Causality in Event Structure

Amir Hossein Seyhani

June 2022

Contents

1 Introduction **2**

2 Preliminaries **2**
2.1 Event Structure 2
2.2 Causal Model 6
2.3 Causal Network 7
2.4 Definition of Actual Cause 7
2.5 Actual Cause in Non-Recursive Models 8
2.6 Extended Causal Model 9

3 Semantics of Communicating Processes **10**
3.1 Nil 10
3.2 Prefix 10
3.3 Sum 11
3.4 Product 11
3.5 Restriction 13
3.6 Relabelling 13
3.7 Denotational Semantics 13

4 Causality in Event Structure **14**
4.1 Causality in DyNetKAT 16
4.2 Examples 16

Appendices **18**

1 Introduction

2 Preliminaries

2.1 Event Structure

Definition 2.1.1 (Event Structure)**.** An event structure is a triple $E = (\mathcal{E}, \#, \vdash)$ where:

1. $\mathcal{E}$ is a set of events
2. $\#$ is a binary symmetric, irreflexive relation on $\mathcal{E}$, the conflict relation. We shall write Con for the set of conflict-free subsets of $\mathcal{E}$, i.e. those finite subsets $X \subseteq \mathcal{E}$ for which: $\forall e, e' \in X. \neg(e \# e')$
3. $\vdash \subseteq Con \times \mathcal{E}$ is the enabling relation which satisfies: $X \vdash e \ \& \ X \subseteq Y \in Con \Rightarrow Y \vdash e$

Notion 1. In an event structure we shall write $\Subset$ for the reflexive conflict relation by which we mean that $e \Subset e'$ in an event structure iff either $e \# e'$ or $e = e'$. With this notion instead of describing the conflict-free sets of an event structure as those sets X such that

$$\forall e, e' \in X. \neg(e \# e')$$

we can say they are those sets X for which:

$$\forall e, e' \in X. e \Subset e' \Rightarrow e = e'$$

Notion 2. For any event structure we can define the minimal enabling relation $\vdash_{min}$ by:

$$X \vdash_{min} e \iff X \vdash e \ \& \ (\forall Y \subseteq X. Y \vdash e \Rightarrow Y = X)$$

Then for any event structure:

$$Y \vdash e \Rightarrow \exists X \subseteq Y. X \vdash_{min} e$$

Definition 2.1.2 (Configuration)**.** Let $E = (\mathcal{E}, \#, \vdash)$ be an event structure. Define a configuration of E to be a subset of events $x \subseteq \mathcal{E}$ which is

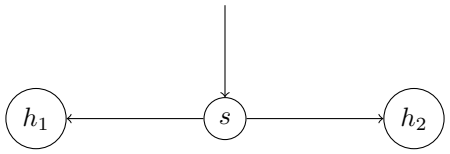
1. conflict-free: $x \in Con$
2. secured: $\forall e \in x \exists e_0, ..., e_n \in x. e_n = e \ \& \ \forall i \leq n. \{e_0, ..., e_{i-1}\} \vdash e_i$

The set of all configurations of an event structure is written as $\mathcal{F}(\mathcal{E})$. It is helpful to unwrap condition (2) a little. It says an event e is secured in a set x iff there is a sequence of events $e_0, ..., e_n = e$ in x such that:

$$\varnothing \vdash e_0, \{e_0\} \vdash e_1, ..., \{e_0, ..., e_{i-1}\} \vdash e_i, ..., \{e_0, ..., e_{n-1}\} \vdash e_n.$$

We call such a sequence $e_0, e_1, ..., e_n = e$ a *securing* for e in x. We use $X \subseteq_{fin} Y$ to mean X is a finite subset of Y.

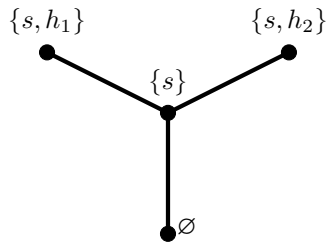
Example 2.1.1. Consider a network with two hosts h_1 and h_2 that are connected to a single switch. Assume that a packet arrives at the switch and then the switch non-deterministically forwards the packets to either h_1 or h_2.



To model this scenario in event structures we can define an event p for the arrival of the packet at the switch and events h_1 and h_2 for forwarding the packet to the hosts. The enabling relation is the least one for which we have:

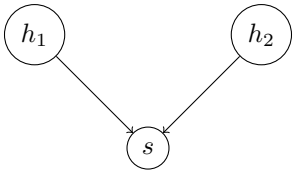
$$\varnothing \vdash_{min} s, \{s\} \vdash_{min} h_1, \{s\} \vdash_{min} h_2,$$

And we consider the conflict relation the least one that satisfies $h_1 \# h_2$. We illustrate configurations of event structures using the Hasse diagram of the partial order of configurations ordered by inclusion:



We can see that non-determinism appears as branching in the partial order of configurations.

Example 2.1.2. Consider another network in which two hosts are connected to a single switch. This time both hosts are sending a packet concurrently to the host. Once the switch received a packet from either host it will drop the packets received from the other host due to capacity limit.

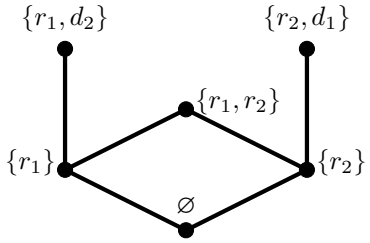


Let r_1, r_2 represent the events of receiving packet from h_1 and h_2 and d_1, d_2 represent the events of dropping packets from these hosts. We need a conflict relation the least one for which we have $r_1 \# d_1$ and $r_2 \# d_2$ and enabling relation the least one that satisfies:

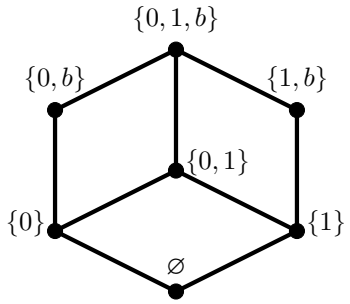
$$\varnothing \vdash_{min} r_1$$

$$\begin{aligned} \varnothing &\vdash_{min} r_2 \\ \{r_1\} &\vdash_{min} d_2 \\ \{r_2\} &\vdash_{min} d_1 \end{aligned}$$

Thus the configurations have the form:



Example 2.1.3. Consider two parallel switches in a circuit where they are connected to a light bulb. An event may be enabled in more than one way even in a single configuration. Assume initially both switches are open. Closing either one enables the event of the bulb lighting up. The configurations have the form:



We look for a special class of event structures for which there is a the partial order of causal dependency on each configuration. This can not be done so obviously for all event structures. Consider the event structure of the previous example in which the event b causally depends not on a unique set of events but rather on either the occurrence of 0 or on the occurrence of 1. It is incorrect to say b causally depends on both 0 and 1 because the occurrence of only one of them enables the occurrence of b. The difficulty arises because there is a configuration $\{0, 1, b\}$ in which there is an event b which is not enabled by a unique minimal set of event occurrences. We can rule out such possibilities by insisting on event structures satisfy the following stability axiom.

Definition 2.1.3 (Stable Event Structure)**.** Let $E = (\mathcal{E}, \#, \vdash)$ be an event structure. Say E is stable if it satisfies the following axiom:

$$X \vdash e \;\&\; Y \vdash e \;\&\; X \cup Y \cup \{e\} \in Con \Rightarrow X \cap Y \vdash e$$

The stability axiom ensures that an event in a configuration is enabled in an essentially unique way. Assume e belongs to a configuration x of a stable event structure. Suppose $X \vdash e$ and $X \subseteq x$. Then $X \cup \{e\} \in Con$, the enabling $X \vdash e$ is consistent. Take

$$X_0 = \cap\{Y | Y \subseteq X \ \& \ Y \vdash e\}$$

Because X is finite this is an intersection of a finite number of sets and we see by the stability axiom that $X_0 \vdash e$. Moreover X_0 is the unique minimal subset of X which enables e. Thus for stable event structures, we have:

$$Y \vdash e \ \& \ Y \cup \{e\} \in Con \Rightarrow \exists! X \subseteq Y . X \vdash_{min} e$$

It follows that for stable event structures

$$X \vdash_{min} e \ \& \ Y \vdash_{min} \ \& \ X \cup Y \cup e \in Con \Rightarrow X = Y$$

Theorem 1. Let $E = (\mathcal{E}, \#, \vdash)$ be an event structure. Let $x = \{e_1, e_2, ..., e_n\}$ be a conflict-free subset of $\mathcal{E}$. Then x is secured according to the definition 2.1.2 iff there exists an event $e_{i_n} \in x$ with a securing sequence $e_{i_1}, e_{i_2}, ..., e_{i_{n-1}}$.

Proof. Let x be secured. We construct a securing sequence of length n. Pick two arbitrary events $e, e' \in x$. Since x is secured there exists some securing sequences $s = e_{i_1}, e_{i_2}, ..., e_{i_m}$ and $s' = e_{i'_1}, e_{i'_2}, ..., e_{i'_{m'}}$ for e and e' respectively where $e_{i_m} = e$ and $e_{i'_{m'}} = e'$. For $1 \leq a \leq b \leq p$, let $s'' = e_{j_1}, e_{j_2}, ..., e_{j_p}$ be the sequence resulting from appending s' to s. Let $es(a, b) = \{e_{j_a}, e_{j_{a+1}}, ..., e_{j_b}\}$. Pick an arbitrary event e_{j_q} with $q > m$. In order for s'' to be a securing sequence, we must have $es(1, j_{p-1}) \vdash e_{j_p}$. Since we have $es(j_{m+1}, j_{p-1}) \vdash e_{j_p}$ and $es(j_{m+1}, j_{p-1}) \subseteq es(1, j_{p-1})$ thus we can conclude that $es(1, j_{p-1}) \vdash e_{j_p}$. Now assume that there exists an event such as e_{j_q} with $q > m$ for which we have $\exists q'.q' \leq m \wedge j_{q'} = j_q$. Let $es'(1, j_{p-1})$ be the set of events preceding e_{j_p}, if we remove e_{j_q} from s''. We still have $e_{j_{q'}} \in es'(1, j_{p-1})$ thus we can conclude that $es'(1, j_{p-1}) = es(1, j_{p-1})$ so $es'(1, j_{p-1}) \vdash e_{j_p}$. So, we can remove any duplicate event after e_{j_m} and still have a securing sequence. This proves that we can construct a securing sequence by appending the securing sequence of two arbitrary events. So, we can first pick two arbitrary events and construct the resulting in securing sequence and then repeating this procedure until we construct the full securing sequence.

Now assume that there exists an event e_{i_n} with a securing sequence $e_{i_1}, e_{i_2}, ..., e_{i_{n-1}}$. Regarding the definition of securing sequence each event has a securing sequence, thus x is secured. □

Definition 2.1.4. Let $E_0 = (\mathcal{E}_0, \#_0, \vdash_0)$ and $E_1 = (\mathcal{E}_1, \#_1, \vdash_1)$ be event Structures. Define

$$\begin{aligned} E_0 \trianglelefteq E_1 \iff & \mathcal{E}_0 \subseteq \mathcal{E}_1, \\ & \forall e, e'.e \#_0 e' \iff e, e' \in \mathcal{E}_0 \ \& \ e \#_1 e' \text{ and} \\ & \forall X, e.X \vdash_0 e \iff X \subseteq \mathcal{E}_0 \ \& \ e \in \mathcal{E}_0 \ \& \ X \vdash_1 e \end{aligned}$$

In this case say E_0 is a substructure of E_1.

Definition 2.1.5 (Restriction)**.** Let $E = (\mathcal{E}, \#, \vdash)$ be an event structure. Let $A \subseteq \mathcal{E}$. Define the restriction of E to A to be

$$E\lceil A = (A, \#_A, \vdash_A)$$

where

$$X \in Con_A \iff X \subseteq A \ \& \ X \in Con$$
$$X \vdash_A e \iff X \subseteq A \ \& \ e \in A \ \& \ X \vdash e$$

Definition 2.1.6. Let a be an event. For an event structure $E = (\mathcal{E}, \#, \vdash)$ define aE to be the event structure $(\mathcal{E}', \#', \vdash')$ where:

$\mathcal{E}' = \{(0, a)\} \cup \{(1, e) | e \in \mathcal{E}\}$,
$e'_0 \#' e'_1 \iff \exists e_0, e_1 . e'_0 = (1, e_0) \ \& \ e'_1 = (1, e_1) \ \& \ e_0 \# e_1$
$X \vdash' e' \iff e' = (0, a)$ or $[e' = (1, e_1) \ \& \ (0, a) \in X \ \& \ \{e | (1, e) \in X\} \vdash e_1]$

Definition 2.1.7. A labelled event structure consists of $(\mathcal{E}, \#, \vdash, L, l)$ where $(\mathcal{E}, \#, \vdash)$ is an event structure, L is a set of labels, not including the element *, and l is a function $l : \mathcal{E} \to L$ from its events to its labels.

Notion 3. We write $(a_1, a_2, ..., a_{n-1}, a_n)$ to denote the event:

$$e = (a_1, (a_2, (a_3, ...(a_{n-1}, a_n))))$$

2.2 Causal Model

A signature $\mathcal{S}$ is a tuple $(\mathcal{U}, \mathcal{V}, \mathcal{R})$, where $\mathcal{U}$ is a set of exogenous variables, $\mathcal{V}$ is a set of endogenous variables, and R associates with every variable $Y \in \mathcal{U} \cup \mathcal{V}$ a nonempty set $\mathcal{R}(Y)$ of possible values for Y. A causal model (or structural model) over signature S is a tuple $M = (\mathcal{S}, \mathcal{F})$, where $\mathcal{F}$ associates with each variable $X \in \mathcal{V}$ a function denoted F_X such that $F_X : (\times_{U \in \mathcal{U}} \mathcal{R}(U)) \times (\times_{Y \in \mathcal{V} - \{X\}} \mathcal{R}(Y)) \to \mathcal{R}(X)$.

F_X determines the value of X given the values of all the other variables in $\mathcal{U} \cup \mathcal{V}$. For example, if $F_X(Y, Z, U) = Y + U$ (which we usually write as $X = Y + U$), then if $Y = 3$ and $U = 2$, then $X = 5$, regardless of how Z is set. These equations can be thought of as representing processes (or mechanisms) by which values are assigned to variables. Hence, like physical laws, they support a counterfactual interpretation. For example, the equation above claims that in the context $U = u$, if Y were 4, then X would be $u + 4$ (which we write as $(M, u) \models [Y \leftarrow 4](X = u + 4)$), regardless of what values X, Y, and Z actually take in the real world.

The function $\mathcal{F}$ defines a set of (*modifiable*) *structural equations* relating to the values of the variables.

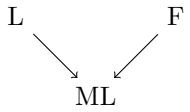
Example 2.2.1. Suppose that we want to reason about a forest fire that could be caused by either lightning or a match lit by an arsonist. Then the causal model would have the following endogenous variables :

- F for fire
- L for lighting
- ML for match lit

The set $\mathcal{U}$ of exogenous variables includes conditions that suffice to render all relationships deterministic (such as whether the wood is dry, whether there is enough oxygen in the air for the match to light, etc.). Suppose that $\vec{u}$ is a setting of the exogenous variables that makes a forest fire possible (i.e., the wood is sufficiently dry, there is oxygen in the air, and so on). Then, for example, $F_F(\vec{u}, L, ML)$ is such that $F = 1$ if either $L = 1$ or $ML = 1$. Note that although the value of F depends on the value L and ML, the value of L does not depend on the values of F and ML.

2.3 Causal Network

We can describe a causal model $\mathcal{M}$ using a causal network. This is a graph with nodes corresponding to the random variables in $\mathcal{V}$ and an edge from a node labeled X to one labeled Y if F_Y depends on the value of X. This graph is a dag that follows from the assumption that the equations are recursive. We occasionally omit the exogenous variables $\vec{U}$ from the causal network. For example, the causal network for example 2.2.1 has the following form:



2.4 Definition of Actual Cause

Given a signature $S = (\mathcal{U}, \mathcal{V}, \mathcal{R})$, a formula of the form $X = x$, for $X \in \mathcal{V}$ and $x \in \mathcal{R}(X)$, is called a *primitive event*. A *basic causal formula* is one of the form $[Y_1 \leftarrow y_, ..., Y_l \leftarrow y_k]\varphi$, where φ is a Boolean combination of primitive events, $Y_1, ..., Y_k$ are distinct variables in $\mathcal{V}$, and $y_i \in \mathcal{R}(Y_i)$. Such a formula is abbreviated as $[\vec{Y} \leftarrow \vec{y}]\varphi$. A *causal formula* is a Boolean combination of basic causal formulas. A causal formula ψ is true or false in a causal model, given a context. We write $(M, \vec{u}) \models \psi$ if ψ is true in causal model M given context $\vec{u}$. $(M, \vec{u}) \models [\vec{Y} \leftarrow \vec{y}](X = x)$ if the variable X has value x in the unique solution to the equation in $M_{\vec{Y} \leftarrow \vec{y}}$ in context $\vec{u}$. The context and structural equations are given. They encode the background knowledge. All relevant events are known. The only question is picking out which of them are the cause of φ or, alternatively, testing whether a given set of events can be considered the cause of φ. The types of events that we allow as actual causes are ones of the form $X_1 = x_1 \wedge ... \wedge X_k = x_k$– that is, conjunctions of primitives events. We abbreviate this as $\vec{X} = \vec{x}$.

Definition 2.4.1. $\vec{X} = \vec{x}$ is an actual cause of φ in $(M, \vec{u})$ if the following three conditions hold:

- **AC1.** $(M, \vec{u}) \models (\vec{X} = \vec{x}) \wedge \varphi$. (both $\vec{X} = \vec{x}$ and φ are true in actual world)
- **AC2.** There exists a partition $(\vec{Z}, \vec{W})$ of $\mathcal{V}$ with $\vec{X} \subseteq \vec{Z}$ and some setting $(\vec{x}', \vec{w}')$ of the variables in $(\vec{X}, \vec{W})$ such that if $(M, \vec{u}) \models \vec{Z} = z^*$ for all $Z \in \vec{Z}$, then both of the following conditions hold:

  (a) $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}']\neg\varphi$.

  (b) $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W}' \leftarrow \vec{w}', \vec{Z}' \leftarrow \vec{z}^*]\varphi$ for all subsets $\vec{W}'$ of $\vec{W}$ and all subsets Z' of $\vec{Z}$.
- **AC3.** $\vec{X}$ is minimal; no subset of $\vec{X}$ satisfies conditions $AC1$ and $AC2$.

We call the tuple $(\vec{W}, \vec{w}, \vec{x}')$ a witness to the fact that $\vec{X} = \vec{x}$ is a cause of φ.

Definition 2.4.2. We say $\vec{X} = \vec{x}$ is a but-for cause of φ in $(M, \vec{u})$ if there exists a witness $(\vec{W}, \vec{w}, \vec{x}')$ for $\vec{X} = \vec{x}$ where $\vec{W} = \varnothing$.

Note that, if we consider a witness $(\vec{W}, \vec{w}, \vec{x}')$ for checking whether $\vec{X} = \vec{x}$ is a cause of φ in $(M, \vec{u})$ where $\vec{W} = \varnothing$, then in the AC2(b) condition we only need to check whether $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{Z}' \leftarrow \vec{z}^*]\varphi$ for all subsets $\vec{Z}'$ of $\vec{Z}$. Since we have $(M, \vec{u}) \models (\vec{X} = \vec{x})$ and $(M, \vec{u}) \models Z = z^*$ for all $Z \in vecZ$, the Interventions $\vec{X} \leftarrow \vec{x}$ and $\vec{Z}' \leftarrow \vec{z}^*$ actually do not change the value of any variable thus checking whether $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{Z}' \leftarrow \vec{z}^*]\varphi$ is true reduces to check whether $(M, \vec{u}) \models \varphi$ which must be already satisfied when we have checked AC1 condition. This means that, to check whether $\vec{X} = \vec{x}$ is an actual cuase when using a witness with an empty $\vec{W}$ we only need to check AC1 and AC2(a) conditions.

2.5 Actual Cause in Non-Recursive Models

In non-recursive models, there may be more than one solution to an equation in a given context, or there may be none. In particular, that means that a context no longer necessarily determines the values of endogenous variables. We identified a primitive event such as $X = x$ with basic causal formula $[](X = x)$, that is, with the special case of a formula of the form $[Y_1 \leftarrow y_1, ..., Y_k \leftarrow y_k]\varphi$ with $k = 0$. We say $(M, \vec{u}) \models [](X = x)$ if $X = x$ in all solutions to the equations where $\vec{U} = \vec{u}$. If seems reasonable to identify $[](X = x)$ with $X = x$ if there is a unique solution to these equations. But it is not so reasonable if there may be several solutions, or no solution. What we really want to do is to be able to say that $X = x$ under a particular setting of variables. Thus, we now take the truth of a primitive event such as $X = x$ relative not just to a context, but to a complete description $(\vec{u}, \vec{v})$ of the values of both the exogenous and the endogenous variables. That is, $(M, \vec{u}, \vec{v}) \models X = x$ if X has value x in $\vec{v}$. Since truth value of $X = x$ depends on just $\vec{v}$, not $\vec{u}$, we sometimes write $(M, \vec{v}) \models X = x$. We then define $(M, \vec{u}, \vec{v}) \models [\vec{Y} \leftarrow \vec{y}]\varphi$ if $(M, \vec{v}') \models \varphi$ for all solutions $(\vec{u}, \vec{v}')$

to the equations in $M_{\vec{Y}\leftarrow\vec{y}}$. Since the truth of $[\vec{Y} \leftarrow \vec{y}](X = x)$ depends only on the context $\vec{u}$ and not on $\vec{v}$, we typically write $(M, \vec{u}) \models [\vec{Y} \leftarrow \vec{y}](X = x)$.

The formula $\langle\vec{Y} \leftarrow \vec{y}\rangle(X = x)$ is the dual of $[\vec{Y} \leftarrow \vec{y}](X = x)$; that is, it is an abbreviation of $\neg[\vec{Y} \leftarrow \vec{y}](X \neq x)$. It is easy to check that $(M, \vec{u}, \vec{v}) \models \langle\vec{Y} \leftarrow \vec{y}\rangle(X = x)$ if in some solution to the equations in $M_{\vec{Y}\leftarrow\vec{y}}$ in context $\vec{u}$, the variable X has value x. For recursive models, it is immediate that $[\vec{Y} \leftarrow \vec{y}](X = x)$ is equivalent to $\langle\vec{Y} \leftarrow \vec{y}\rangle(X = x)$ since all equations have exactly one solution. Now we can state the definition of causality for arbitrary models.

Definition 2.5.1. $\vec{X} = \vec{x}$ is an actual cause of φ in $(M, \vec{u}, \vec{v})$ if the following three conditions hold.

- **AC1.** $(M, \vec{v}) \models (\vec{X} = \vec{x}) \wedge \varphi$.
- **AC2.** There exists a partition $(\vec{Z}, \vec{W})$ of $\mathcal{V}$ with $\vec{X} \subseteq \vec{Z}$ and some setting $(\vec{x}', \vec{w}')$ of the variables in $(\vec{X}, \vec{W})$ such that if $(M, \vec{u}, \vec{v}) \models \vec{Z} = z^*$ for all $Z \in \vec{Z}$, then both of the following conditions hold:

  (a) $(M, \vec{u}) \models \langle\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}'\rangle\neg\varphi$.

  (b) $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W}' \leftarrow \vec{w}', \vec{Z}' \leftarrow \vec{z}^*]\varphi$ for all subsets $\vec{W}'$ of $\vec{W}$ and all subsets Z' of $\vec{Z}$.
- **AC3.** $\vec{X}$ is minimal; no subset of $\vec{X}$ satisfies conditions $AC1$ and $AC2$.

2.6 Extended Causal Model

An extended causal model is a tuple $(\mathcal{S}, \mathcal{F}, \mathcal{E})$, where $(\mathcal{S}, \mathcal{F})$ is a causal model, and $\mathcal{E}$ is a set of allowable settings for the endogenous variables. That is, if the endogenous variables are $X_1, ..., X_n$ then $(x_1, ..., x_n) \in \mathcal{E}$ if $X_1 = x_1, ..., X_n = x_n$ is an allowable setting. We say that a setting of a subset of the endogenous variables is allowable if it can be extended to a setting in $\mathcal{E}$. We then modify the clauses AC2(a) and (b) in the definition of causality to restrict to allowable settings.

Definition 2.6.1. $\vec{X} = \vec{x}$ is an actual cause of φ in $(M, \vec{u})$ if the following three conditions hold:

- **AC1.** $(M, \vec{u}) \models (\vec{X} = \vec{x}) \wedge \varphi \wedge \vec{v} \in \mathcal{E}$.
- **AC2.** There exists a partition $(\vec{Z}, \vec{W})$ of $\mathcal{V}$ with $\vec{X} \subseteq \vec{Z}$ and some setting $(\vec{x}', \vec{w}')$ of the variables in $(\vec{X}, \vec{W})$ such that if $(M, \vec{u}) \models \vec{Z} = z^*$ for all $Z \in \vec{Z}$, then both of the following conditions hold:

  (a) $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}']\neg\varphi \wedge \vec{v} \in \mathcal{E}$.

  (b) $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W}' \leftarrow \vec{w}', \vec{Z}' \leftarrow \vec{z}^*]\varphi \wedge \vec{v} \in \mathcal{E}$ for all subsets $\vec{W}'$ of $\vec{W}$ and all subsets Z' of $\vec{Z}$.
- **AC3.** $\vec{X}$ is minimal; no subset of $\vec{X}$ satisfies conditions $AC1$ and $AC2$.

Where $\vec{v}$ is the value of endogenous variables.

3 Semantics of Communicating Processes

One use of event structure is to give denotational semantics of the language of parallel processes that reflect the parallelism in processes as causal independence between events. The nature of the events, how they interact with the environment, is specified in the language by associating each event with a label from the synchronization algebra L. The language we shall use is one where processes communicate by events of synchronization with no value passing. Its syntax has the form:

$$p ::= nil|\alpha p|p_0 + p_1|p_0 \times p_1|p\lceil\Lambda|p[\Xi]|x|recx.p$$

where x is in some set of variables X over processes, α is a label, Λ is a subset of labels, in $p[\Xi]$ the symbol Ξ denotes a relabelling function between two sets of labels.

Informally, the product $p_0 \times p_1$ is a form of parallel composition which introduces arbitrary events of synchronization between processes. Unwanted synchronizations can be restricted away with the help of the restriction operation $p\lceil\Lambda$ and the existing events renamed with the relabelling operation $p[\Xi]$. So in this way, we can define specialized parallel compositions of the kind that appear in CCS and CSP, for example. To explain formally the behavior of the constructs in the language we describe them as constructions on labeled event structures, so a closed process term in this language is to denote a **stable event structure** but where the events are labeled.

3.1 Nil

The term *nil* represents the *nil* process that has stopped and refuses to perform any event; it will denoted by the empty labelled event structure $(\varnothing, \varnothing, \varnothing, \varnothing, \varnothing)$ no events, no labels.

3.2 Prefix

Definition 3.2.1. Let $(\mathcal{E}, L, l)$ be a labelled event structure. Let α be a label. Define $\alpha(\mathcal{E}, L, l)$ to be a labelled event structure $(\alpha\mathcal{E}, L', l')$ with labels:

$$L' = \{\alpha\} \cup L$$

and

$$l'(e') = \begin{cases} \alpha & \text{if } e' = (0, \alpha) \\ l(e) & \text{if } e' = (1, e) \end{cases}$$

for all $e' \in \mathcal{E}'$.

The configurations of αE, a prefixed labeled event structure, have the simple and expected characterization. (By $\mathcal{F}(E)$ of a labeled event structure E we shall understand the set of configurations of the underlying event structure)

Proposition 1. Let E be a labelled event structure. Let α be a label.

$$x \in \mathcal{F}(\alpha E) \iff x = \varnothing \text{ or } [(0,\alpha) \in x \ \&\ \{e | (1,e) \in x\} \in \mathcal{F}(E)]$$

3.3 Sum

A sum $p_0 + p_1$ behaves like p_0 or p_1; which branch of a sum is followed will often be determined by the context and what kinds of events the process is restricted to.

Definition 3.3.1. Let $E_0 = (\mathcal{E}_0, \#_0, \vdash_0, L_0, l_0)$ and $E_1 = (\mathcal{E}_1, \#_1, \vdash_1, L_1, l_1)$ be labelled event structures. Their sum $E_0 + E_1$, is defined to be the structure $(\mathcal{E}, \#, \vdash, l)$ with events $\mathcal{E} = \{(0,e) | e \in \mathcal{E}_0\} \cup \{(1,e) | e \in \mathcal{E}_1\}$, the disjoint union of sets $\mathcal{E}_0$ and $\mathcal{E}_1$, with injections $\iota_k : \mathcal{E}_k \to \mathcal{E}$, given by $\iota_k(e) = (k,e)$, for $k = 0, 1$, conflict relation

$$\begin{aligned} e \# e' \iff & \exists e_0, e_0'. e = \iota_0(e_0) \ \&\ e' = \iota_0(e_0') \ \&\ e_0 \#_0 e_0' \\ & \text{or } \exists e_1, e_1'. e = \iota_1(e_1) \ \&\ e' = \iota_1(e_1') \ \&\ e_1 \#_1 e_1' \\ & \text{or } \exists e_0, e_1. (e = \iota_1(e_0) \ \&\ e' = \iota_1(e_1)) \text{ or } (e' = \iota_1(e_0) \ \&\ e = \iota_1(e_1)) \end{aligned}$$

and enabling relation

$$\begin{aligned} X \vdash e \iff & X \in Con \ \&\ e \in \mathcal{E} \ \& \\ & (\exists X_0 \in Con_0, e_0 \in \mathcal{E}_0. X = \iota_0 X_0 \ \&\ e = \iota_0(e_0) \ \&\ X_0 \vdash_0 e_0) \text{ or} \\ & (\exists X_1 \in Con_1, e_1 \in \mathcal{E}_1. X = \iota_1 X_1 \ \&\ e = \iota_1(e_1) \ \&\ X_1 \vdash_1 e_1) \end{aligned}$$

We define the set of labels as $L_0 \cup L_1$ and the labelling function as:

$$l(e) = \begin{cases} l_0(e_0) & \text{if } e = \iota_0(e_0) \\ l_1(e_1) & \text{if } e = \iota_1(e_1) \end{cases}$$

The configurations of a sum are obtained from copies of the configurations of the components identified at their empty configurations.

Proposition 2. Let E_0 and E_1 be labelled event structures.

$$x \in \mathcal{F}(E_0 + E_1) \iff (\exists x_0 \in \mathcal{F}(E_0). x = \iota_0 x_0) \text{ or } (\exists x_1 \in \mathcal{F}(E_1). x = \iota_1 x_1)$$

3.4 Product

A product process $p_0 \times p_1$ behaves like p_0 and p_1 set in parallel. Their events of synchronization are those pairs of events (e_0, e_1), one from each process; if e_0 is labelled α_0 and e_1 is labelled α_1 the synchronization event is then labelled (α_0, α_1). Events need not synchronize however; an event in one component may not synchronize with any event in the other. We shall use events of the form $(e_0, *)$ to stand for the occurrence of an event e_0 from one component

unsynchronized with any event of the other. Such an event will be labeled by $(\alpha_0, *)$ where α_0 is the original label of e_0 and * is a sort of undefined.

In fact we shall often want to take the first or second coordinates of such paris and, of course, this could give the value * which we think of as undefined, so that, in effect, we are working with partial functions with * understood to be undefined. We can keep expressions tidier by adopting some conventions about how to treat this undefined value when it appears in expressions and assertions.

$$\Theta(e) \in X \Rightarrow \Theta(e)\text{is defined, and}$$
$$\Theta(e) = \Theta(e') \Rightarrow \Theta(e)\text{is defined } \ \& \ \Theta(e') \text{ is defined.}$$

We adopt a similar strict interpretation for function application. So if f is a function applied to some value, denoted by a, the $f(a)$ is undefined (gives *) if a is undefined. As usual we represent the image of a set under a partial function by

$$\Theta X = \{\Theta(e) | e \in X \ \& \ \Theta(e) \text{ is defined}\}$$

Definition 3.4.1. Let $E_0 = (\mathcal{E}_0, \#_0, \vdash_0, L_0, l_0)$ and $E_1 = (\mathcal{E}_1, \#_1, \vdash_1, L_1, l_1)$ be labelled event structures. Define their product $E_0 \times E_1$ to be the structure $E = (\mathcal{E}, \#, \vdash, L, l)$ consisting of events $\mathcal{E}$ of the form

$$\mathcal{E}_0 \times_* \mathcal{E}_1 = \{(e_0, *) | e_0 \in \mathcal{E}_0\} \cup \{(*, e_1) | e_1 \in \mathcal{E}_1\} \cup \{(e_0, e_1) | e_0 \in \mathcal{E}_0 \ \& \ e_1 \in \mathcal{E}_1\}$$

with projections $\pi_i : \mathcal{E} \to_* \mathcal{E}_i$, given by $\pi_i(e_0, e_1) = e_i$, for $i = 0, 1$, reflexive conflict relation $\between$ given by

$$e \between e' \iff \pi_0(e) \between_0 \pi_0(e') \text{ or } \pi_1(e) \between_1 \pi_1(e')$$

for all e, e' we use Con for the conflict-free finite sets, enabling relation $\vdash$ given by

$$X \vdash e \iff X \in Con \ \& \ e \in \mathcal{E} \ \&$$
$$(\pi_0(e) \text{ is defined } \Rightarrow \pi_0 X \vdash_0 \pi_0(e)) \ \& \ (\pi_1(e) \text{ is defined } \Rightarrow \pi_1 X \vdash_1 \pi_1(e))$$

Its set of labels is

$$L_0 \times_* L_1 = \{(\alpha_0, *) | \alpha_0 \in L_0\} \cup \{(*, \alpha_1) | \alpha_1 \in L_1\} \cup \{(\alpha_0, \alpha_1) | \alpha_0 \in L_0 \ \& \ \alpha_1 \in L_1\}$$

with projections: $\lambda_i : \mathcal{E} \to_* \mathcal{E}_i$ given by $\lambda_i(\alpha_0, \alpha_1) = \alpha_i$, for $i = 0, 1$. Its labelling function is defined to act on an event e so

$$l(e) = (l_0 \pi_0(e), l_1 \pi_1(e))$$

We characterize the configurations of the product of two event structures in terms of their configurations.

Proposition 3. Let $E_0 \times E_1$ be the product of labelled event structures with projections π_0, π_1. Let $x \subseteq \mathcal{E}_0 \times \mathcal{E}_1$, the events of the product. Then $x \in \mathcal{F}(\mathcal{E}_0 \times \mathcal{E}_1)$ iff

$\pi_0(x) \in \mathcal{F}(E_0)$ & $\pi_1 x \in \mathcal{F}(E_1)$
$\forall e, e' \in x. \pi_0(e) = \pi_0(e')$ or $\pi_1(e) = \pi_1(e') \Rightarrow e = e'$
$\forall e \in x \exists y \subseteq x. \pi_0 y \in \mathcal{F}(E_0)$ & $\pi_1 y \in \mathcal{F}(E_1)$ & $e \in y$ & $|y| <$ infinite
$\forall e, e' in x. e \neq e' \Rightarrow \exists y \subseteq x. \pi_0 y \in \mathcal{F}(E_0)$ & $\pi_1 y \in \mathcal{F}(E_1)$ & $(e \in y \iff e' \notin y)$

The Proposition above expresses the intuition that an allowable behavior of the product of two processes is precisely that which projects to allowable behaviors in the component processes. The complicated-looking conditions (c) and (d) are there just to ensure that the family of sets is finitary and coincidence-free.

3.5 Restriction

The restriction $t\lceil\Lambda$ behaves like the process p but with its events restricted to those with labels that lie in the set Λ.

Definition 3.5.1. Let $E = (\mathcal{E}, \#, \vdash, L, l)$ be a labelled event structure. Let Λ be a subset of labels. Define the restriction $E\lceil\Lambda$ to be $(\mathcal{E}', \#', \vdash', L \cap \Lambda, l')$ where $(\mathcal{E}', \#', \vdash')$ is the restriction of $(\mathcal{E}, \#, \vdash)$ to events $\{e \in \mathcal{E} | l(e) \in \Lambda\}$ and the labelling function l' is the restriction of the original labelling function to the domain $L \cap \Lambda$.

Proposition 4. Let $E = (\mathcal{E}, \#, \vdash, L, l)$ be a labelled event structure. Let $\Lambda \subseteq L$.

$$x \in \mathcal{F}(E\lceil\Lambda) \iff x \in \mathcal{F}(E) \;\&\; e \in x.l(e) \in \Lambda$$

3.6 Relabelling

A relabelled process $p[\Xi]$ behaves like p but with the events relabelled according to Ξ.

Definition 3.6.1. Let $E = (\mathcal{E}, \#, \vdash, L, l)$ be a labelled event structure. Let Λ, L' be sets of labels and $\Xi : \Lambda \to L'$. Define the relabelling $E[\Xi]$ to be $(\mathcal{E}, \#, \vdash, L', l')$ where

$$l'(e) = \begin{cases} \Xi l(e) & \text{if } l(e) \in \Lambda \\ l(e) & \text{otherwise} \end{cases}$$

3.7 Denotational Semantics

Definition 3.7.1. Define an environment for process variables to be a function ρ from process variables X to labeled event structures. For a term t and environment ρ, define the denotation of t with respect to ρ written $[\![t]\!]\rho$ by the

following structural induction syntactic operators appear on the left and their semantics counterparts on the right.

$$\begin{aligned} [\![nil]\!]\rho &= (\varnothing, \varnothing) & [\![t\lceil\Lambda]\!]\rho &= [\![t]\!]\rho\lceil\Lambda \\ [\![x]\!]\rho &= \rho(x) & [\![t[\Xi]]\!]\rho &= [\![t]\!]\rho[\Xi] \\ [\![\alpha t]\!]\rho &= \alpha([\![t]\!]\rho) & [\![t_1 \times t_2]\!]\rho &= [\![t_1]\!]\rho \times [\![t_2]\!]\rho \\ [\![t_1 + t_2]\!]\rho &= [\![t_1]\!]\rho + [\![t_2]\!]\rho & [\![recx.t]\!]\rho &= fix\Gamma \end{aligned}$$

where Γ is an operation on labelled event structures given by $\Gamma(E) = [\![t]\!]\rho[E/x]$ and fix is the least-fixed-point operator.

4 Causality in Event Structure

Let $\mathrm{E} = (E, \#, \vdash)$ be an event structure with $E = \{e_1, e_2, ..., e_n\}$ and let $\{s_1, s_2, ..., s_p\}$ be the power set of E. Given an event structure E, we define $\mathfrak{M}(\mathrm{E}) = (\mathcal{S}, \mathscr{F}, \mathcal{E})$ to be its causal model where $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$. We define $\mathcal{U}$ to be empty and $\mathcal{V}$ consisting of boolean variables as follows:

$$\begin{aligned} \mathcal{V} =& \{C_{e_i,e_j} | 1 \le i < j \le n . e_i \in E \ \&\ e_j \in E\} \\ & \cup \{EN_{s,e} | s \in \mathcal{P}(E), e \in E . e \notin s\} \end{aligned}$$

For $x, y \in \mathcal{P}(E)$ we say x is covered by y written $x \prec y$ iff:

$$x \subseteq y \ \&\ x \neq y \ \&\ (\forall z . x \subseteq z \subseteq y \Rightarrow x = z \text{ or } y = z)$$

For each variable $x \in \mathcal{V}$ we define $\vec{V}_x$ as a vector of all variables in $\mathcal{V}$ excluding x. We define the functions in $\mathscr{F}$ as follows:

$$Con(s) = \left(\bigwedge_{1 \le j < j' \le n \wedge e_j, e_{j'} \in s} \neg C_{e_j, e_{j'}} \right)$$

$$F_{C_{e,e'}}(\vec{V}_{C_{e,e'}}) = \begin{cases} true & \text{if } e \# e' \ \&\ e' \# e \\ false & \text{otherwise} \end{cases}$$

$$F_{EN_{s,e}}(\vec{V}_{EN_{s,e}}) = \begin{cases} \neg \left(\bigvee_{s' \prec s} \neg EN_{s',e} \vee \neg Con(s) \right) & \text{if } s \vdash e \\ \left(\bigvee_{s' \prec s} EN_{s',e} \right) \wedge Con(s) & \text{otherwise} \end{cases}$$

Note that since we have defined $\mathcal{U}$ to be empty and the model is recursive, the values of all variables in $\mathcal{V}$ can be uniquely determined in the model.

Let $\mathcal{M}$ be a causal model and $\mathcal{V}$ be its set of endogenous variables. Let $E = \{e | \exists e' . C_{e,e'} \in \mathcal{V} \vee C_{e',e} \in \mathcal{V}\}$. We define $\mathbb{E}$ be the set of all triples of the form $(E, \#, \vdash)$ where $\# \subseteq E \times E$ and $\vdash \subseteq \mathcal{P}(E) \times E$. Let $\mathcal{M} = \mathfrak{M}(\mathrm{E})$ for some event structure E on the set E. We add two additional endogenous variables ES and IC where $\mathcal{R}(ES) = \mathbb{E}$ and IC is boolean. We define

$$F_{ES}(\vec{V}_{ES}) = (E, \#, \vdash)$$

where we have:

$$\forall e, e' \in E. e \#' e' \wedge e' \#' e \iff C_{e,e'} = \mathrm{T}$$
$$\forall s \in \mathcal{P}(E), e \in E. s \vdash' e \iff EN_{s,e} = \mathrm{T}$$

We define the function of IC to determine whether ES contains any configurations of a set of configurations $\mathcal{C}$:

$$F_{IC}(\vec{V}_{IC}) = \bigvee_{c \in \mathcal{C}} c \in \mathcal{F}(ES)$$

We define $\mathcal{E}$ to be the set of all allowable settings of the endogenous variables such as $\vec{v}'$ for which if $M \vDash \vec{V} = \vec{v}' \wedge ES = \mathrm{E}$ then E be an event structure.

Finally, for a given set of configurations $\mathcal{C}$ we want to find the cause of $\bigvee_{c \in \mathcal{C}} c \in \mathcal{F}(\mathrm{E})$.

Definition 4.0.1. Let $\mathrm{E} = (E, \#, \vdash)$ be an event structure and $\mathcal{M} = \mathfrak{M}(\mathrm{E})$ be its causal model. For a $\sigma \in \mathcal{F}(\mathrm{E})$ given as a counterexample, $\vec{X} = \vec{x}$ is an actual cause of $\sigma \in \mathcal{F}(\mathrm{E})$ in $\mathcal{M}$ if the following three conditions hold:

- **AC1.** $M \vDash \vec{X} = \vec{x} \wedge \bigvee_{c \in \mathcal{C}} c \in \mathcal{F}(\mathrm{E})$.
- **AC2.** There exists a partition $(\vec{Z}, \vec{W})$ of $\mathcal{V}$ with $\vec{X} \subseteq \vec{Z}$ and some setting $(\vec{x}', \vec{w}')$ of the variables in $(\vec{X}, \vec{W})$ such that if $(M, \vec{u}) \vDash \vec{Z} = z^*$ for all $Z \in \vec{Z}$, then both of the following conditions hold:

  (a) $M \vDash [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}']IC = \mathrm{F} \wedge \vec{V} = \vec{v} \wedge \vec{v} \in \mathcal{E}$.

  (b) $M \vDash [\vec{X} \leftarrow \vec{x}, \vec{W}' \leftarrow \vec{w}', \vec{Z}' \leftarrow \vec{z}^*]IC = \mathrm{T} \wedge \vec{V} = \vec{v} \wedge \vec{v} \in \mathcal{E}$ for all subsets $\vec{W}'$ of $\vec{W}$ and all subsets Z' of $\vec{Z}$.
- **AC3.** $\vec{X}$ is minimal; no subset of $\vec{X}$ satisfies conditions $AC1$ and $AC2$.

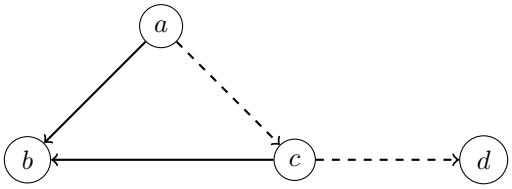
Where $\vec{v}$ is the value of endogenous variables.

4.1 Causality in DyNetKAT

We describe how to use causality for a given DyNetKAT expression.

4.2 Examples

Example 4.2.1. Consider the following network where we want to check a blacklist property, i.e. no packets arrive at d:



We define this network using the following DyNetKAT term:

$$\begin{aligned} S_{xy} &= l = x \cdot l \leftarrow y \\ P &= u!S_{ac} \\ Q &= u!S_{cd} \\ N_{x,y} &= (S_x + S_y)^* \oplus u?x'; N_{x',y} \oplus u?y'; N_{x,y'} \\ SDN &= N_{ab,cb} \parallel P \parallel Q \end{aligned}$$

We may rewrite the terms as follows:

$$\begin{aligned} N_{ab,cb} &= (S_{ab} + S_{cb})^* \oplus u?S_{ac}; N_{ac,cb} \oplus u?S_{cd}; N_{ab,cd} \\ N_{ac,cb} &= (S_{ac} + S_{cb})^* \oplus u?S_{cd}; N_{ac,cd} \\ N_{ab,cd} &= (S_{ab} + S_{cd})^* \oplus u?S_{ac}; N_{ac,cd} \\ N_{ac,cd} &= (S_{ac} + S_{cd})^* \end{aligned}$$

Given a packet σ where $\sigma(l) = a$ we can replace NetKAT policies with actions of the form (σ, σ'). In this network we denote the action (σ, σ') with xy where $\sigma(l) = x$ and $\sigma'(l) = y$. Let $N'_{x,y}$ be the term where we replaced NetKAT policies with actions of the form (σ, σ') and let we use p, p', q, q' to denote the actions $u?S_{ac}, u!S_{ac}, u?S_{cd}, u!S_{cd}$. Thus we have:

$$\begin{aligned} SDN &= N'_{ab,cb} \parallel P \parallel Q \\ P &= p' \\ Q &= q' \\ N'_{ab,cb} &= ab \oplus p; N'_{ac,cb} \oplus q; N'_{ab,cd} \\ N'_{ac,cb} &= ac \oplus ab \oplus q; N'_{ac,cd} \\ N'_{ab,cd} &= ab \oplus p; N'_{ac,cd} \\ N'_{ac,cd} &= ac \oplus ad \end{aligned}$$

Now, we wish to derive the event structure of the SDN. We consider $\oplus, ;, \|$ operators in DyNetKAT as $+, \cdot, \times$ operators in the language respectively.

$$
\begin{aligned}
[\![P]\!] &= (\{p'\}, \varnothing, \{\varnothing \vdash p'\}) \\
[\![Q]\!] &= (\{q'\}, \varnothing, \{\varnothing \vdash q'\}) \\
[\![P \parallel Q]\!] &= (\{p', q'\}, \varnothing, \{\varnothing \vdash p', \varnothing \vdash q'\}) \\
[\![N'_{ac,cd}]\!] &= (\{ac, ad\}, \{ac\#ad\}, \{\varnothing \vdash ac, \varnothing \vdash ad\}) \\
[\![p; N'_{ac,cd}]\!] &= (\{p, ac, ad\}, \{ac\#ad\}, \{\varnothing \vdash p, p \vdash ac, p \vdash ad\}) \\
[\![ab \oplus p; N'_{ac,cd}]\!] &= (\{p, ab, ac, ad\}, \{ab\#p, ab\#ac, ab\#ad, ac\#ad\}, \\
&\qquad \{\varnothing \vdash ab, \varnothing \vdash p, p \vdash ac, p \vdash cd\}) \\
[\![q; N'_{ac,cd}]\!] &= (\{q, ac, ad\}, \{ac\#ad\}, \{\varnothing \vdash q, q \vdash ac, q \vdash ad\} \\
[\![ab \oplus q; N'_{ac,cd}]\!] &= (\{q, ab, ac, ad\}, \{ac\#ad, ab\#q, ab\#ac, ab\#ad, \}, \\
&\qquad \{\varnothing \vdash ab, \varnothing \vdash q, q \vdash ac, q \vdash ad\}) \\
[\![ac \oplus ab \oplus q; N'_{ac,cd}]\!] &= (\{q, ab, ac_1, ac_2, ad\}, \\
&\qquad \{ac_1\#ad, ab\#q, ab\#ac_1, ab\#ad, ac_2\#ab, ac_2\#q, ac_2\#ac_1, ac_2\#ad\}, \\
&\qquad \{\varnothing \vdash ab, \varnothing \vdash q, q \vdash ac_1, q \vdash ad, \varnothing \vdash ac_2\}) \\
[\![q; N'_{ab,cd}]\!] &= (\{p, q, ab, ac, ad\}, \{ab\#p, ab\#ac, ab\#ad, ac\#ad\}, \\
&\qquad \{\varnothing \vdash q, q \vdash ab, q \vdash p, \{p, q\} \vdash ac, \{p, q\} \vdash cd\}) \\
[\![p; N'_{ac,cb}]\!] &= (\{p, q, ab, ac_1, ac_2, ad\}, \\
&\qquad \{ac_1\#ad, ab\#q, ab\#ac_1, ab\#ad, ac_2\#ab, ac_2\#1, ac_2\#ac_1, ac_2\#ad\}, \\
&\qquad \{\varnothing \vdash p, \{p\} \vdash ab, \{p\} \vdash q, \{p, q\} \vdash ac_1, \{p, q\} \vdash ad, \{p\} \vdash ac_2\}) \\
[\![p; N'_{ac,cb} \oplus q; N'_{ab,cd}]\!] &= (\{p_1, p_2, q_1, q_2, ab_1, ab_2, ac_{21}, ac_{22}, ac_1, ad_1, ad_2\}, \\
&\qquad \{ab_1\#p_1, ab_1\#ac_1, ab_1\#ad_1, ac_1\#ad_1 \\
&\qquad ac_{21}\#ad_2, ab_2\#q_2, ab_2\#ac_{21}, ab_2\#ad_2, \\
&\qquad ac_{22}\#ab_2, ac_{22}\#q_2, ac_{22}\#ac_{21}, ac_{22}\#ad_2\} \\
&\qquad \cup \{e_i\#e_{i'} | i \neq i'\} \\
&\qquad \{\varnothing \vdash p_2, \{p_2\} \vdash ab_2, \{p_2\} \vdash q_2, \{p_2, q_2\} \vdash ac_{21}, \{p_2, q_1\} \vdash ad_2, \{p_2\} \vdash ac_{22}, \\
&\qquad \varnothing \vdash q_1, q_1 \vdash ab_1, q_1 \vdash p_1, \{p_1, q_1\} \vdash ac_1, \{p_1, q_1\} \vdash cd_1\}) \\
[\![ab \oplus p; N'_{ac,cb} \oplus q; N'_{ab,cd}]\!] &= (\{p_1, p_2, q_1, q_2, ab_1, ab_2, ab_3, ac_{21}, ac_{22}, ac_1, ad_1, ad_2\}, \\
&\qquad \{ab_1\#p_1, ab_1\#ac_1, ab_1\#ad_1, ac_1\#ad_1 \\
&\qquad ac_{21}\#ad_2, ab_2\#q_2, ab_2\#ac_{21}, ab_2\#ad_2, \\
&\qquad ac_{22}\#ab_2, ac_{22}\#q_2, ac_{22}\#ac_{21}, ac_{22}\#ad_2\} \\
&\qquad \cup \{e_i\#e_{i'} | i \neq i'\} \\
&\qquad \{\varnothing \vdash p_2, \{p_2\} \vdash ab_2, \{p_2\} \vdash q_2, \{p_2, q_2\} \vdash ac_{21}, \{p_2, q_1\} \vdash ad_2, \{p_2\} \vdash ac_{22}, \\
&\qquad \varnothing \vdash ab_3, \varnothing \vdash q_1, q_1 \vdash ab_1, q_1 \vdash p_1, \{p_1, q_1\} \vdash ac_1, \{p_1, q_1\} \vdash cd_1\})
\end{aligned}
$$

Appendices