrow{\gamma} (\delta_{\mathcal{L}}(p'),H_1,H_1')}}{(\delta_{\mathcal{L}}(p) \oplus \delta_{\mathcal{L}}(q),H_0,H_0') \xrightarrow{\gamma} (\delta_{\mathcal{L}}(p'),H_1,H_1')}$$

As demonstrated in (a) and (b), if $(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$ holds then both of the terms $\delta_{\mathcal{L}}(p \oplus q)$ and $\delta_{\mathcal{L}}(p) \oplus \delta_{\mathcal{L}}(q)$ converge to the same expression with the γ transition:

$$\begin{aligned}(\delta_{\mathcal{L}}(p \oplus q), H_0, H_0') &\xrightarrow{\gamma} (\delta_{\mathcal{L}}(p'), H_1, H_1')\\ (\delta_{\mathcal{L}}(p) \oplus \delta_{\mathcal{L}}(q), H_0, H_0') &\xrightarrow{\gamma} (\delta_{\mathcal{L}}(p'), H_1, H_1')\end{aligned} \tag{110}$$

Case (2): $(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')$

The derivations of $\delta_{\mathcal{L}}(p \oplus q)$ are as follows:

(c)

$$(\delta)\dfrac{(\mathbf{cpol}_{\oplus\_})\dfrac{(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}{(p \oplus q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}}{(\delta_{\mathcal{L}}(p \oplus q), H_0, H_0') \xrightarrow{\gamma} (\delta_{\mathcal{L}}(q'), H_1, H_1')}$$

The derivations of $\delta_{\mathcal{L}}(p) \oplus \delta_{\mathcal{L}}(q)$ are as follows:

(d)

$$(\mathbf{cpol}_{\oplus\_})\dfrac{(\delta)\dfrac{(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}{(\delta_{\mathcal{L}}(q), H_0, H_0') \xrightarrow{\gamma} (\delta_{\mathcal{L}}(q'), H_1, H_1')}}{(\delta_{\mathcal{L}}(p) \oplus \delta_{\mathcal{L}}(q), H_0, H_0') \xrightarrow{\gamma} (\delta_{\mathcal{L}}(q'), H_1, H_1')}$$

As demonstrated in (c) and (d), if $(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')$ holds then both of the terms $\delta_{\mathcal{L}}(p \oplus q)$ and $\delta_{\mathcal{L}}(p) \oplus \delta_{\mathcal{L}}(q)$ converge to the same expression with the γ transition:

$$\begin{aligned}(\delta_{\mathcal{L}}(p \oplus q), H_0, H_0') &\xrightarrow{\gamma} (\delta_{\mathcal{L}}(q'), H_1, H_1')\\ (\delta_{\mathcal{L}}(p) \oplus \delta_{\mathcal{L}}(q), H_0, H_0') &\xrightarrow{\gamma} (\delta_{\mathcal{L}}(q'), H_1, H_1')\end{aligned} \tag{111}$$

Therefore, by (110), and (111) it is straightforward to conclude that the following holds:

$$(\delta_{\mathcal{L}}(p \oplus q)) \sim (\delta_{\mathcal{L}}(p) \oplus \delta_{\mathcal{L}}(q)) \tag{112}$$

- Axiom under consideration:

$$\pi_0(p) \equiv \bot \quad (\Pi_0) \tag{113}$$

for $p \in$ DyNetKAT. Observe that according to the semantic rules of DyNetKAT, the terms $\pi_0(p)$ and $\bot$ do not afford any transition. Hence, the following trivially holds:

$$\pi_0(p) \sim \bot \tag{114}$$

- Axiom under consideration:

$$\pi_n(\bot) \equiv \bot \quad (\Pi_\bot) \tag{115}$$

for $n \in \mathbb{N}$. Observe that according to the semantic rules of DyNetKAT, the terms $\pi_0(\bot)$ and $\bot$ do not afford any transition. Hence, the following trivially holds:

$$\pi_n(\bot) \sim \bot \tag{116}$$

- Axiom under consideration:

$$\pi_{n+1}(at\,;p) \equiv at\,;\pi_n(p) \quad (\Pi_;) \tag{117}$$

for $at \in \{\alpha \cdot \pi, x?z, x!z, \mathbf{rcfg}_{x,z}\}$, $z \in \text{NetKAT}^{-\mathbf{dup}}$, $n \in \mathbb{N}$ and $p \in \text{DyNetKAT}$. In the following, we make a case analysis on the shape of at and show that the terms $\pi_{n+1}(at\,;p)$ and $at\,;\pi_n(p)$ are bisimilar.

Case (1): $at \triangleq \alpha \cdot \pi$

Consider an arbitrary but fixed network packet σ, let $S_{\alpha\pi} \triangleq [\![\alpha \cdot \pi]\!](\sigma{::}\langle\rangle)$. The derivations of $\pi_{n+1}((\alpha\cdot\pi)\,;p)$ are as follows:

(a)

$$\text{For all } \sigma' \in S_{\alpha\pi}: \quad (\pi)\dfrac{(\mathbf{cpol}^{\checkmark}_{\_;})\dfrac{}{((\alpha\cdot\pi)\,;p, \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (p, H, \sigma' :: H')}}{(\pi_{n+1}((\alpha\cdot\pi)\,;p), \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (\pi_n(p), H, \sigma' :: H)}$$

The derivations of $(\alpha\cdot\pi)\,;\pi_n(p)$ are as follows:

(b)

$$\text{For all } \sigma' \in S_{\alpha\pi}: \quad (\mathbf{cpol}^{\checkmark}_{\_;})\dfrac{}{((\alpha\cdot\pi)\,;\pi_n(p), \sigma :: H, H') \xrightarrow{(\sigma,\sigma')} (\pi_n(p), H, \sigma' :: H')}$$

As demonstrated in (a) and (b), both of the terms $\pi_{n+1}((\alpha\cdot\pi)\,;p)$ and $(\alpha\cdot\pi)\,;\pi_n(p)$ initially only afford the same set of transitions of shape (σ,σ') and they converge to the same expression after taking these transitions:

$$\begin{aligned}(\pi_{n+1}((\alpha\cdot\pi)\,;p), \sigma :: H, H') &\xrightarrow{(\sigma,\sigma')} (\pi_n(p), H, \sigma' :: H')\\ ((\alpha\cdot\pi)\,;\pi_n(p), \sigma :: H, H') &\xrightarrow{(\sigma,\sigma')} (\pi_n(p), H, \sigma' :: H')\end{aligned} \tag{118}$$

Case (2): $at \triangleq x?z$

The derivations of $\pi_{n+1}(x?z\,;p)$ are as follows:

(c)

$$(\pi)\dfrac{(\mathbf{cpol}_?)\dfrac{}{(x?z\,;p, \sigma :: H, H') \xrightarrow{x?z} (p, H, H')}}{(\pi_{n+1}(x?z\,;p), H, H') \xrightarrow{x?z} (\pi_n(p), H, H')}$$

The derivations of $x?z\,;\delta_{\mathcal{L}}(p)$ are as follows:

(d)

$$(\mathbf{cpol}_?)\dfrac{}{(x?z\,;\pi_n(p), H, H') \xrightarrow{x?z} (\pi_n(p), H, H')}$$

As demonstrated in (c) and (d), both of the terms $\pi_{n+1}(x?z\,;p)$ and $x?z\,;\pi_n(p)$ initially only afford the $x?z$ transition and they converge to the same expression after taking this transition:

$$\begin{aligned}(\pi_{n+1}(x?z\,;p), H, H') &\xrightarrow{x?z} (\pi_n(p), H, H')\\ (x?z\,;\pi_n(p), H, H') &\xrightarrow{x?z} (\pi_n(p), H, H')\end{aligned} \tag{119}$$

Case (3): $at \triangleq x!z$

The derivations of $\pi_{n+1}(x!z\,;p)$ are as follows:

(e)

$$\dfrac{(\mathbf{cpol}_!)\dfrac{}{(x!z\,;p, H, H') \xrightarrow{x!z} (p, H, H')}}{(\pi)\ (\pi_{n+1}(x!z\,;p), H, H') \xrightarrow{x!z} (\pi_n(p), H, H')}$$

The derivations of $x!z\,;\pi_n(p)$ are as follows:

(f)

$$(\mathbf{cpol}_!)\dfrac{}{(x!z\,;\pi_n(p), H, H') \xrightarrow{x!z} (\pi_n(p), H, H')}$$

As demonstrated in (e) and (f), both of the terms $\pi_{n+1}(x!z\,;p)$ and $x!z\,;\pi_n(p)$ initially only afford the $x!z$ transition and they converge to the same expression after taking this transition:

$$\begin{aligned}(\pi_{n+1}(x!z\,;p), H, H') &\xrightarrow{x!z} (\pi_n(p), H, H')\\ (x!z\,;\pi_n(p), H, H') &\xrightarrow{x!z} (\pi_n(p), H, H')\end{aligned} \tag{120}$$

Case (4): $at \triangleq \mathbf{rcfg}_{x,z}$

The derivations of $\pi_{n+1}(\mathbf{rcfg}_{x,z}\,;p)$ are as follows:

(g)

$$\dfrac{(\mathbf{rcfg_{x,z}})\dfrac{}{(\mathbf{rcfg}_{x,z}\,;p, H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (p, H, H')}}{(\delta)\ (\pi_{n+1}(\mathbf{rcfg}_{x,z}\,;p), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (\pi_n(p), H, H')}$$

The derivations of $\mathbf{rcfg}_{x,z}\,;\pi_n(p)$ are as follows:

(h)

$$(\mathbf{rcfg_{x,z}})\dfrac{}{(\mathbf{rcfg}_{x,z}\,;\pi_n(p), H, H') \xrightarrow{\mathbf{rcfg(x,z)}} (\pi_n(p), H, H')}$$

As demonstrated in (g) and (h), both of the terms $\pi_{n+1}(\mathbf{rcfg}_{x,z}\,;p)$ and $\mathbf{rcfg}_{x,z}\,;\pi_n(p)$ initially only afford the $\mathbf{rcfg}(\mathbf{x},\mathbf{z})$ transition and they converge to the same expression after taking this transition:

$$\begin{aligned}(\pi_{n+1}(\mathbf{rcfg}_{x,z}\,;p), H, H') &\xrightarrow{\mathbf{rcfg(x,z)}} (\pi_n(p), H, H')\\ (\mathbf{rcfg}_{x,z}\,;\pi_n(p), H, H') &\xrightarrow{\mathbf{rcfg(x,z)}} (\pi_n(p), H, H')\end{aligned} \tag{121}$$

Therefore, if $at \notin \mathcal{L}$, by (118), (119), (120) and (121) it is straightforward to conclude that the following holds:

$$(\pi_{n+1}(at\,;p)) \sim (at\,;\pi_n(p)) \tag{122}$$

- Axiom under consideration:

$$\pi_n(p \oplus q) \equiv \pi_n(p) \oplus \pi_n(q) \quad (\pi\oplus) \tag{123}$$

for $p, q \in$ DyNetKAT. Observe that if $n = 0$, then both of the terms do not afford any transition and bisimilarity holds trivially. If $n > 0$, according to the semantic rules of DyNetKAT, the following are the possible transitions that can initially occur in the terms $\pi_n(p \oplus q)$ and $\pi_n(p) \oplus \pi_n(q)$:

$$\begin{cases} (1)\ (p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1') \\ (2)\ (q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1') \end{cases}$$

$\gamma ::= (\sigma, \sigma') \mid x!z \mid x?z \mid \mathbf{rcfg(x, z)}$

Case (1): $(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$

The derivations of $\pi_n(p \oplus q)$ are as follows:

(a)

$$(\pi)\dfrac{(\mathbf{cpol}_{\_\oplus})\dfrac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(p \oplus q, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}}{(\pi_n(p \oplus q), H_0, H_0') \xrightarrow{\gamma} (\pi_{n-1}(p'), H_1, H_1')}$$

The derivations of $\pi_n(p) \oplus \pi_n(q)$ are as follows:

(b)

$$(\mathbf{cpol}_{\_\oplus})\dfrac{(\pi)\dfrac{(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')}{(\pi_n(p), H_0, H_0') \xrightarrow{\gamma} (\pi_{n-1}(p'), H_1, H_1')}}{(\pi_n(p) \oplus \pi_n(q), H_0, H_0') \xrightarrow{\gamma} (\pi_{n-1}(p'), H_1, H_1')}$$

As demonstrated in (a) and (b), if $(p, H_0, H_0') \xrightarrow{\gamma} (p', H_1, H_1')$ holds then both of the terms $\pi_n(p \oplus q)$ and $\pi_n(p) \oplus \pi_n(q)$ converge to the same expression with the γ transition:

$$\begin{aligned} (\pi_n(p \oplus q), H_0, H_0') &\xrightarrow{\gamma} (\pi_{n-1}(p'), H_1, H_1') \\ (\pi_n(p) \oplus \pi_n(q), H_0, H_0') &\xrightarrow{\gamma} (\pi_{n-1}(p'), H_1, H_1') \end{aligned} \tag{124}$$

Case (2): $(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')$

The derivations of $\pi_n(p \oplus q)$ are as follows:

(c)

$$(\pi)\dfrac{(\mathbf{cpol}_{\oplus\_})\dfrac{(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}{(p \oplus q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}}{(\pi_n(p \oplus q), H_0, H_0') \xrightarrow{\gamma} (\pi_{n-1}(q'), H_1, H_1')}$$

The derivations of $\pi_n(p) \oplus \pi_n(q)$ are as follows:

(d)

$$(\mathbf{cpol}_{\oplus\_})\frac{(\pi)\dfrac{(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')}{(\pi_n(q), H_0, H_0') \xrightarrow{\gamma} (\pi_{n-1}(q'), H_1, H_1')}}{(\pi_n(p) \oplus \pi_n(q), H_0, H_0') \xrightarrow{\gamma} (\pi_{n-1}(q'), H_1, H_1')}$$

As demonstrated in (c) and (d), if $(q, H_0, H_0') \xrightarrow{\gamma} (q', H_1, H_1')$ holds then both of the terms $\pi_n(p \oplus q)$ and $\pi_n(p) \oplus \pi_n(q)$ converge to the same expression with the γ transition:

$$\begin{aligned} (\pi_n(p \oplus q), H_0, H_0') &\xrightarrow{\gamma} (\pi_{n-1}(q'), H_1, H_1') \\ (\pi_n(p) \oplus \pi_n(q), H_0, H_0') &\xrightarrow{\gamma} (\pi_{n-1}(q'), H_1, H_1') \end{aligned} \tag{125}$$

Therefore, by (124) and (125) it is straightforward to conclude that the following holds:

$$(\pi_n(p \oplus q)) \sim (\pi_n(p) \oplus \pi_n(q)) \tag{126}$$