Information Flow and Homotopy Theory

Gerard Allwein[a], William L. Harrison[b]

[a]*Naval Research Laboratory, Code 5543, Washington, DC 20375, U.S.A.*
[b]*Dept. of Computer Science, University of Missouri, Columbia, Missouri, U.S.A.*

Abstract

Simulation and bisimulation are used in many areas of security in Computing Science, however the methods used do not come with a supporting logic giving the regularities of information flow. We have developed Distributed Logic to represent regularities governing information flow between localities of a distributed system, each locality can be outfitted with its own notion of state and language. The flows, either internal to a locality or between localities, follow a connection structure specified by a graph in the logic and semantically by relations in a category. We use some basic homotopy notions for use in distributed systems by using the notions as a template for properties of (bi)simulations useful for security and other properties of distributed systems. The logic can express formulas whose validity is unwritten by the (bi)simulations. This presents new ways of formalizing and proving properties of distributed systems.

Keywords: Kripke models, homotopy, simulation, bisimulation, FPGAs

1. Introduction

Information flow has been modeled in several logical formalisms, namely Channel Theory [1], Chu Spaces [2], and Institutions [3]. In those formalisms, information flow is handled entirely at the meta-logical level. We have developed a logic, Distributed Logic, in which the regularities that govern information flow can be specified. Distributed Logic [4, 5] is a distributed form of mathematical logic analogous to distributed systems in computing, which are sets of computers interacting over a network.

One principal difference between these other logical formulations and Distributed Logic that the distribution is also represented at the logical level in terms of a graph that is part of the syntax of the logic. The distribution in models is via distributed Kripke relations and hence in this version of Distributed Logic, there are distributed modal operators.

*Corresponding Author
Email addresses: gerard.allwein@nrl.navy.mil (Gerard Allwein), harrisonwl@missouri.edu (William L. Harrison)

[1]This work was supported by U.S. Naval Research Laboratory, Award No.: N00173 -19-1-G004

We will term the regularities that govern information flow *flow control*. An advantage of the explicit, syntactic representation of flow control in Distributed Logic is the potential for mechanization—i.e., incorporation into a proof checking system like Coq (`https://coq.inria.fr`). Its amenability for use in formal verification is part of the motivation behind the development of Distributed Logic. Another motivation is that we are aiming to verify security and safety properties of embedded hardware systems realized on Field Programmable Gate Arrays (FPGAs), that are themselves distributed systems.

FPGAs are composed of a substrate, usually called FPGA fabric. This is a uniform array of elements that can be put together in combinations to construct components of an application. The components all run in parallel whereas within a single component, there is frequently a state machine as well as some local storage. FPGAs are reprogrammable and have become quite sophisticated containing stock components manufactured into them as resources that the applications can use. Another technology is Application Specific Integrated Circuits (ASICs). These are similar to FPGAs yet not reprogrammable. Computer CPUs are larger and contain many of the same elements, and some CPUs are now available with FPGA fabric built-in. These systems come under the heading of Systems-on-a-Chip (SoCs).

SoC systems are distributed systems (in the traditional sense of Computing Science). Each is built from interconnected components, each of which may possess its own internal storage. A logical framework that includes a local logic for each component, yet also includes logical mechanisms for interconnection, is a natural setting for formal verification of such systems. In particular, we (and many others) are interested in formal verification of security properties. It is also the case that simulations from modal logic are at the heart of information security (as we elaborate on below) and it is such security properties we seek to establish for embedded hardware.

2. Information Flow and Computing Science

Information flow has been a central theme in Computing Science. In this paper, we will be very precise about how we represent information and how it flows. We take our notion of information and information flow from Barwise and Seligman [1]. We use the usual notion of state in Computing Science, which is the contents of memory. From modal logic, we use the usual notion of world as an entity that values all propositions true or false. In Barwise and Seligman, they use neutral term *token*. We use the terms *state* and *world* interchangeably; the reader may choose either *state* or *world* depending upon the reader's viewpoint.

A state is not information. A state can only carry information when it turns a particular proposition true or false. By itself, a state is mere data signifying little. Similarly, a proposition can only indicate information about states. By itself, a proposition is untethered and a free floating concept, it is not information. If x is a state and P is a proposition, then the formula $x \models P$

indicates a basic unit of information; the $\models$ (models) relation is interpreted as x is of type P or x makes P true.

To flow, information must be involved in statements that express connections. For example, $(x \models P)$ iff $(Hxy$ implies $y \models Q)$, where $H : \mathsf{h} \rightharpoonup \mathsf{h}$ expresses connection information within the single universe of discourse h. H is the route of information flow. The information flow is backwards along $\mathcal{H}$ from $y \models Q$ to $x \models P$ when this prescription is looked upon as telling us when $x \models P$. The relation $\mathcal{H}$ relates x to y where the converse of $\mathcal{H}$ is not necessarily a transformation such as a mathematical function. This allows for generality to cover different use cases in Computing Science.

Expressed within a single universe of discourse, the flows follow internal connection information; this was captured by H above. This is still a very conventional concept where there is a single background against which all reasoning is internal and somewhat hidden. A distributed view would treat the relation as connection information between universes which we call *localities*. We view a flow of this type as being underwritten by a relation $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$ with localities h and k. Each locality contains a collection of states, but the collections are different for each locality. Now information flow can be internal to a locality, but it can be also between localities. Hence we have model theoretic statements such as $x \overset{\mathsf{h}}{\models} P$ iff $\mathcal{F}xy$ implies $y \overset{\mathsf{k}}{\models} Q$.

A more involved kind of information flow is used in the concept of a simulation relation. It is built upon the preceding framework, but provides more structure. A simulation relates states in one locality to states in another locality. Simulations in the semantic theory of modal logic capture a particular kind of information flow between Kripke frames and bring in the propositions using the $x \models P$ format. Hence, they are statements about information flow. Typically, these statements are in the meta-mathematics involved in the semantics of modal logic.

Distributed Logic is able to treat simulation relations as modal Kripke relations between localities. We term relations between localities as *distributed relations* to distinguish them from the usual Kripke relations on single localities. Just as modal logic provides axioms that determine, via validity, models that have Kripke relations with special properties, we use Distributed Logic to also provide special properties for distributed relations. The particular property we shall be using in this paper is that of simulation.

A simulation relation may be thought of as one half of a bisimulation, where bisimulation is well-known and originating in modal logic [6] and finding application across a wide swath of theoretical Computing Science (e.g., process algebras [7], program verification [8], etc.). A bisimulation relation is a simulation relation such that its converse is also a simulation relation. When $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$ is a simulation relation in a modal system, it relates the states of h and k but it also relates two modal relations, $\mathcal{H} : \mathsf{h} \rightharpoonup \mathsf{h}$ and $\mathcal{K} : \mathsf{k} \rightharpoonup \mathsf{k}$ according the following prescription:

$$\mathcal{F}xy \text{ and } \mathcal{H}xx' \text{ implies } \exists y'(\mathcal{K}yy' \text{ and } \mathcal{F}x'y')$$

Simulations in modal logic also relate propositions between the localities, and generally, the languages of h and k are the same. In our uses, the languages of h and k are different. To express the relationship between the propositions of the languages, we treat the simulation relation as a modal relation and the above prescription validates the following *simulation formula*:

$$\langle f\rangle[k]Q \overset{\mathsf{h}}{\supset} [h]\langle f\rangle Q.$$

Here Q is a formula of k, $[k]$ is the modal necessity operator at k interpreted by $\mathcal{K}$. Similarly, $[h]$ is the modal necessity operator at h interpreted by $\mathcal{H}$. The distributed operator $\langle f\rangle$ can be thought of as mapping formulas at k to formulas at h or making a particular identification between formulas at k and at least some formulas at h. The symbol $\overset{\mathsf{h}}{\supset}$ is classical implication in the locality h. We use the locality designations h and k as meta-linguistic variable ranging over localities. Each locality contains states, a modal logic, and a $\models$ relation.

The flow of information is defined in the validation of this statement and is from k to h. The validation by states and relations says that this statement holds in h just when Q holds in k under conditions of the model satisfying the simulation condition. The mapping $\langle f\rangle$ from propositions in k to propositions in h reveals this.

This notion of simulation underlies many models of information flow security, including, for example, the classic Goguen-Meseguer (GM) model of non-interference [9]. Goguen and Meseguer's notion of information flow is defined as a violation of a simulation, and we illustrate this notion with the following scenario.

Threads are sequences of pairs, each pair containing an instruction and a state. Instructions are either L or H instructions. States are either L states or $L + H$ states; L states contain only L memory, but $L + H$ states contain a mixture of L and H memory. We combine the memory and control for the L machine and call the machine M_L. The total memory of L and H and both L and H instructions is the machine M_{L+H}. We equate each machine with its collection of threads. Threads of M_L contain only L instructions, threads of M_{L+H} contain L and H instructions (each instruction paired with a state).

The GM model compares a projection of the threads from the M_{L+H} machine with the threads of the M_L machine. The projection strips threads from M_{L+H} of their H content. Any pair containing an H instruction is stripped out. Of the remaining pairs, any H memory is stripped from the state portion of the pairs. We call the resulting collection of threads M_{L-H}.

An anomaly occurs when the threads of M_{L-H} are not the same as the threads of M_L. Some residual influence of H instructions has been left in L memory. Information is said to flow when anomalies occur. We analyze this situation in the Applications section. In particular, we establish a bisimulation between M_{L-H} and M_{L+H}. When no anomalies occur, there is a bisimulation between M_L and M_{L-H}. When anomalies occur, we can reprocess some of the data we use to represent the relation between threads in M_{L-H} to establish a relation from M_{L-H} to M_L. This relation can be used to specify what information flows.

Computer security abounds with this simulation-based view of information flow. For another example, consider control-flow integrity (CFI) of Abadi, et al. [10, 11] and, in particular, hardware-based monitoring of CFI [12, 13, 14, 15]. Program control-flow graphs (Figure 1) are a well-known structure in language compilation and CFI uses them to track the normal execution of programs. The essential insight of CFI is that deviation from its control-flow graph by a program in execution indicates a successful malicious attack. Using a program's control-flow graph, a CFI hardware monitor (Figure 2) simulates the program's execution by tracking its instruction *address* fetch requests. The failure of the simulation then indicates the security breach: if these addresses do not conform to those predicted by the control-flow graph, then the *alarm* bit is set. Hardware-based CFI monitoring is a particularly apt motivating example for Distributed Logic because this scenario is distributed—i.e., the processor executing the program and the monitor itself do not share storage but, rather, communicate via signals.

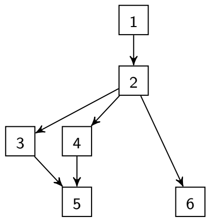

Figure 1: Call Graph with No Monitor

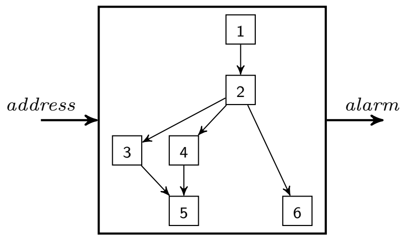

Figure 2: Call Graph with Monitor

The box around the graph in the second diagram represents the security monitor. This situation is said to be a simulation, actually it is a bisimulation. If the machine makes a move, then we expect the monitor to also make a move. However, the monitor should not be able to make a move if the move is not forced by the machine. What is this bisimulation precisely? If it fails, how does information flow in terms of simulations? We also answer this question precisely.

To provide more structure for bisimulations expressible in Distributed Logic, we extrapolated from homotopy theory and, in particular, its use of groupoids. This allowed us to express a machine not entering danger zones by representing those as propositions over Stone spaces. Stone spaces and Kripke relations are the modal logic way to represent machines from Computing Science. Stone spaces, due to their topology, do not have holes. However, the connection structure is expressed in Kripke relations that separate the topology from the connection structure. This allowed us to think of holes as being propositions to avoid, i.e., danger zones.

Another extrapolation from homotopy is homotopy type theory. We view our extrapolation in a similar vein. Ours is more extensional than the intensional homotopy type theory. However, we have plans to make ours more intensional when we incorporate it into a language we have called ReWire [16, 15]. ReWire is a language for applications in FPGAs and Systems-on-a-Chip (SoCs).

3. Conventions

Distributed Logic requires as many localities as are needed for a particular application. The rather untyped structure of classical logic needs to be augmented with indices for localities. Connectives and relations must be properly typed with the localities. The conventions for this paper are collected here for easy reference.

entity	description
$\mathsf{h}, \mathsf{k}, \mathsf{l}$	nodes, called *localities*, in a graph $\mathfrak{G}$
f, g	distributed arrows in graph $\mathfrak{G}$
h, k, l, r, s	endo-arrows in graph $\mathfrak{G}$
$\langle t \rangle, [t]$	forward modalities associated with $t \in \{f, g, h, k, l, r, s\}$
$\langle t \rangle, [\cdot t \cdot]$	converse modalities associated with $t \in \{f, g, h, k, l, r, s\}$
$(H, \mathcal{H}, BA(\mathsf{h}))$	general, descriptive frame for the logic, Log(h), at h
H, K, L	sets of worlds at $\mathsf{h}, \mathsf{k}, \mathsf{l}$ respectively
$\mathcal{H}, \mathcal{K}, \mathcal{L}$	modal relations interpreting the local arrows at $\mathsf{h}, \mathsf{k}, \mathsf{l}$ respectively
$\mathcal{F}, \mathcal{G}$	relations interpreting distributed arrows f, g
x, x', u, u'	states at h
y, y', v, v'	states at k
w, z	states as needed
P, R	metalinguistic variables at node h
Q, S	metalinguistic variables at node k

Table 1: Notation

For the most part, only two localities h and k are necessary as meta-linguistic variables for nodes in a graph. The locution "x at h" is to be understood as the state x in the Stone space at h.

We have neglected to mention the algebraic models in the sequel due to length considerations. Those models are heterogeneous algebras [17]. We used them in [18] and [4]. A local algebra is a Boolean lattice with normal operators. The distributed operators are heterogeneous operators. This formulation provides soundness and completeness results for Distributed Logic. We will only address soundness with respect to Kripke models in this paper due to length considerations and leave completeness for a future paper.

4. Simulations

Simulations are a model theoretic notion for systems and appear quite widely in the literature. We concentrate on simulations for modal systems. Typically simulations in modal logic appear as preserving formulas of a certain form from one modal system to another. They also assume the same language appears on both sides of the simulation. We do not make this assumption here although it can be enforced by the inclusion of certain axioms involving propositional letters (see Axioms below).

Simulation Logic [18] treats simulation relations as distributed Kripke relations. As such, they satisfy a particular kind of axiom, the simulation axiom. Bisimulations satisfy two axioms, but they not merely the reversals of the main implication connective. They do however specify that a bisimulation condition occurs in any interpretation validating the axioms. From this basic machinery, we develop rule forms for simulation and show they bear a resemblance to residuation rules. The rules use some derived formulas which we originally developed for talking about features of applications in FPGAs.

4.1. Normal Distributed Modal Logic

The connective $\langle f \rangle$ is a distributed possibility, and $[h]$, $[k]$ are local neeessities. We denote local arrows in the graph using the unadorned math italic h, k, etc.. We denote distributed arrows using the unadorned f and g. Hence an arc $h : \mathsf{h} \rightharpoonup \mathsf{h}$ represents an endo-arrow. An arc $f : \mathsf{h} \rightharpoonup \mathsf{k}$ represents a distributed arrow from h to k. We will generally assume an endo-arrow for any node h, k, etc. Out of practicality, more could be added depending upon the application. When dealing with homotopy notions, we will have need for more endo-arrows. We will always type f as $f : \mathsf{h} \rightharpoonup \mathsf{k}$, however the type of g will vary as needed.

There are no restrictions on a Distributed Logic's graph except that it be finite. The arcs of the graph specify relations among the Kripke frames rather than modal connectives. The reason is that a single distributed relation can support several different modal connectives (see [19] but generalized to be distributed connectives as in [4]).

From [18], the axiom set for the base Simulation Logic is

Graph

S1. A directed graph $\mathfrak{G}$ of nodes and arrows

S2. An endo-arrow i_{h} for each node in $\mathfrak{G}$

Axiom Schemes A: For each node in $\mathsf{h} \in \mathfrak{G}$, and endo-arrow i_{h} at h,

A1. all truth functional theorems of a propositional logic at h

A2. Normal modal axioms for a local logic at h

A3. $P \overset{\mathsf{h}}{\equiv} [i_{\mathsf{h}}]P$

Axiom Schemes B: For each arrow $f : \mathsf{h} \rightharpoonup \mathsf{k}$ and $g : \mathsf{k} \rightharpoonup \mathsf{l}$ in $\mathfrak{G}$,

B1. $[f][g]V \overset{\mathsf{h}}{\equiv} [f \circ g]V$

B2. $[\cdot g \cdot][\cdot f \cdot]P \overset{\mathsf{h}}{\equiv} [\cdot f \circ g \cdot]P$

The following Axioms are optional and we do not assume them. We add them here to give a flavor of what is possible.

Axiom Schemes D: To force arrows to be functions, use the following **D** axioms, just as they would in any normal modal logic systems. We do not assume these axioms hold in the sequel:

D1. $[f]Q \overset{\mathsf{h}}{\supset} \langle f \rangle Q$

D2. $\langle f \rangle Q \overset{\mathsf{h}}{\supset} [f]Q$

Axiom Schemes E: The axiom E1 is only necessary if you wish the classical proposition logic at Log(h) to be included in the logic at Log(k). It is not strictly necessary although it does pick up the clause in the definition of simulation [6] requiring this of a simulation. We do not assume these axioms hold in the sequel:

For all propositional letters p,

E1. $p \overset{\mathsf{h}}{\supset} [f]p$ E2. $p \overset{\mathsf{k}}{\supset} [\cdot f \cdot]p$

We assume each locality h has at least a modal relation $\mathcal{H}$ to interpret formulas at Log(h).

Definitions and Rules (Normal Systems)

Definition of Possibility: $\langle m \rangle P \overset{def}{=} \neg[m]\neg P, \quad m \in \{h, k, f\}$

Local Rules: For a local logic at h,

$$\frac{\overset{\mathsf{h}}{\vdash} P \qquad \overset{\mathsf{h}}{\vdash} P \supset R}{\overset{\mathsf{h}}{\vdash} R} \qquad\qquad \frac{\overset{\mathsf{h}}{\vdash} (P_1 \wedge \cdots \wedge P_n) \supset P}{\overset{\mathsf{h}}{\vdash} ([h]P_1 \wedge \cdots \wedge [k]P_n) \supset [k]P}$$

The second rule is a bit redundant with respect to the normal modal logic axioms A2 since it will force any local logic to be normal.

Distributed Rules For each $f : \mathsf{h} \rightharpoonup \mathsf{k}$ arrow in $\mathfrak{G}$,

$$\frac{\overset{\mathsf{k}}{\vdash} (Q_1 \wedge \cdots \wedge Q_n) \supset Q}{\overset{\mathsf{h}}{\vdash} ([f]Q_1 \wedge \cdots \wedge [f]Q_n) \supset [f]Q} \qquad\qquad \frac{\overset{\mathsf{h}}{\vdash} (P_1 \wedge \cdots \wedge P_n) \supset P}{\overset{\mathsf{k}}{\vdash} ([\cdot f \cdot]P_1 \wedge \cdots \wedge [\cdot f \cdot]P_n) \supset [\cdot f \cdot]P}$$

In the rest of the paper, we will be adding extra axioms to address simulations in various forms. We will include their valuation conditions where we introduce them. Some involve the Kleene * and converse $\breve{}$. We could add the propositional dynamic logic axioms for * since * must be a local logic connective due to an application of * as reflexive-transitive closure of a distributed relation would not make sense. Since we are only concerned in this paper with validation over what we consider our standard models in which * is reflexive-transitive closure and $\breve{}$ is converse, we will ignore those axioms. We do however have the ability to use converse in modalities, say $[f \circ \breve{g}]$. We can always define converse in $[\breve{g}]$ as $[\cdot g \cdot]$ and due to our axiom set, $[f \circ \breve{g}]$ becomes $[f][\cdot g \cdot]$. In a follow on paper, we will address these axioms and more.

In any particular application, we may want to express properties of particular propositional constants, say, open face fonts in $\mathbb{P}$ in h and $\mathbb{Q}$ in k. We will use this notation in talking about holes in Stone spaces, i.e., we use a propositional constant to represent a hole. We also use them for expressing axioms about the domain of an interpreting relation.

4.2. Frames and Models

The interpretation of formulas follows the usual modal logic interpretations with care for the distributed structure. Let $f : \mathsf{h} \rightharpoonup \mathsf{k}$, the evaluation conditions for $[f]$ and $\langle f\rangle$ are

$$x \overset{\mathsf{h}}{\models} [f]Q \text{ iff } \forall y \in K(\mathcal{F}xy \text{ implies } y \overset{\mathsf{k}}{\models} Q), \text{and}$$
$$x \overset{\mathsf{h}}{\models} \langle f\rangle Q \text{ iff } \exists y \in K(\mathcal{F}xy \text{ and } y \overset{\mathsf{k}}{\models} Q).$$

The local connectives are interpreted similarly for local relations called *endo-relations*, say $\mathcal{H}$ and $\mathcal{K}$, for $\mathcal{F}$. As is typical, the set algebra provides a $\models$ relation (respecting localities) using the set theoretical membership relation, $\in$. Hence this formula is evaluated thusly

$$x \in_{\mathsf{h}} [f]Q \text{ iff } \forall y \in K(\mathcal{F}xy \text{ implies } y \in_{\mathsf{k}} Q), \text{and}$$
$$x \in_{\mathsf{h}} \langle f\rangle Q \text{ iff } \exists y \in K(\mathcal{F}xy \text{ and } y \in_{\mathsf{k}} Q).$$

where Q is now a set of points in $BA(\mathsf{k})$. The proposition Q, when viewed as a set of points, is a termed a UCLA proposition.

A *distributed frame* has a Stone space, $\text{Stone}(\mathsf{h}) = (H, \mathcal{H}, BA(\mathsf{h}))$, at each node where H is collection of points, $\mathcal{H}$ is a endo-relation, and $BA(\mathsf{h})$ is a Boolean set algebra of points. We sometimes display the type of relations through an abuse of notation as $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$ for $\mathcal{F} \subseteq H \times K$ where the direction is by fiat from H to K. This abuse is justified since there will be a relation in the distributed frame for every arc in the graph of the logic. In general, the lower case letters in modal connectives and in arcs of the graph are depicted using their script upper case versions in the frames.

Definition 1 (Kupke, Kurz, and Venema [20]). *A* general frame *is a structure* $(H, \mathcal{H}, BA(\mathsf{h}))$ *such that* $(H, \mathcal{H})$ *is a Kripke frame and* $BA(\mathsf{h})$ *is a collection of so-called* admissible *subsets of* H *that is closed under the Boolean operations and under the operation* $[h] : BA(\mathsf{h}) \to BA(\mathsf{h})$ *given by:*

$$[h]P \overset{def}{=} \{x \in X \mid \mathcal{H}xy \text{ implies } y \in P\}$$

with $\langle h\rangle P = \neg[h]\neg P$ *where* $\neg$ *is set complement. A general frame* $\mathcal{H}$ *is called* differentiated *if for all distinct* $x, y \in H$ *there is a 'witness'* $a \in BA(\mathsf{h})$ *such that* $y \in a$ *while* $x \notin a$*;* tight *if whenever* y *is not an* $\mathcal{H}$*-successor of* x*, then there is a 'witness'* $a \in BA(\mathsf{h})$ *such that* $y \in a$ *while* $x \notin \langle h\rangle a$*; and* compact *if* $\bigcap A \neq \emptyset$ *for every subset* A *of* $BA(\mathsf{h})$ *which has the finite intersection property. A general frame is* descriptive *if it is differentiated, tight, and compact.*

As it is noted in [20], tightness can be re-expressed as a relation being point closed, i.e., $\mathcal{H}x$ (equal to $\mathcal{H}\{x\}$), the forward image of x under $\mathcal{H}$, is a closed set in the topology generated by the algebra $BA(\mathsf{h})$ of clopen sets.

We use a category of DG frames for a Distributed Logic.

Definition 2. *A* distributed frame category *has a descriptive, general frame, called a* local frame*, for each node in the graph, and a point closed relation for each arrow. That is, for* $f : \mathsf{h} \rightharpoonup \mathsf{k}$ *in the graph,* f*'s interpretation* $\mathcal{F} : \mathcal{H} \rightharpoonup \mathcal{K}$ *must be point closed.*

The corresponding Kripke frame conditions for the logical axioms are

Frame Conditions S:

FS1. A category of *local modal frames* and simulations

FS2. An endo-arrow $\mathcal{I}_{\mathsf{h}}$ for the arc $i_{\mathsf{h}} \in \mathfrak{G}$

Frame Conditions A: For each node in $\mathfrak{G}$,

FA1. A set of classical worlds

FA2. Modal frame conditions for a logic at this node

FA3. $\mathcal{I}_{\mathsf{h}}$ is an identity relation

Frame Conditions B:

FB1. Frame category contains all compositions

FB2. Frame category contains all converse compositions

with the convention that upper case script relation letters interpret modalities, the latter using lower case Roman letters. Each distributed frame category interpreting a distributed logic will have the conditions matching the axioms. The frame conditions **S**, **A**, and **B** are always assumed, additional axioms require additional frame conditions.

4.3. Modal Composition

Suppose we have the following situation

$$\begin{tikzcd}
{[h][k]V} \arrow[d, dash, "\models^{\mathsf{h}}"'] & {[k]V} \arrow[l, "{[f]}"'] \arrow[d, dash, "\models^{\mathsf{k}}"] & V \arrow[l, "{[g]}"'] \arrow[d, dash, "\models^{\mathsf{l}}"] \\
x \arrow[r, "\mathcal{F}"'] & y \arrow[r, "\mathcal{G}"'] & z
\end{tikzcd}$$

Figure 3: Composition of Formulas and relations

One can see from the diagram that the modal operators are applied in the opposite order as the relations are composed. Relational composition of $\mathcal{F}$ and $\mathcal{G}$ is denoted $\mathcal{F} \cdot \mathcal{G}$. As morphisms, this flips to the categorical composition $\mathcal{G} \circ \mathcal{F}$. Hence $[f \circ g]$ will be interpreted by the relation $\mathcal{G} \circ \mathcal{F}$ and the direction is suitably flipped between the logic and the distributed Kripke frame.

Lemma 3. *The following axioms are valid: for each $f : \mathsf{h} \rightharpoonup k$ and $g : \mathsf{k} \rightharpoonup \mathsf{l}$ in $\mathfrak{G}$,*

$$[f][g]U \overset{\mathsf{h}}{\equiv} [f \circ g]U \qquad [\cdot g \cdot][\cdot f \cdot]P \overset{\mathsf{l}}{\equiv} [\cdot f \circ g \cdot]P.$$

Lemma 4. *The following formulas are valid for $f : \mathsf{h} \rightharpoonup \mathsf{k}$, $g : \mathsf{k} \rightharpoonup \mathsf{l}$, and $p : \mathsf{l} \rightharpoonup \mathsf{t}$,*

$$[(f \circ g) \circ p]V \overset{\mathsf{h}}{\equiv} [f \circ (g \circ p)]V \qquad [\cdot (p \circ g) \circ f \cdot]P \overset{\mathsf{t}}{\equiv} [\cdot p \circ (g \circ f) \cdot]P.$$

4.4. Simulations in Distributed Logic

Formulas of the following form are from Simulation Logic [18]:

$$\langle f\rangle[k]Q \overset{\mathsf{h}}{\supset} [h]\langle f\rangle Q$$

where the distributed connective $\langle f\rangle$ has type $\langle f\rangle : \mathrm{Log}(\mathsf{h}) \to \mathrm{Log}(\mathsf{k})$ for $\mathrm{Log}(\mathsf{h})$ and $\mathrm{Log}(\mathsf{k})$, which are the logics at node h and k respectively. The connective $\overset{\mathsf{h}}{\supset}$ indicates that $\supset$ is classical implication at node h. The nodes h and k are part of the logic's graph. The local modal connectives are $[h] : \mathrm{Log}(\mathsf{h}) \to \mathrm{Log}(\mathsf{h})$ and $[k] : \mathrm{Log}(\mathsf{k}) \to \mathrm{Log}(\mathsf{k})$.

This formula is validated by the frame condition ([18], [6])

$$\mathcal{F}xy \text{ and } \mathcal{H}xx' \text{ implies } \exists y'(\mathcal{K}yy' \text{ and } \mathcal{F}x'y')$$

where $[h]$ and $[k]$ are interpreted by $\mathcal{H}$ and $\mathcal{K}$ respectively, and $\langle f\rangle$ is interpreted by $\mathcal{F}$. A diagram of the relevant localities is

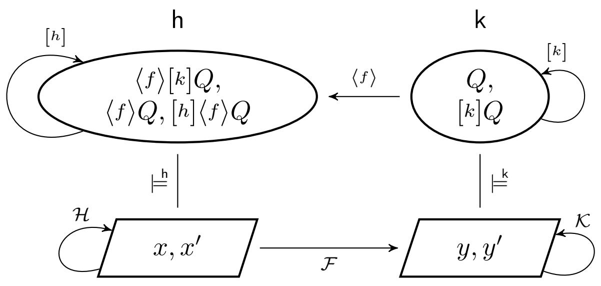

Figure 4: Intuitive View of Subformulas in a Simulation Axiom and its Interpretation

The simulation condition is (in a hack of the graphical notation from Freyd and Scedrov [21]):

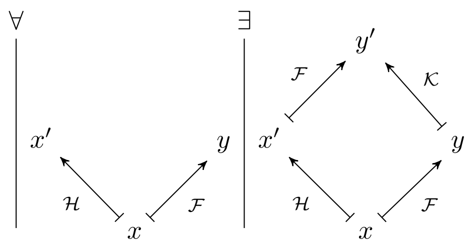

Figure 5: Diagrammatic Form of Simulation Condition

This usage of small italicized letters in the logic interpreted by their large italicized counterparts in the frames is carried on throughout the paper. To simplify the exposition, we generally assume a single collection of local modalities: $[h]$, $\langle h\rangle$, $[\cdot h\cdot]$, and $\langle\cdot h\cdot\rangle$ for two forward and two converse modalities (see [19]) at a node h. We use an interpreting relation $\mathcal{H}$ for those modalities. Local notions generally use h and k (in their various fonts), and distributed notions are similar using f and g.

A bisimulation between h and k is specified with the following formulas

$$\langle f\rangle[k]Q \overset{\mathsf{h}}{\supset} [h]\langle f\rangle Q \qquad \langle f\rangle[h]P \overset{\mathsf{k}}{\supset} [k]\langle f\rangle P.$$

Notice that the second formula is not a mere switching of the direction of the $\supset$ connective. The first uses the $\supset$ connective of h and the second of k. Even if only a single local logic was used, say, at h, the two formulas would not satisfy the reversal of direction of $\overset{\mathsf{h}}{\supset}$. These formula have the following validation conditions (respectively)

$$\mathcal{F}xy \text{ and } \mathcal{H}xx' \text{ implies } \exists y'(\mathcal{K}yy' \text{ and } \mathcal{F}x'y') \text{ and}$$
$$\mathcal{F}xy \text{ and } \mathcal{K}yy' \text{ implies } \exists x'(\mathcal{H}xx' \text{ and } \mathcal{F}x'y').$$

The proof that the first formula is validated by the first condition is paradigmatic of many of the simulations in the sequel and hence we repeat it here (in somewhat pedantic form) using the Fitch-style proofs of [22] and elide such validation proofs in the sequel:

1	$x \in_{\mathsf{h}} \langle f\rangle[k]Q$	assume
2	$\mathcal{F}xy$ and $y \in_{\mathsf{k}} [k]Q$	def. $\langle f\rangle$ for some y, line 1
3	$\quad \mathcal{H}xx'$	assume
4	$\quad \mathcal{F}xy$ and $\mathcal{H}xx'$	$\wedge$-Elim, $\wedge$-Intro, lines 2, 3
5	$\quad \mathcal{K}yy'$ and $\mathcal{F}x'y'$	Simulation Condition for some y', line 4
6	$\quad y' \in_{\mathsf{k}} Q$	def. $[k]$, lines 2, 5
7	$\quad x' \in_{\mathsf{h}} \langle f\rangle Q$	def. $\langle f\rangle$, lines 5, 6
8	$x \in_{\mathsf{h}} [h]\langle f\rangle Q$	def. $[h]$, line 3

The validation of the second formula is similar.

4.5. A Slight Generalization

We need to slightly generalize the notion of simulation so that we can talk about a simulation from a distributed relation to another distributed relation. This is in preparation for thinking of a distributed relations as continuous relations. The old picture is the diagram

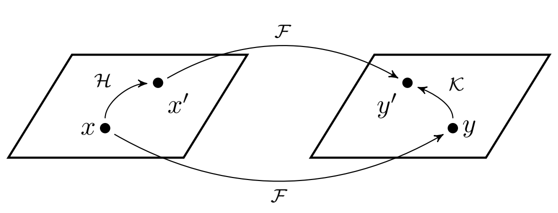

Figure 6: Intuitive view of Simulation Condition

and generalization where the upper arrow $\mathcal{F}$ is replaced by a new arrow $\mathcal{G}$:

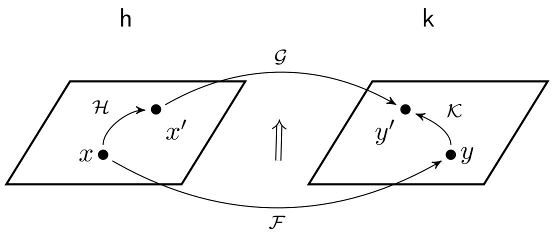

Figure 7: Intuitive view of Generalized Simulation Condition

where the pair $\mathcal{H}$ and $\mathcal{K}$ is thought of as a simulation from $\mathcal{F}$ to $\mathcal{G}$ satisfying

$$\mathcal{F}xy \text{ and } \mathcal{H}xx' \text{ implies } \exists y'(\mathcal{K}yy' \text{ and } \mathcal{G}x'y')$$

In the general case, the reverse simulation is

$$\mathcal{G}x'y' \text{ and } \breve{\mathcal{H}}x'x \text{ implies } \exists y(\breve{\mathcal{K}}y'y \text{ and } \mathcal{F}xy)$$

In view of latter work in this paper, we switch the use of $\mathcal{F}$ and $\mathcal{G}$ here so that the *proverse simulation* is

$$\mathcal{F}x'y' \text{ and } \breve{\mathcal{H}}x'x \text{ implies } \exists y(\breve{\mathcal{K}}y'y \text{ and } \mathcal{G}xy).$$

which is equivalent to

$$\mathcal{F}x'y' \text{ and } \mathcal{H}xx' \text{ implies } \exists y(\mathcal{K}yy' \text{ and } \mathcal{G}xy).$$

There are two more forms of simulation. We collect all the forms we use together here with their sentential prescriptions:

Definition 5.

Name	*Frame Condition*	*Axiom*
Forward Simulation	$\mathcal{F}xy$ and $\mathcal{H}xx'$ $\exists y'(\mathcal{K}yy'$ and $\mathcal{G}x'y')$	$\langle f\rangle[k]Q \overset{\mathsf{h}}{\supset} [h]\langle g\rangle Q$
Proverse Simulation	$\mathcal{F}x'y'$ and $\mathcal{H}xx'$ implies $\exists y(\mathcal{K}yy'$ and $\mathcal{G}xy)$	$\langle f\rangle[\cdot k\cdot]Q \overset{\mathsf{h}}{\supset} [\cdot h\cdot]\langle g\rangle Q$
Converse Simulation	$\breve{\mathcal{F}}yx$ and $\mathcal{K}yy'$ implies $\exists x'(\mathcal{H}xx'$ and $\breve{\mathcal{G}}y'x')$	$\langle\cdot f\cdot\rangle[h]P \overset{\mathsf{k}}{\supset} [k]\langle\cdot g\cdot\rangle P$
Backward Simulation	$\mathcal{F}x'y'$ and $\mathcal{K}yy'$ implies $\exists x(\mathcal{H}xx'$ and $\mathcal{G}xy)$	$\langle\cdot f\cdot\rangle[\cdot h\cdot]P \overset{\mathsf{k}}{\supset} [\cdot k\cdot]\langle\cdot g\cdot\rangle P$

Table 2: Frame Conditions and Axioms

Theorem 6. *The Frame Conditions validate their corresponding Axioms in the table.*

5. Rule Forms for Simulation

Working with entire spaces is reflected in the logic by the use of metalinguistic variables that quantify over all the propositions and respecting the localities. We had developed *spider connectives* for a different purpose in FPGA applications. A spider is a form of realtime monitor. We use them to prove facts about the design of an FPGA application. However, we did not want to trust that the hardware specification output by our hardware compiler did what we proved since many things can go wrong having nothing to with the design of the application itself, i.e., hardware faults, use of untrusted components, etc. Spiders were a way of watching over a design and as such are expressed in logical terms, not at the level of states and relations.

We will work with two spider connectives, $\xrightarrow{\langle f\rangle}$ and $\xrightarrow{[f]}$. The first has the modeling condition

$$x \overset{\mathsf{h}}{\models} P \xrightarrow{\langle f\rangle} Q \text{ iff } \exists u(u \overset{\mathsf{h}}{\models} P \text{ and } \exists y(\mathcal{F}uy \text{ and } y \overset{\mathsf{k}}{\models} Q))$$

and the second

$$x \overset{\mathsf{h}}{\models} P \xrightarrow{[f]} Q \text{ iff } \forall u(u \overset{\mathsf{h}}{\models} P \text{ implies } \forall y(\mathcal{F}uy \text{ implies } y \overset{\mathsf{k}}{\models} Q)).$$

Notice both evaluations throw away the original state x and evaluate globally. In [23], they describe a global modality that appears quite like an undistributed version of $P \xrightarrow{[f]} Q$. They say that this modality has been discovered numerous times and list some references for it. Clearer references are [6] and [24]. There are the following modeling conditions:

$$\mathfrak{M}, x \models \mathrm{E}\,\varphi \text{ iff } \mathfrak{M}, u \models \varphi \text{ for some world } u,$$

and

$$\mathfrak{M}, x \models \mathrm{A}\,\varphi \text{ iff } \mathfrak{M}, u \models \varphi \text{ for all worlds } u.$$

The model structure $\mathfrak{M}$, in our case, is a distributed Kripke model. It is also parametric so we drop the $\mathfrak{M}$. This yields, where P and Q on the right hand sides are, under interpretation, sets

$$\begin{aligned} x \models_{\mathsf{h}} P \xrightarrow{\langle f\rangle} Q &\text{ iff } \exists u(u \in_{\mathsf{h}} P \text{ and } \exists y(\mathcal{F}uy \text{ and } y \in_{\mathsf{k}} Q)) \\ &\text{ iff } \exists u(u \in P \text{ and } u \in_{\mathsf{h}} \langle f\rangle Q) \\ &\text{ iff } \models_{\mathsf{h}} \mathrm{E}\,(P \wedge [f]Q) \end{aligned}$$

and

$$\begin{aligned} x \models_{\mathsf{h}} P \xrightarrow{[f]} Q &\text{ iff } \forall u(u \in_{\mathsf{h}} P \text{ implies } \forall y(\mathcal{F}uy \text{ implies } y \in_{\mathsf{k}} Q)) \\ &\text{ iff } \forall u(u \in P \text{ implies } u \in_{\mathsf{h}} [f]Q) \\ &\text{ iff } \models_{\mathsf{h}} \mathrm{A}\,(P \supset [f]Q) \end{aligned}$$

Hence we can define the spider connectives as

$$P \xrightarrow{\langle f\rangle} Q \overset{\mathit{def}}{=} \mathrm{A}\,(P \wedge \langle f\rangle Q), \qquad P \xrightarrow{[f]} Q \overset{\mathit{def}}{=} \mathrm{E}\,(P \supset [f]Q).$$

The rule forms for simulations are as follows. Each has three equivalent forms.

5.1. *Forward Simulation*

$$\frac{Q \xrightarrow{[k]} S \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{\langle f\rangle Q \xrightarrow{[h]} \langle g\rangle S}$$

and

$$\frac{Q \xrightarrow{[k]} [\cdot g\cdot]R \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{\langle f\rangle Q \xrightarrow{[h]} R} \qquad\qquad \frac{\langle f\rangle Q \xrightarrow{\langle h\rangle} R \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{Q \xrightarrow{\langle k\rangle} \langle g\rangle R}$$

5.2. *Proverse Simulation*

$$\frac{Q \xrightarrow{[\cdot k\cdot]} S \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{\langle f\rangle Q \xrightarrow{[\cdot h\cdot]} \langle g\rangle S}$$

and

$$\frac{\langle\cdot g\cdot\rangle P \xrightarrow{[k]} S \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{P \xrightarrow{[h]} [f]S} \qquad\qquad \frac{P \xrightarrow{\langle h\rangle} \langle f\rangle S \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{\langle\cdot g\cdot\rangle P \xrightarrow{\langle k\rangle} S}$$

5.3. *Converse Simulation*

$$\frac{P \xrightarrow{[h]} R \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{\langle\cdot f\cdot\rangle P \xrightarrow{[k]} \langle\cdot g\cdot\rangle R}$$

and

$$\frac{P \xrightarrow{[h]} [g]S \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{\langle\cdot f\cdot\rangle P \xrightarrow{[k]} S} \qquad\qquad \frac{\langle\cdot f\cdot\rangle P \xrightarrow{\langle k\rangle} S \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{P \xrightarrow{\langle h\rangle} \langle g\rangle S}$$

5.4. *Backward Simulation*

$$\frac{R \xrightarrow{[\cdot h\cdot]} P \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{\langle\cdot f\cdot\rangle R \xrightarrow{[\cdot k\cdot]} \langle\cdot g\cdot\rangle P}$$

and

$$\frac{\langle g\rangle Q \xrightarrow{[h]} R \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{Q \xrightarrow{[k]} [\cdot f\cdot]R} \qquad\qquad \frac{Q \xrightarrow{\langle k\rangle} \langle\cdot f\cdot\rangle R \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{\langle g\rangle Q \xrightarrow{\langle h\rangle} R}$$

Theorem 7. *The Simulation Axioms and their respective Simulation Rules are interderivable.*

5.5. Residuation Rules

The following rules are valid for Normal Modal Distributed Logic

$$\begin{array}{c} \langle f\rangle Q \overset{\mathsf{h}}{\supset} R \qquad f : \mathsf{h} \rightharpoonup \mathsf{k} \\ \hline\hline Q \overset{\mathsf{k}}{\supset} [\cdot f\cdot]R \end{array} \qquad\qquad \begin{array}{c} \langle\cdot f\cdot\rangle P \overset{\mathsf{k}}{\supset} S \qquad f : \mathsf{h} \rightharpoonup \mathsf{k} \\ \hline\hline P \overset{\mathsf{h}}{\supset} [f]S \end{array}$$

where the double line indicates the rule can be read from top to bottom and from bottom to top. The reason these are in rule form rather than axiom form is that with Distributed Logic, there is no global logic to support a non-localized $\supset$ operator. The rules allow one to move from one local logic to another.

Simplifying the rules by identifying f and g, and including both forward and converse simulations, reveals the residuated nature between the possibility and necessity operators. The residuation rules mediated by the spider connectives are induced from

$$\begin{array}{c} Q \xrightarrow{[k]} [\cdot f\cdot]R \qquad f : \mathsf{h} \rightharpoonup \mathsf{k} \\ \hline \langle f\rangle Q \xrightarrow{[h]} R \end{array} \qquad\qquad \begin{array}{c} \langle f\rangle Q \xrightarrow{[h]} R \qquad f : \mathsf{h} \rightharpoonup \mathsf{k} \\ \hline Q \xrightarrow{[k]} [\cdot f\cdot]R \end{array}$$

yielding the following two sided residuation rule

$$\begin{array}{c} \langle f\rangle Q \xrightarrow{[h]} R \qquad f : \mathsf{h} \rightharpoonup \mathsf{k} \\ \hline\hline Q \xrightarrow{[k]} [\cdot f\cdot]R \end{array}$$

The interesting residuation rule is the following that normally does not resemble a residuation rule in modal logic:

$$\begin{array}{c} Q \xrightarrow{\langle k\rangle} \langle\cdot f\cdot\rangle R \qquad f : \mathsf{h} \rightharpoonup \mathsf{k} \\ \hline \langle f\rangle Q \xrightarrow{\langle h\rangle} R \end{array} \qquad\qquad \begin{array}{c} \langle f\rangle Q \xrightarrow{\langle h\rangle} R \qquad f : \mathsf{h} \rightharpoonup \mathsf{k} \\ \hline Q \xrightarrow{\langle k\rangle} \langle\cdot f\cdot\rangle R \end{array}$$

yields

$$\begin{array}{c} Q \xrightarrow{\langle k\rangle} \langle\cdot f\cdot\rangle R \qquad f : \mathsf{h} \rightharpoonup \mathsf{k} \\ \hline\hline \langle f\rangle Q \xrightarrow{\langle h\rangle} R \end{array}$$

If fact, if one imposes the rule

$$\begin{array}{c} Q \overset{\mathsf{k}}{\supset} \langle\cdot f\cdot\rangle R \qquad f : \mathsf{h} \rightharpoonup \mathsf{k} \\ \hline\hline \langle f\rangle Q \overset{\mathsf{h}}{\supset} R \end{array}$$

one is in effect forcing the underlying relation $\mathcal{F}$ to have extra properties. Reading the rule from bottom to top forces $\mathcal{F}$ to be surjective, and from top to bottom forces $\mathcal{F}$ to be injective.

The third residuation rule comes from equating $\mathcal{F}$ and $\mathcal{G}$ in the Proverse and Converse Simulation Rules (respectively)

$$\begin{array}{c} \langle\cdot g\cdot\rangle P \xrightarrow{[k]} S \qquad f, g : \mathsf{h} \rightharpoonup \mathsf{k} \\ \hline P \xrightarrow{[h]} [f]S \end{array} \qquad\qquad \begin{array}{c} P \xrightarrow{[h]} R \qquad f, g : \mathsf{h} \rightharpoonup \mathsf{k} \\ \hline \langle\cdot f\cdot\rangle P \xrightarrow{[k]} \langle\cdot g\cdot\rangle R \end{array}$$

yielding

$$\frac{\langle f\rangle P \xrightarrow{[k]} S \qquad f : \mathsf{h} \rightharpoonup \mathsf{k}}{P \xrightarrow{[h]} [f]S}$$

The last residuation rule stemming from the last of the rules for Proverse and Converse Simulation (respectively)

$$\frac{P \xrightarrow{\langle h\rangle} \langle f\rangle S \qquad f, g : \mathsf{h} \rightharpoonup \mathsf{k}}{\langle g\rangle P \xrightarrow{\langle k\rangle} S} \qquad\qquad \frac{\langle f\rangle P \xrightarrow{\langle k\rangle} S \qquad f, g : \mathsf{h} \rightharpoonup \mathsf{k}}{P \xrightarrow{\langle h\rangle} \langle g\rangle S}$$

is

$$\frac{P \xrightarrow{\langle h\rangle} \langle f\rangle S \qquad f : \mathsf{h} \rightharpoonup \mathsf{k}}{\langle f\rangle P \xrightarrow{\langle k\rangle} S}$$

The analogous classical rule

$$\frac{P \overset{\mathsf{h}}{\supset} \langle f\rangle S \qquad f : \mathsf{h} \rightharpoonup \mathsf{k}}{\langle f\rangle P \overset{\mathsf{k}}{\supset} S}$$

also forces the underlying relation $\mathcal{F}$ to have more properties. Reading the rule from top to bottom forces $\mathcal{F}$ to be a partial function, and from bottom to top forces $\mathcal{F}$ to be entire, i.e., every element of the domain of $\mathcal{F}$ is mapped to some element of the codomain.

Using similar reasoning in a selection of axioms and rules, one can force many properties of the morphisms in the underlying interpretation just as in traditional modal logic.

6. Homotopy and Bisimulation

There are three notions from homotopy theory with which we are concerned. From Munkres [25], where we have made slight notation changes,

Definition 8.

- ***Homotopic Maps:*** *If f and g are continuous maps on the space X into the space Y, we say that f is* homotopic *to g if there is a continuous map $F : X \times I \rightarrow Y$ such that*

$$F(x, 0) = f(x) \quad \text{and} \quad F(x, 1) = g(x)$$

  for each $x \in X$. (Here $I = [0, 1]$.) The map F is called the homotopy *between f and g. If f is homotopic to g, we write $f \simeq g$.*

- ***Homotopy Equivalence:*** *A continuous map $f : X \rightarrow Y$ is called a* homotopy equivalence *if there is a continuous map $g : Y \rightarrow X$ such that $g \circ f$ is homotopic to the identity map i_X of X and $f \circ g$ is homotopic to the identity map i_Y of Y. The map g is said to be a* homotopy inverse *for the map f.*

- ***Path Homotopy:*** *Two paths f and f', mapping the interval $I = [0,1]$ into X, are said to be* path homotopic *if they have the same initial point x_0 and the same final point x_1, and if there is a continuous map $F : I \times I \to X$ such that*
$$F(s,0) = f(s) \quad \text{and} \quad F(s,1) = f'(s),$$
$$F(0,t) = x_0 \quad \text{and} \quad F(1,t) = x_1$$
for each $s,t \in I$. F is the path homotopy *between f and f' and we write $f \simeq_p f'$.*

We will use Stone spaces each with a local Kripke relation as topological spaces and distributed Kripke relations in place of continuous maps. The topology on the Stone space is almost beside the point. Homotopy theory combines connectivity on a space with the topology on a space. Here, these are separated and the endo-relations $\mathcal{H}$ hold the connectivity. In talking about path homotopy, this will be generalized a bit to allow a distributed relation connectivity.

A distributed relation $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$ is continuous since it will take any clopen set Q of a Stone space into another clopen set $\mathcal{F}^{-1}Q$. In modal set algebras, this is displayed as $\langle f \rangle Q$. The semantic connection between Distributed Logic and distributed Kripke frame maps (abusing the notation a bit) takes the proposition Q to the clopen set $\langle f \rangle Q$.

We will generally ignore the completeness nature $[0,1]$ (in the sense of analysis of real spaces) in the formulations that follow. One could always go to yet another level of indexing beyond the graph of Stone spaces that we use, but it does not appear there is much to gain for the purposes this paper.

In general, we make the following replacements

homotopic	::	bisimilar	homotopy	::	bisimulation
map	::	distributed relation	path	::	endo-relation

Table 3: Relationships between Homotopy and Simulation Logic

although we generalize the notion of path a bit in Section 6.3 on Path Bisimilarity.

We need to make two accommodations in going from homotopy to bisimilarity. The first accommodation concerns an endo-relation, say $\mathcal{H}$ at h, containing all the connection structure. A path should be a subset of $\mathcal{H}$. To do so requires specifying a partial order at a locality as syntactic structure as done in [18] with axioms of the form
$$[h]P \overset{\mathsf{h}}{\supset} [h']P, \qquad h \geq h'$$
where $[h']$ is interpreted by a relation $\mathcal{H}'$ such that $\mathcal{H}' \subseteq \mathcal{H}$. This does not complicate the logic and modeling we do in the sequel other than to replace $\mathcal{H}$ with $\mathcal{H}'$. In view of this, we will simply assume $\mathcal{H}$ to be a path, it simply happens to contain all the connection structure. The more involved case of using a partial order may be useful for some application areas.

The second accommodation concerns end points of paths. Relations do not generally have end points. Ronald Brown [26] introduced in 1967 a set of base points into homotopy theory. We adopt the usage of a set of base points, namely as a collection of points that are in the domain, dom($\mathcal{F}$), of a relation. Let $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$, then

$$\mathrm{dom}(\mathcal{F}) \overset{def}{=} \{u \mid \exists v(\mathcal{F}uv)\}.$$

The set $\mathrm{dom}\,\breve{\mathcal{F}}$ is defined similarly and we use this rather than define a range function. We also require that there are certain subsets of the diagonal relation:

Definition 9. *$\mathcal{J}_{\mathcal{F}}$ is a subset of the diagonal satisfying*

$$\mathcal{J}_{\mathcal{F}} \subseteq \mathcal{I}_{\mathsf{h}} \quad \text{and} \quad \mathrm{dom}(\mathcal{J}_{\mathcal{F}}) \overset{def}{=} \mathrm{dom}(\mathcal{F}).$$

Bisimilar Distributed Relations. A map f homotopic to g has the following intuitive diagram for all s

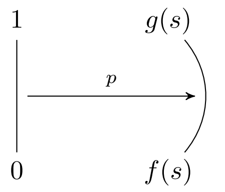

Figure 8: Path Between Points of Continuous Functions

The path is from $p(0)$ $(= f(s))$ to $p(1)$ $(= g(s))$. This path is also considered reversible. We replace $X \times [0,1]$ by a Stone space at h, and Y by a Stone space at k. In place of s we will use the set dom($\mathcal{H}$) and the identity relation $\mathcal{I}_{\mathsf{h}}$. The path p will be replaced with the relation $\mathcal{K}$ on the Stone space at k. We replace the continuous maps f and g with the continuous relations $\mathcal{F}, \mathcal{G} : \mathsf{h} \rightharpoonup \mathsf{k}$. If we require that $\mathcal{K}$ and $\breve{\mathcal{K}}$ form a bisimulation relation between $\mathcal{F}$ and $\mathcal{G}$, then the bi of bisimulation picks up the reversibility of the path. In symbols, we denote this

$$\mathcal{F} \overset{\mathcal{K}}{\simeq} \mathcal{G}.$$

Unfortunately, this is not enough. The homotopy relation between maps must be an equivalence relation. In order to satisfy this, $\mathcal{K}$ must be reflexive and transitive. As a bisimulation, we have the symmetry

$$\mathcal{F} \overset{\mathcal{K}}{\simeq} \mathcal{G} \text{ iff } \mathcal{G} \overset{\breve{\mathcal{K}}}{\simeq} \mathcal{F}.$$

We will instead use the Kleene * applied to $\mathcal{K}$. This is reflexive and transitive and since $(\mathcal{K}^*)\breve{} = \breve{\mathcal{K}}^*$, we will also get symmetry. Hence we will depict the bisimulations between distributed relations $\mathcal{F}, \mathcal{G} : \mathsf{h} \rightharpoonup \mathsf{k}$ with

$$\mathcal{F} \overset{\mathcal{K}^*}{\simeq} \mathcal{G}.$$

We use the convention that the output type of $\mathcal{F}$ and $\mathcal{G}$, namely k, determines that $\mathcal{K}^*$ be used in the depiction.

Simulation Equivalence. A homotopy equivalence between spaces requires that each space's connection structure be represented or mapped to by the other. We model this as a simulation $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$ and a simulation $\mathcal{G} : \mathsf{h} \rightharpoonup \mathsf{k}$. In addition, we require that

$$\mathcal{G} \circ \mathcal{F} \overset{\mathcal{R}}{\simeq} \mathcal{I}_{\mathsf{h}}, \quad \text{and} \quad \mathcal{I}_{\mathsf{k}} \overset{\mathcal{S}}{\simeq} \mathcal{F} \circ \mathcal{G},$$

for some bisimulations $\mathcal{R}$ and $\mathcal{S}$.

Path Bisimilarity. Considering a path homotopy between f and f', we first separate f and f' as being paths in two different spaces. The map F then represents a continuum of homotopies (and consequently spaces) between $\lambda s.F(s,t)$ and $\lambda s.F(s,t')$ for any two $t, t' \in [0,1]$. We will only consider the case for two specific t and t' which are essentially labels for the space containing the path f and the space containing the space f'. Since Distributed Logic uses a graph of spaces, we will use h and k for our two spaces.

The two relations $\mathcal{H}$ and $\mathcal{K}$ will be substituted for the two paths $\lambda s.F(s,t)$ and $\lambda s.F(s,t')$ respectively. The specification says there should be a homotopy between them, hence we require the bisimulation.

$$\mathcal{H} \overset{\mathcal{F}}{\simeq} \mathcal{K},$$

where $\mathcal{F}$ is substituted for the map $F(-,t)$ to $F(-,t')$, which is the map between $f = \lambda s.F(s,t)$ and $f' = \lambda s.F(s,t')$ with $\mathcal{H}$ for f and $\mathcal{K}$ for f'.

The parameter t requires that there be a collection of bisimulations between $\mathcal{H}$ and $\mathcal{K}$ with the completeness property of $[0,1]$. This could be added with indexing $\mathcal{F}$, i.e., $\mathcal{F}_t$. It will not change the fundamental prescription for the bisimulation requirement and we elide it.

6.1. Bisimilar Distributed Relations (Homotopy Maps)

A first pass at a bisimulation between distributed relations, let $\mathcal{F}, \mathcal{G} : \mathsf{h} \rightharpoonup \mathsf{k}$ be a bisimulation from $\mathcal{F}$ to $\mathcal{G}$ is a relation $\mathcal{K}$ such that

$$\mathcal{I}_{\mathsf{h}}xx \text{ and } \mathcal{F}xy \text{ implies } \exists y'(\mathcal{G}xy' \text{ and } \mathcal{K}yy') \text{ and}$$
$$\mathcal{I}_{\mathsf{h}}xx \text{ and } \mathcal{G}xy' \text{ implies } \exists x'(\mathcal{F}xy \text{ and } \breve{\mathcal{K}}y'y)$$

In diagrams,

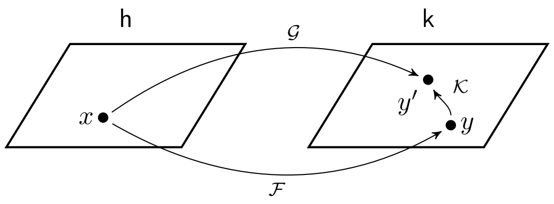

Figure 9: Forward Simulation for Continuous Relations

and

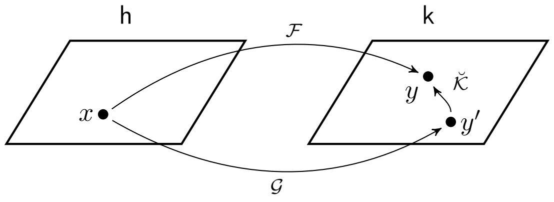

Figure 10: Converse Simulation for Continuous Relations

Lemma 10. *The conditions above validate*

$$\langle f\rangle[k]Q \subseteq [i_{\mathsf{h}}]\langle g\rangle Q, \quad \text{and} \quad \langle g\rangle[\cdot k\cdot]Q \subseteq [i_{\mathsf{h}}]\langle f\rangle Q.$$

However, homotopies between continuous maps must form an equivalence relation. This requires that the relation $\mathcal{K}$ be composed with itself for transitivity yet still be $\mathcal{K}$. Reflexivity requires $\mathcal{K}$ be reflexive. The bisimulation is

$$\mathcal{F} \overset{\mathcal{K}^*}{\simeq} \mathcal{G}.$$

We need the following property of reflexive-transitive closure for an endo-relation:

Definition 11. *The following formula defines reflexive-transitive closure for an endo-relation $\mathcal{K}$:*

$$\mathcal{K}^* \overset{def}{=} \bigcup_{i\geq 0} \mathcal{K}^i$$

where $\mathcal{K}^0 = \mathcal{I}_{\mathsf{k}}$.

Reflexive-transitive closure for a distributed relation, say $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$ does not make sense since it cannot be composed with itself.

Lemma 12.

$$(\mathcal{K}^*)^{\smile} = \breve{\mathcal{K}}^*.$$

where $^{\smile}$ binds tighter than *, and

$$\mathcal{K}^* \circ \mathcal{K}^* = \mathcal{K}^*.$$

Kleene * operator distributes over unions and converse where the latter is $(\mathcal{K} \circ \mathcal{K}')^{\smile} = \breve{\mathcal{K}}' \circ \breve{\mathcal{K}}$. The second property falls easily out of the definition. We then have the following theorem:

Theorem 13. *For $\mathcal{F}, \mathcal{G}, \mathcal{P} : \mathsf{h} \rightharpoonup \mathsf{k}$*

- $\mathcal{F} \overset{\mathcal{K}^*}{\simeq} \mathcal{F}$,
- $\mathcal{F} \overset{\mathcal{K}^*}{\simeq} \mathcal{G}$ implies $\mathcal{G} \overset{\mathcal{K}^*}{\simeq} \mathcal{F}$,

- $\mathcal{F} \overset{\mathcal{K}^*}{\simeq} \mathcal{G}$ and $\mathcal{G} \overset{\mathcal{K}^*}{\simeq} \mathcal{P}$ implies $\mathcal{F} \overset{\mathcal{K}^*}{\simeq} \mathcal{P}$.

Theorem 14. *The formulas*

$$\langle f \rangle [k^*] Q \subseteq [i_{\mathsf{h}}] \langle g \rangle Q, \qquad \langle g \rangle [\cdot k^* \cdot] Q \subseteq [i_{\mathsf{h}}] \langle f \rangle Q.$$

are validated by

$$\mathcal{I}_{\mathsf{h}} xx \text{ and } \mathcal{F}xy \text{ implies } \exists y'(\mathcal{G}xy' \text{ and } \mathcal{K}^* yy'), \quad \text{and}$$
$$\mathcal{I}_{\mathsf{h}} xx \text{ and } \mathcal{G}xy' \text{ implies } \exists x'(\mathcal{F}xy \text{ and } \mathcal{K}^* y'y)$$

6.2. Simulation Equivalence (Homotopy Equivalence)

Two spaces X and Y are *simulation equivalent* if there are two continuous relations $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$ and $\mathcal{G} : \mathsf{k} \rightharpoonup \mathsf{h}$ such that they can act somewhat like weak inverses of each other and each specifies that the connection structure of their source is simulated in the connection structure of their target:

Definition 15. *Let* $f : \mathsf{h} \rightharpoonup \mathsf{k}$ *and* $g : \mathsf{k} \rightharpoonup \mathsf{h}$*. The two localities* h *and* k *are* bisimulation equivalent *when these two formula are valid*

$$\langle f \rangle [k] Q \supset [h] \langle f \rangle Q, \qquad \langle g \rangle [h] P \supset [k] \langle g \rangle P.$$

and the following four formulas are valid

$$\langle r \rangle [j_f] P \subseteq [f \circ g] \langle r \rangle P, \qquad \langle r \rangle [f \circ g] P \subseteq [j_f] \langle r \rangle P.$$

and

$$\langle s \rangle [j_g] Q \subseteq [g \circ f] \langle s \rangle Q, \qquad \langle s \rangle [g \circ f] Q \subseteq [j_g] \langle s \rangle Q.$$

The first pair of formulas require that the interpreting relations $\mathcal{F}$ and $\mathcal{G}$ are simulations. The other pairs of formula state that the interpreting relations $\mathcal{F}$ and $\mathcal{G}$ act like weak inverses.

Theorem 16. *The formulas*

$$\langle f \rangle [k] Q \supset [h] \langle f \rangle Q, \qquad \langle g \rangle [h] P \supset [k] \langle g \rangle P.$$

are validated by

$$\mathcal{F}xy \text{ and } \mathcal{H}xx' \text{ implies } \exists y'(\mathcal{K}yy' \text{ and } \mathcal{F}x'y'), \quad \text{and}$$
$$\mathcal{G}yx \text{ and } \mathcal{K}yy' \text{ implies } \exists x'(\mathcal{H}xx' \text{ and } \mathcal{G}y'x').$$

where $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$ *and* $\mathcal{G} : \mathsf{k} \rightharpoonup \mathsf{h}$.

The validation proof is the usual sort of validation of simulation formulas. The second two pairs of formulas in the Definition require a bit more work to which we now attend.

Theorem 17. *The formulas*

$$\langle r\rangle[j_f]P \subseteq [f\circ g]\langle r\rangle P, \qquad \langle r\rangle[f\circ g]P \subseteq [j_f]\langle r\rangle P.$$

and

$$\langle s\rangle[j_g]Q \subseteq [g\circ f]\langle s\rangle Q, \qquad \langle s\rangle[g\circ f]Q \subseteq [j_g]\langle s\rangle Q.$$

are validated with bisimulations $\mathcal{R}$ and $\mathcal{S}$ satisfying

$$\mathcal{G}\circ\mathcal{F} \overset{\mathcal{R}}{\simeq} \mathcal{J}_{\mathcal{F}}, \qquad \mathcal{J}_{\mathcal{G}} \overset{\mathcal{S}}{\simeq} \mathcal{F}\circ\mathcal{G},$$

using the two bisimulation relations $\mathcal{R}$ and $\mathcal{S}$ and $\mathcal{J}_{\mathcal{F}}$ is from Definition 9. $\mathcal{J}_{\mathcal{G}}$ is defined similarly. The operators $[j_f]$ and $[j_g]$ are the modal necessity operators validated by $\mathcal{J}_{\mathcal{F}}$ and $\mathcal{J}_{\mathcal{G}}$ respectively.

The validation proof again the usual sort of validation of simulation formulas. The relations $\mathcal{R}$ and $\mathcal{S}$ can be manufactured from $\mathcal{F}$ and $\mathcal{G}$ with a little help from imposing the formulas in the following Lemma as axioms:

Lemma 18. *Let $f : \mathsf{h} \rightharpoonup \mathsf{k}$ and $g : \mathsf{k} \rightharpoonup \mathsf{h}$. The formulas*

$$[g\circ f]Q \subseteq [\mathcal{J}_{\mathcal{G}}]Q \quad \text{and} \quad [f\circ g]P \subseteq [\mathcal{J}_{\mathcal{F}}]P$$

are validated by the conditions

$$\mathcal{J}_{\mathcal{G}} \subseteq \mathcal{F}\circ\mathcal{G} \quad \text{and} \quad \mathcal{J}_{\mathcal{F}} \subseteq \mathcal{G}\circ\mathcal{F}.$$

respectively

The proof is straight-forward.

The following Lemma is a bit technical and used in the proofs of simulation equivalence.

Lemma 19. *Let $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$ and $\mathcal{G} : \mathsf{k} \rightharpoonup \mathsf{h}$, then*

$$\mathcal{R}ux \text{ and } (\mathcal{G}\circ\mathcal{F})xx' \text{ implies } u \in \mathrm{dom}(\mathcal{F}) \text{ and } \langle u,u\rangle \in \mathcal{J}_{\mathcal{F}}.$$

and

$$\mathcal{S}vy \text{ and } (\mathcal{F}\circ\mathcal{G})yy' \text{ implies } v \in \mathrm{dom}(\mathcal{G}) \text{ and } \langle v,v\rangle \in \mathcal{J}_{\mathcal{G}}.$$

Proof. We prove the second statement as the first is similar. We have

$$\mathcal{S} = \bigcup_{i\geq 0}(\mathcal{F}\circ\mathcal{G})^i.$$

Assume $\mathcal{S}vy$ and $(\mathcal{F}\circ\mathcal{G})yy'$. Case 1 is $n = 0$ and Case 2 is $n \geq 1$.

Case 1: $\langle v,y\rangle \in (\mathcal{F}\circ\mathcal{G})^0$. Hence $\langle v,y\rangle \in \mathcal{I}_{\mathsf{k}}$ since $(\mathcal{F}\circ\mathcal{G})^0 = \mathcal{I}_{\mathsf{k}}$. Therefore, $v = y$ and $(\mathcal{F}\circ\mathcal{G})vy'$ from the assumptions. This latter in relational composition is $(\mathcal{G}\cdot\mathcal{F})vy'$, and there is some w such that $\mathcal{G}vw$ and $\mathcal{F}wy'$. In this case, $v \in \mathrm{dom}(\mathcal{G})$.

Case 2: $\langle v, y\rangle \in (\mathcal{F}\circ\mathcal{G})^n$ for some $n \geq 1$. Hence $\langle v, y\rangle \in (\mathcal{F}\circ\mathcal{G})^{n-1}\circ(\mathcal{F}\circ\mathcal{G})$. This latter in relational composition is $\langle v, y\rangle \in (\mathcal{G}\cdot\mathcal{F})\cdot(\mathcal{G}\cdot\mathcal{F})^{n-1}$, so there is some w such that $\langle v, w\rangle \in \mathcal{G}\cdot\mathcal{F}$ and $\langle w, y\rangle \in (\mathcal{G}\cdot\mathcal{F})^{n-1}$. Further, there is some z such that $\langle v, z\rangle \in \mathcal{G}$ and $\langle z, w\rangle \in \mathcal{F}$. From $\langle v, z\rangle \in \mathcal{G}$, $v \in \mathrm{dom}(\mathcal{G})$.

For both cases, since $\mathrm{dom}(\mathcal{J}_\mathcal{G}) = \mathrm{dom}(\mathcal{G})$, then $v \in \mathrm{dom}(\mathcal{J}_\mathcal{G})$ and $\langle v, v\rangle \in \mathcal{J}_\mathcal{G}$.

Lemma 20. *Let $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$ and $\mathcal{G} : \mathsf{k} \rightharpoonup \mathsf{h}$.*
If

$$\mathcal{R} \overset{def}{=} \bigcup_{i\geq 0}(\mathcal{G}\circ\mathcal{F})^i, \text{and } \mathcal{J}_\mathcal{F} \overset{def}{=} \{\langle x, x\rangle \mid x \in \mathrm{dom}(\mathcal{F})\}, \text{and } \mathcal{J}_\mathcal{F} \subseteq (\mathcal{G}\circ\mathcal{F}),$$

then

$$\mathcal{G}\circ\mathcal{F} \overset{\breve{\mathcal{R}}}{\simeq} \mathcal{J}_\mathcal{F}$$

and note that $\breve{\mathcal{R}}$ is use in the latter formula rather than $\mathcal{R}$.
If

$$\mathcal{S} \overset{def}{=} \bigcup_{i\geq 0}(\mathcal{F}\circ\mathcal{G})^i, \text{and } \mathcal{J}_\mathcal{G} \overset{def}{=} \{\langle y, y\rangle \mid y \in \mathrm{dom}(\mathcal{G})\}, \text{and } \mathcal{J}_\mathcal{G} \subseteq (\mathcal{F}\circ\mathcal{G}),$$

then

$$\mathcal{J}_\mathcal{G} \overset{\breve{\mathcal{S}}}{\simeq} \mathcal{F}\circ\mathcal{G}.$$

Similarly, we use $\breve{\mathcal{S}}$ rather than $\mathcal{S}$.

Proof. The proof of the first statement is similar to the proof of the second statement. We show that $\mathcal{S}$ satisfies

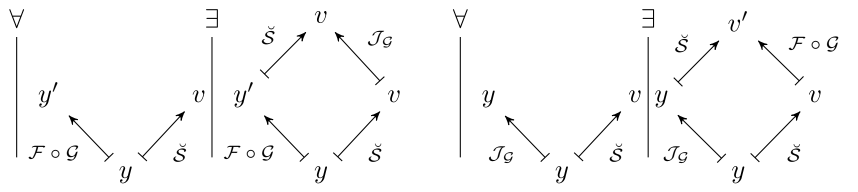

Figure 11: Conditions for Simulation Equivalence

Assume the premises and let $(\mathcal{F}\circ\mathcal{G})yy'$ and $\breve{\mathcal{S}}yv$, then from relational converse, $\mathcal{S}vy$ holds. Using the categorical composition of relations $((\mathcal{F}\circ\mathcal{G})\circ\mathcal{S})vy'$. From relational transitivity, it follows that $\mathcal{S}vy'$. From $\mathcal{S}vy$ and $(\mathcal{F}\circ\mathcal{G})yy'$, Lemma 19 yields $\mathcal{J}_\mathcal{G}vv$. Using relational converse again, it follows that $\mathcal{S}vy'$ and $\mathcal{J}_\mathcal{G}vv$ as desired.

For the next condition, assume the premises and let $\breve{\mathcal{S}}yv$ and $\mathcal{J}_{\breve{\mathcal{F}}}yy$. From relational converse, $\mathcal{S}vy$. The definition of reflexive-transitive closure means that $(\mathcal{F}\circ\mathcal{G})^n vy$ for some $n \geq 0$. We divide the proof into the case $n = 0$ or $n \geq 1$. Let $n = 0$, then $(\mathcal{F}\circ\mathcal{G})^0 = \mathcal{I}_\mathsf{k}$, and so $y = v$ and $\mathcal{J}_\mathcal{G}vv$. From the premises, $\mathcal{J}_\mathcal{G} \subseteq (\mathcal{F}\circ\mathcal{G})$ and so $(\mathcal{F}\circ\mathcal{G})vv$. Letting v' be v, it follows that

$\mathcal{S}yv'$ and $(\mathcal{F}\circ\mathcal{G})vv$. Next, let $n \geq 1$, hence $(\mathcal{F}\circ\mathcal{G})vv'$ and $(\mathcal{F}\circ\mathcal{G})^{n-1}v'y$ for some v'. Since $(\mathcal{F}\circ\mathcal{G})^{n-1} \subseteq \mathcal{S}$, it follows that $\mathcal{S}v'y$. Hence $(\mathcal{F}\circ\mathcal{G})vv'$ and $\breve{\mathcal{S}}yv'$ obtains.

Since $(\mathcal{F}\circ\mathcal{G})vv'$ and $\breve{\mathcal{S}}yv'$ obtains from both branches of the proof, the condition is satisfied.

6.3. Path Bisimilarity (Path Homotopy)

As mentioned previously, one of the accommodations necessary is that paths become relations. If the endo-relation $\mathcal{H}$ is all of the connection structure at h, then a path should be some subset of $\mathcal{H}$. This amounts to using axioms of the form

$$[h]P \overset{\mathsf{h}}{\supset} [h']P, \qquad h' \leq h$$

where $[h']$ is interpreted by a relation $\mathcal{H}'$ such that $\mathcal{H}' \subseteq \mathcal{H}$. However, we may also consider $\mathcal{H}$ itself as a path. Since nothing in bisimilarity of paths relies upon being a subset of a endo-relation, we simply choose to present the paths in bisimilarity using the endo-relations $\mathcal{H}$ and $\mathcal{K}$. In order to not have a trivial path bisimilarity relation at $\mathcal{H}$, we will assume that these subsets exist. This will also lend extra structure to the path bisimilarity relation, but we will ignore that for this paper.

Bisimilarity of local paths is rendered by the following theorem:

Theorem 21.

$$\langle f\rangle[k]Q \supset [h]\langle f\rangle Q, \quad \text{and} \quad \langle f\rangle[h]P \supset [k]\langle f\rangle P.$$

are validated by

$$\mathcal{F}xy \text{ and } \mathcal{H}xx' \text{ implies } \exists y'(\mathcal{K}yy' \text{ and } \mathcal{F}x'y') \text{ and}$$
$$\breve{\mathcal{F}}yx \text{ and } \mathcal{K}yy' \text{ implies } \exists x'(\mathcal{H}xx' \text{ and } \breve{\mathcal{F}}y'x'),$$

respectively.

We use

$$\mathcal{H} \overset{\mathcal{F}}{\simeq} \mathcal{K},$$

as the short form for the condition.

The proofs are the usual sort of simulation proofs. It is the quantification over all Q and P in the two formulas that achieves the equivalence in *shape* between the relations $\mathcal{H}$ and $\mathcal{K}$ as paths.

6.4. The Traditional Fundamental Groupoid

From Munkres [25], path homotopy is an equivalence relation. In particular, given two paths, f and g. Identity and symmetry are easy. Two homotopies are easily composed as are two path homotopies, hence transitivity obtains as well. We will use the notation $[f]$ for the equivalence class under path homotopy equivalence.

Define

$$[f] \star [g] \overset{def}{=} [f \circ g]$$

where $\circ$ is functional composition. Clearly $([f] \star [g]) \star [h] = [f] \star ([g] \star [h])$. The identity e_x is a path that starts at x and goes nowhere else. Paths have converses, i.e., functions $f : [0, 1] \rightarrow X$ turns into $\breve{f} : [0, 1] \rightarrow X$ with $\breve{f}(t) = f(1 - t)$. Also if $f(0) = x_0$ and $f(1) = x_1$, then

$$[f] \star [e_{x_0}] = [f], \quad \text{and} \quad [e_{x_1}] \star [f] = [f],$$

and

$$[f] \star [\breve{f}] = [e_{x_1}], \quad \text{and} \quad [\breve{f}] \star [f] = [e_{x_0}].$$

where e_{x_0} and e_{x_1} are paths that go nowhere but the points in their subscripts.

6.5. Fundamental Groupoid for Distributed Logic

The fundamental groupoid works with equivalence classes, these will be equivalence classes of bisimulations. Let $\mathcal{W}, \mathcal{V} \subseteq \mathcal{H}$ be endo-relations at h, then $\mathcal{W} \overset{\mathcal{R}}{\simeq} \mathcal{V}$ implies $\mathcal{V} \in [\mathcal{W}]$ where $[\mathcal{W}]$ is the bisimulation equivalence class containing the representative $\mathcal{W}$. The particular $\mathcal{R}$ is immaterial, there just needs to exist a bisimulation $\mathcal{W} \rightharpoonup \mathcal{V}$. We will first show some more general properties of bisimulations between localities and then define the fundamental groupoid

Relational composition respects bisimulation. This shows for bisimulations $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$ and $\mathcal{G} : \mathsf{k} \rightharpoonup \mathsf{l}$, that $\mathcal{G} \circ \mathcal{F}$ is again a bisimulation. Were we to restrict to bisimulations of endo-relations, say $\mathcal{W}, \mathcal{Z} \subseteq \mathcal{H}$ at h, then

$$[\mathcal{W}] \star [\mathcal{Z}] \overset{def}{=} [\mathcal{W} \circ \mathcal{Z}],$$

where $[\mathcal{W}]$ and $[\mathcal{Z}]$ are the bisimulation equivalence classes for relations on h.

We only need to show simulations compose as converse simulations are similar. A bisimulation is a forward simulation and a converse simulation where converse uses the converse of the simulation relation. The intuitive diagram is

$\forall$

$$\begin{tikzcd}
 & & & v \\
x' & & y \arrow[ur, mapsto, "\mathcal{G}"'] & \\
 & x \arrow[ul, mapsto, "\mathcal{H}"] \arrow[ur, mapsto, "\mathcal{F}"'] & &
\end{tikzcd}$$

$\exists$

$$\begin{tikzcd}
 & & v' & \\
 & y' \arrow[ur, mapsto, "\mathcal{G}"] & & v \arrow[ul, mapsto, "\mathcal{L}"'] \\
x' \arrow[ur, mapsto, "\mathcal{F}"] & & y \arrow[ul, mapsto, "\mathcal{K}"'] \arrow[ur, mapsto, "\mathcal{G}"'] & \\
 & x \arrow[ul, mapsto, "\mathcal{H}"] \arrow[ur, mapsto, "\mathcal{F}"'] & &
\end{tikzcd}$$

Figure 12: Composing Bisimulations

The follow Lemma is well-known and the proof is simple.

Lemma 22. *Let $\mathcal{H} : \mathsf{h} \rightharpoonup \mathsf{h}$, $\mathcal{K} : \mathsf{k} \rightharpoonup \mathsf{k}$ and $\mathcal{L} : \mathsf{l} \rightharpoonup \mathsf{l}$. If $\mathcal{F} : \mathcal{H} \rightharpoonup \mathcal{K}$ and $\mathcal{G} : \mathcal{K} \rightharpoonup \mathcal{L}$ are bisimulations, then $\mathcal{G} \circ \mathcal{F} : \mathcal{H} \rightharpoonup \mathcal{L}$ is a bisimulation.*

Bisimulations on our interpretation of paths, i.e., endo-relations such as $\mathcal{H}$ and $\mathcal{K}$, satisfy the following:

Theorem 23. *Let $\mathcal{F} : \mathcal{H} \rightharpoonup \mathcal{K}$, $\mathcal{G} : \mathcal{K} \rightharpoonup \mathcal{L}$,*

- $\mathcal{H} \overset{\mathcal{I}_\mathsf{h}}{\simeq} \mathcal{H}$,
- $\mathcal{H} \overset{\mathcal{F}}{\simeq} \mathcal{K}$ implies $\mathcal{K} \overset{\breve{\mathcal{F}}}{\simeq} \mathcal{H}$,
- $\mathcal{H} \overset{\mathcal{F}}{\simeq} \mathcal{K} \overset{\mathcal{G}}{\simeq} \mathcal{L}$ implies $\mathcal{H} \overset{\mathcal{G}\circ\mathcal{F}}{\simeq} \mathcal{L}$.

The first is true since $\mathcal{I}_\mathsf{h}$ is a bisimulation relation on $\mathcal{H}$. The second holds because $\mathcal{F} : \mathcal{H} \rightharpoonup \mathcal{K}$ being a bisimulation implies $\breve{\mathcal{F}} : \mathcal{K} \rightharpoonup \mathcal{H}$ is a bisimulation. The third holds because bisimulations compose to yield bisimulations.

Corollary 24. *For purposes of this Corollary, let $\mathcal{H}, \mathcal{K}, \mathcal{L} : \mathsf{h} \rightharpoonup \mathsf{h}$ in the Lemma, then $\simeq$, indicating the existence of a bisimulation relation, is an equivalence relation.*

The three properties to be satisfied by a fundamental groupoid are satisfied by those as well. The first is satisfied as the result of the following Corollary:

Corollary 25. *Let $\mathcal{H}, \mathcal{W}, \mathcal{V} : \mathsf{h} \rightharpoonup \mathsf{h}$ be endo-relations at h, then*

$$[\mathcal{H}] \star ([\mathcal{W}] \star [\mathcal{V}]) = ([\mathcal{H}] \star [\mathcal{W}]) \star [\mathcal{V}].$$

The unit paths e_{x_0} and e_{x_1} from traditional path homotopy, will be modeled as $\mathcal{J}_\mathcal{F}$ and $\mathcal{J}_{\breve{\mathcal{F}}}$. Hence the groupoid on distributed Kripke frames requires, for two bisimilarity relation $\mathcal{R}$ and $\mathcal{S}$, that

$$\breve{\mathcal{F}} \circ \mathcal{F} \overset{\mathcal{R}}{\simeq} \mathcal{J}_\mathcal{F} \qquad \mathcal{J}_{\breve{\mathcal{F}}} \overset{\mathcal{S}}{\simeq} \mathcal{F} \circ \breve{\mathcal{F}}.$$

We show that $\breve{\mathcal{F}} \circ \mathcal{F} \simeq \mathcal{J}_\mathcal{F}$ and $\mathcal{F} \circ \breve{\mathcal{F}} \simeq \mathcal{J}_{\breve{\mathcal{F}}}$ via reflexive-transitive closures $(\breve{\mathcal{F}} \circ \mathcal{F})^*$ and $(\mathcal{F} \circ \breve{\mathcal{F}})^*$ respectively, where, decoding the second accommodation we made at the beginning of this section,

$$\mathcal{J}_\mathcal{F} = \{\langle u, u\rangle \mid \exists v(\mathcal{F}uv)\} \qquad \mathcal{J}_{\breve{\mathcal{F}}} = \{\langle v, v\rangle \mid \exists u(\mathcal{F}uv)\}.$$

The bisimulation relation $\simeq$ replaces the homotopy path relation $\sim_p$ of Munkres. These are very weak forms of equivalence and have no overt relationship to the local Kripke frame relations at h and k unless subsets of these endo-relations were used as paths. The following Lemma will allow us to to use the work of Lemma 20:

Lemma 26. *Let $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$. $\breve{\mathcal{F}} \circ \mathcal{F}$ and $\mathcal{F} \circ \breve{\mathcal{F}}$ are fixed points of the converse operator, i.e.,*

$$(\breve{\mathcal{F}} \circ \mathcal{F})^{\smile} = \breve{\mathcal{F}} \circ \mathcal{F}, \quad \text{and} \quad (\mathcal{F} \circ \breve{\mathcal{F}})^{\smile} = \mathcal{F} \circ \breve{\mathcal{F}}$$

The next Lemma is similar to Lemma 20:

Lemma 27. *Let $\mathcal{F} : \mathsf{h} \rightharpoonup \mathsf{k}$. If*

$$\mathcal{R} \overset{def}{=} \bigcup_{i \geq 0} (\breve{\mathcal{F}} \circ \mathcal{F})^i \quad \text{and} \quad \mathcal{J}_{\mathcal{F}} \overset{def}{=} \{\langle x, x \rangle \mid x \in \text{dom}(\mathcal{F})\},$$

then

$$\breve{\mathcal{F}} \circ \mathcal{F} \overset{\mathcal{R}}{\simeq} \mathcal{J}_{\mathcal{F}};$$

if

$$\mathcal{S} \overset{def}{=} \bigcup_{i \geq 0} (\mathcal{F} \circ \breve{\mathcal{F}})^i \quad \text{and} \quad \mathcal{J}_{\breve{\mathcal{F}}} \overset{def}{=} \{\langle y, y \rangle \mid y \in \text{dom}(\breve{\mathcal{F}})\},$$

then

$$\mathcal{J}_{\mathcal{G}} \overset{\mathcal{S}}{\simeq} \mathcal{F} \circ \breve{\mathcal{F}}.$$

In light of Lemma 26, we could have used $\breve{\mathcal{R}}$ and $\breve{\mathcal{S}}$ respectively as in Lemma 20. The proof of this Lemma is similar to the proof of Lemma 20. All that is required is to note that $\mathcal{J}_{\mathcal{F}} \subseteq (\breve{\mathcal{F}} \circ \mathcal{F}) \subseteq \mathcal{R}$ since $\mathcal{J}_{\mathcal{F}} \subseteq \mathcal{I}_{\mathsf{h}}$ and $\mathcal{I}_{\mathsf{h}} \subseteq \mathcal{R}$. Similarly $\mathcal{J}_{\mathcal{G}} \subseteq (\mathcal{F} \circ \breve{\mathcal{F}}) \subseteq \mathcal{S}$. Hence the use of these as extra conjuncts in the premises of the Lemma are unnecessary.

From Lemma 27 we have

Theorem 28. *Let $\mathcal{F} : \mathcal{H} \rightharpoonup \mathcal{K}$, then*

$$[\breve{\mathcal{F}}] \star [\mathcal{F}] = [\mathcal{J}_{\mathcal{F}}], \quad \text{and} \quad [\mathcal{J}_{\mathcal{F}}] = [\mathcal{F}] \star [\breve{\mathcal{F}}].$$

Theorem 29. *The formulas*

$$\langle r \rangle [j_f] P \overset{\mathsf{h}}{\supset} [f \circ \breve{f}] \langle r \rangle P, \quad \text{and} \quad \langle r \rangle [f \circ \breve{f}] P \overset{\mathsf{h}}{\supset} [j_f] \langle r \rangle P.$$

and

$$\langle s \rangle [j_{\breve{f}}] Q \overset{\mathsf{k}}{\supset} [\breve{f} \circ f] \langle s \rangle Q, \quad \text{and} \quad \langle s \rangle [\breve{f} \circ f] Q \overset{\mathsf{k}}{\supset} [j_{\breve{f}}] \langle s \rangle Q.$$

are valid.

Lemma 27 gives the conditions validating the formulas and $\mathcal{R}$ and $\mathcal{S}$ can be defined using $\mathcal{F}$ and $\breve{\mathcal{F}}$. $\mathcal{J}_{\mathcal{F}}$ and $\mathcal{J}_{\breve{\mathcal{F}}}$ can be defined in the local frames at h and k.

6.6. Specifying the Identity Elements of the Groupoid

The set $\mathrm{dom}(\mathcal{F})$ can be modally defined using a special proposition, we will call it $\mathbb{D}_f$. The modal formula is

$$[f][\breve{f}]\mathbb{D}_f \overset{\mathsf{h}}{\supset} [j_f]\mathbb{D}_f.$$

This can also be defined a bit cleaner:

$$[f][\cdot f\cdot]\mathbb{D}_f \overset{\mathsf{h}}{\supset} [j_f]\mathbb{D}_f.$$

The frame condition is

$$\mathcal{J}_{\mathcal{F}} = \{\langle x, x\rangle \mid x \in \mathrm{dom}(\mathcal{F})\}.$$

Lemma 30. *The formula* $[f][\cdot f\cdot]\mathbb{D}_f \overset{\mathsf{h}}{\supset} [j_f]\mathbb{D}_f$ *is valid in any frame satisfying* $\mathcal{J}_{\mathcal{F}} = \{\langle x, x\rangle \mid x \in \mathrm{dom}(\mathcal{F})\}$.

PROOF. Under interpretation, $[f][\cdot f\cdot]\mathbb{D}_f \subseteq [j_f]\mathbb{D}_f$. Assume $x \in_{\mathsf{h}} [f][\cdot f\cdot]\mathbb{D}_f$. Let $\mathcal{J}_{\mathcal{F}}xu$, then $x \in_{\mathsf{h}} \mathrm{dom}(\mathcal{F})$ and $\mathcal{F}xv$ for some v. From the valuations conditions of $[f]$ and $[\cdot f\cdot]$, then $v \in [\cdot f\cdot]\mathbb{D}_f$ and $x \in_{\mathsf{h}} \mathbb{D}_f$. From the definition of $\mathcal{J}_{\mathcal{F}}$, $x = u$ and hence $u \in_{\mathsf{h}} \mathbb{D}_f$. By definition, $x \in_{\mathsf{h}} [j_f]\mathbb{D}_f$.

Under interpretation

Lemma 31.

$$\langle j_f\rangle\mathbb{D}_f = \mathbb{D}_f.$$

PROOF. Let $x \in \langle j_f\rangle\mathbb{D}_f$, then for some x', we have $\mathcal{J}_{\mathcal{F}}xx'$ and $x' \in \mathbb{D}_f$. However, $\mathcal{J}_{\mathcal{F}}xx'$ means that $x \in \mathrm{dom}\,\mathcal{F}$. Since, $x = x'$, so $x \in \mathbb{D}_f$ and $\mathrm{dom}(\mathcal{F}) \subseteq \mathbb{D}_f$. Next, let $x \in \mathbb{D}_f$, then $x \in \mathrm{dom}(\mathcal{F})$ and hence $\mathcal{J}_f xx$ holds. Therefore, $x \in \langle j_f\rangle\mathbb{D}_f$.

Hence in the axiom set, we require

$$\langle j_f\rangle\mathbb{D}_f \overset{\mathsf{h}}{\equiv} \mathbb{D}_f.$$

where $\mathbb{D}_f$ is a special constant proposition.

Note that by choosing smaller relations, either endo-relations or distributed relations, there multiple sets as endpoints that can be chosen. That is, choose subsets of the domains of $\mathcal{F}$ and $\breve{\mathcal{F}}$ (or $\mathcal{H}$ and $\breve{\mathcal{H}}$ if only endo-relations are used). To specify this in the logic, use the axioms

$$\mathbb{D}_{f'} \overset{\mathsf{h}}{\supset} \mathbb{D}_f, \quad \mathbb{D}_{\breve{f}} \overset{\mathsf{k}}{\supset} \mathbb{D}_{\breve{f}},$$

and

$$[f]\mathbb{D}_f \overset{\mathsf{h}}{\supset} [f']\mathbb{D}_{f'}, \quad [\cdot f\cdot]\mathbb{D}_{\breve{f}} \overset{\mathsf{k}}{\supset} [\cdot f'\cdot]\mathbb{D}_{\breve{f}'}.$$

7. Applications

For the applications, we do not follow table of conventions Table 1.

7.1. Goguen-Meseguer (GM) Noninterference

Many security arguments such as GM work by extracting information about a system and comparing it with a perfectly functioning ideal system. The extraction is not necessarily easy to perform operationally yet it does serve to highlight the important features. The extraction and comparing frequently can be seen as a classic modal logic simulation although the original work does need to be reinterpreted. We use distributed logic to express the reinterpretation of the argument of GM. This does not make the argument shorter, rather it puts it into a common framework so that other security properties can be compared. Incidentally, were GM to outfit their next state function to a next state relation (for non-determinism), the analysis below would not change.

We first make some simplifying assumptions that do not detract from the core argument. GM divides their users into two groups, we can consider two exemplars of each group and call them L and H. We assume there are two machines M_L and M_{L+H}. The first runs L commands and the second runs an interleaving of commands from L and H. M_L has access to L memory and M_{L+H} has access to both L and H memory. A state of M_L is just the contents of L's memory whereas a state of M_{L+H} is the contents of both L's and H's memory. Each state is merely a snap shot at some time in the course of the machines' executing commands.

L can tell if H is running when certain sequences of commands of M_{L+H}, all starting from an initial state and stripped of the high commands fails to match M_L which runs without H running. The key is the term "match". GM uses "read" commands for determining this match. Read commands result in some output.

GM non-interference is defined using L and H commands. The commands are all assumed to start from a single initial state. Rather than use read commands, we will abstract the result of those into merely the state. This too is immaterial to the argument.

We define three localities. l (lower case L in sans serif) for M_L in isolation from H and h for M_{L+H}. The last locality k is for M_{L-H} is a bit delicate. It stands for M_{L+H} minus the effect of H. To state this effect, threads are used.

We pair commands and states, (c, s) where c is a command and s is state of memory. An L state is the contents of L memory, and $L + H$ state is the contents of L and H memory. Threads in M_L and M_{L+H} are sequences of of these tuples

$$(c_1, s_1) \dots (c_i, s_i) \dots (c_n, s_n) \quad \text{such that } c_i(s_{i-1}) = s_i, i > 0,$$

where s_0 is an initial state and juxtaposition represents concatenation. All threads in M_{L+H} start from the same initial state in M_{L+H}; stripping this state of H memory gives us the initial state of M_L. In effect, we have embedded the next state function of GM into the threads.

Now we describe the locality k for M_{L-H}. Threads in M_{L-H} are derived from the threads of M_{L+H} by stripping out tuples (c, s) with c an H command. Any remaining (c, s) tuples will have their H part of the state s stripped out. Let Π be a "stripper" or projection function which takes threads from M_{L+H} to threads in M_{L-H}. If there is no interference, then $c_j(s_{j-1}) = s_j$ for c_j in all threads of M_{L-H}.

If there is interference, for at least one (c_i, s_i) of one thread of M_{L-H}, $c_i(s_{i-1}) \neq s_i$. The way to see this is assume there is a sequence $x_{i-1}x_ix_{i+1}$ in one thread where the x are tuples $(c.s)$ and c_{i-1}, c_{i+1} are L commands and c_i is an H command. Assume further that c_i changes some L memory that c_{i+1} uses. With (c_i, s_i) stripped out, c_{i+1} applied to s_{i-1} (and not s_i) computes some state s. If s is different than s_{i+1}, then interference has occurred.

There is one bisimulation which simply records the relationship between M_{L-H} and M_{L+H}. A second bisimulation between M_L and M_{L-H} occurs if there is no interference. We attend to the first bisimulation. Let the variables u, v, etc. refer to threads. Let $\mathit{LastSt}(w(c, s)) = s$ (for any thread w) where $\mathit{LastSt}(w) = s_0$ if w is the empty thread. The "next state" relation, more accurately "next thread" relation on h (recall this includes commands and memory both from L and H) is

$$\mathcal{H}vv' \text{ iff } v' = v(c', s') \text{ and } c'(\mathit{LastSt}(v)) = s'.$$

The next thread relation on k (for M_{L-H}) builds the result of the projection function Π from threads of h to threads of k:

$$\mathcal{K}uu' \text{ iff } u' = z(c, s) \text{ and } u = z, \quad \text{for some } (c, s).$$

We let $\mathcal{S}$ from threads of M_{L-H} to threads of M_{L+H} be the inverse of the projection function, Π. The relation $\breve{S}$ is the function Π in relational form. In order to state the simulation conditions, it is necessary to use the reflexive-transitive closures of $\mathcal{K}$ and $\mathcal{H}$. The two simulation relations are:

$$\mathcal{S}uv \text{ and } \mathcal{K}^*uu' \text{ implies } \exists v'(\mathcal{H}^*vv' \text{ and } \mathcal{S}u'v'),$$

and

$$\breve{\mathcal{S}}vu \text{ and } \mathcal{H}^*vv' \text{ implies } \exists u'(\mathcal{K}^*uu' \text{ and } \breve{\mathcal{S}}v'u'),$$

thus $\mathcal{S}$ forms a bisimulation. Considering the second condition, the reason for the reflexive-transitive closure of $\mathcal{K}$ and $\mathcal{H}$ is because $\mathcal{H}vv'$ does not imply $\mathcal{K}(\Pi v)(\Pi v')$ since it might be that $\Pi v = \Pi v'$ and $\mathcal{K}uu$ for any u does not obtain. Hence Π forces $\mathcal{K}$ to be reflexive. However, reflexivity of $\mathcal{K}$ over all sequences of k forces $\mathcal{H}$ to be reflexive as well. Since it is unpredictable just how many tuples may be removed by Π before generating a different thread from a preceding thread, so the transitive closure on $\mathcal{H}$ is needed. Transitivity of $\mathcal{H}$ forces $\mathcal{K}$ to be transitive as well. Hence rather than build reflexive-transitive closure into $\mathcal{K}$ and $\mathcal{H}$, we apply the reflexive-transitive closure to those relations and indicate it by the Kleene *.

These two simulations underwrite the distributed logic formulas

$$\langle s\rangle[h^*]V \overset{\mathsf{k}}{\supset} [k^*]\langle s\rangle V, \quad \langle\!\cdot s\cdot\!\rangle[k^*]U \overset{\mathsf{h}}{\supset} [h^*]\langle\!\cdot s\cdot\!\rangle U$$

respectively. It helps to determine precisely what are $\langle s\rangle$ and $\langle\!\cdot s\cdot\!\rangle$ in terms of UCLA propositions (sets of threads), and under interpretation, we will use

$$\langle s\rangle[h^*]V \subseteq_{\mathsf{k}} [k^*]\langle s\rangle V, \quad \langle\!\cdot s\cdot\!\rangle[k^*]U \subseteq_{\mathsf{h}} [h^*]\langle\!\cdot s\cdot\!\rangle U.$$

letting U and V be UCLA propositions in k and h respectively,

$$\begin{aligned} u \in_{\mathsf{k}} \langle s\rangle V \text{ iff } & \exists v(\mathcal{S}uv \text{ and } v \in_{\mathsf{h}} V) \\ \text{iff } & \exists v(\breve{\Pi}uv \text{ and } v \in_{\mathsf{h}} V) \\ \text{iff } & \exists v(\breve{\breve{\Pi}}\,uv \text{ and } v \in_{\mathsf{h}} V) \\ \text{iff } & \exists v(\Pi vu \text{ and } v \in_{\mathsf{h}} V) \\ \text{iff } & \exists v(\Pi v = u \text{ and } v \in_{\mathsf{h}} V) \end{aligned}$$

Letting Π be extended to a forward image function on UCLA propositions, $\langle s\rangle V \subseteq_{\mathsf{k}} \Pi V$. Since Π is a function, then $\forall v \in_{\mathsf{h}} V$, there is some u such that $\Pi v = u$. By definition $\Pi V \subseteq_{\mathsf{k}} \langle s\rangle V$. Hence $\langle s\rangle V \overset{\mathsf{k}}{=} \Pi V$. Similarly, it is easy to determine that $\langle\!\cdot s\cdot\!\rangle U \overset{\mathsf{h}}{=} \Pi^{-1}U$.

Rewriting our formulas

$$\Pi[h^*]V \subseteq_{\mathsf{k}} [k^*]\Pi V, \quad \Pi^{-1}[k^*]U \subseteq_{\mathsf{k}} [h^*]\Pi^{-1}U.$$

This bisimulation cannot be broken in the sense that it merely expresses the situation that is posited to exist between M_{L-H} and M_{L+H}. The second bisimulation is between M_L and M_{L-H} in the case where there is no interference. There is then a bisimulation using the identity function id. Here the reflexive-transitive closures are not necessary:

$$\mathrm{id}\,xy \text{ and } \mathcal{L}xx' \text{ implies } \exists y'(\mathcal{K}yy' \text{ and } \mathrm{id}\,x'y')$$

and

$$\mathrm{id}^{\smile}\,xy \text{ and } \mathcal{K}xx' \text{ implies } \exists y'(\mathcal{L}yy' \text{ and } \mathrm{id}^{\smile}\,x'y')$$

validating

$$\langle 1\rangle[k]Q \subseteq_{\mathsf{l}} [l]\langle 1\rangle Q$$

and

$$\langle \breve{1}\rangle[l]P \subseteq_{\mathsf{k}} [k]\langle \breve{1}\rangle P.$$

Consequently, L cannot tell whether H is executing since L can detect no difference between running in isolation from H and running with H.

Now we analyze what happens when this latter bisimulation is broken. Let $\mathcal{T} : \mathsf{k} \rightharpoonup \mathsf{l}$ be defined with

$$\mathcal{T} \overset{\mathit{def}}{=} \{\langle z(c_n, s'_n), z(c_n, s_n)\rangle \mid c_n(s_{n-1}) \neq s'_n\}.$$

Since $c_n(s_{n-1}) = s_n$, this becomes

$$\mathcal{T} \overset{def}{=} \{\langle z(c_n, s'_n), z(c_n, s_n)\rangle \mid s_n \neq s'_n\},$$

where z is a prefix of tuples common between the two threads. Hence $\mathcal{T}$ is a relation from k to l whose pairs are such that L running in isolation is not arriving at the same state as L while H is running.

The formulas $\langle t\rangle P$ for $P \in \mathsf{l}$ represent the propositions of M_L that supply possible "witnesses" of detection for H running. Let Cx be the sequence of operations derived from x by deleting the state components of the tuples in x. Extending $\mathcal{T}$ to $\mathcal{T}'$ with

$$\mathcal{T}' = \{\langle x', x\rangle \mid \exists z, z'(z \leq x \text{ and } z' \leq x' \text{ and } \langle z, z'\rangle \in \mathcal{T} \text{ and } Cx = Cx')\}$$

for $\leq$ being the prefix relation, will cause $\mathcal{T}'$ to be a simulation relation from k to l and represents persistence of the anomalies between running without H and running with H. This builds reflexive-transitive closure into $\mathcal{T}'$ along $\mathcal{K}$ on the left hand members (of $\mathcal{T}$) and along $\mathcal{L}$ on the right hand members. Another way of achieving this is to define reflexive-transitive closure directly. Define the persistent anomalous propositions as those of the form

$$\langle k^*\rangle\langle t\rangle P$$

Hence

$$v \in_{\mathsf{k}} \langle k^*\rangle\langle t\rangle P \text{ iff } \exists z_1, \ldots, z_n, x(\mathcal{K}vz_n \text{ and } \mathcal{K}z_nz_{n-1}\cdots\mathcal{K}z_2z_1 \text{ and } \mathcal{T}'z_1x \text{ and } x \in_{\mathsf{h}} P).$$

7.2. Call Graph Monitoring

Each node in a call graph indicates a collection of program states that can occur when that node is reached during execution. As a collection of states, the node can be represented by a UCLA proposition. We will use the propositions Q_i refer to a call graph's states, see Figure 1. The next state relation of the program describes in microscopic scale the execution of the call graph. That is, there may be several collections of states that satisfy the call graph. The next state relation in the monitor describes the monitor's tracking of the program's execution.

There is a co-occurrence relation $\mathcal{F}$ from states of the monitor to states of the call graph and this relation is a simulation relation. Let h refer to the monitor locality and k the call graph locality. The Q_i below are nodes of the target's call graph. The formula

$$\langle f\rangle[k]Q_i \overset{\mathsf{h}}{\supset} [h]\langle f\rangle Q_i, \quad 1 \leq i \leq 6$$

represents the regularity that describes proper execution of the program under the watchful gaze of the monitor. The information flow is from the program to the monitor. The simulation states that the monitor's movement through its state space is matched by the movement of the program through its state space. The term $[h]\langle f\rangle Q_i$ corresponds to the monitor's precondition of $\langle f\rangle Q_i$ while this latter is a representation of a target's call graph node Q_i. The term $\langle f\rangle[k]Q_i$ is reported precondition $[k]$ of the target's node Q_i.

However, there is also a simulation in the other direction. This is the converse of the co-occurrence relation. This relation says that every move in the program through its state space must be matched by a movement of the monitor through its state space. The information flow in this case is from the monitor to the call graph.

Were a fault to occur in either the monitor or the call graph, the bisimulation relation would be out of step. We can view this as a debugging situation. Using the same devices as were used to diagnose Goguen-Meseguer failures, there are two more simulations that can be set up to describe the fault and the information that flows. That information is information that a debugger uses.

7.3. Path Bisimulations and Holes

Traditional homotopy is able to work with holes in the space. The problem is that Stone spaces, due to their topology, do not have holes. However, we can cut holes in a Stone space by using propositions and then state conditions for bisimulations to avoid the holes. Hence the points of a hole are still present but avoidable. We use a single locality h. The intuitive diagrams of the bisimulation is

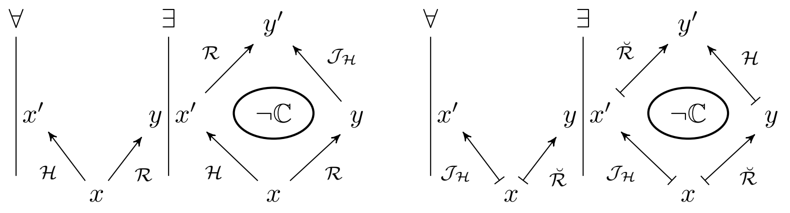

Figure 13: Bisimulation Avoiding a Hole

The logic must be outfitted with a new constant for every hole we wish to consider. We use a single example hole denoted $\neg\mathbb{C}$. We will interpret $\mathcal{H}$ avoiding the hole $\neg\mathbb{C}$ as $\mathrm{dom}(\mathcal{H}) \subseteq \mathbb{C}$ and $\mathrm{dom}(\breve{\mathcal{H}}) \subseteq \mathbb{C}$. The domain of $\mathcal{H}$ is captured in the constant $\mathbb{D}_h$. The three axioms, now thought of as application induced, are

$$\mathbb{D}_h \supset \mathbb{C}, \quad \text{and} \quad \mathbb{C} \supset [h]\mathbb{C} \quad \text{and} \quad \mathbb{C} \supset [r]\mathbb{C}$$

expresses (in order)

- the domain of $\mathcal{H}$ avoids the hole $\neg\mathbb{C}$,
- $\mathcal{H}$, started in $\mathbb{C}$, never wanders into $\neg\mathbb{C}$, and
- $\mathcal{R}$, started in $\mathbb{C}$, never wanders into $\neg\mathbb{C}$.

Taking the last as an example, observe the validation conditions for $\mathbb{C} \supset [r]\mathbb{C}$. Let $x \in \mathbb{C}$, then because of the classical logic base, $x \notin \neg\mathbb{C}$. Assume $\mathcal{R}xy$. Since $\mathbb{C} \supset [r]\mathbb{C}$, then $x \in [r]\mathbb{C}$. Hence, from the valuation conditions for $[r]\mathbb{C}$, one has $y \in \mathbb{C}$.

The axioms and simulation conditions guarantee that the elements of the diagrams avoid the hole when $x \in \mathbb{D}_h$. To take account of the hole, we can use the forward and converse rules (respectively)

$$\frac{\mathbb{D}_h \xrightarrow{[jr]} \mathbb{C} \qquad r : \mathsf{h} \rightharpoonup \mathsf{h}}{\langle r \rangle \mathbb{D}_h \xrightarrow{[h]} \langle r \rangle \mathbb{C}} \qquad \frac{\mathbb{D}_h \xrightarrow{[h]} \mathbb{C} \qquad r : \mathsf{h} \rightharpoonup \mathsf{h}}{\langle r \rangle \mathbb{D}_h \xrightarrow{[jr]} \langle r \rangle \mathbb{C}}$$

Hence for any $x \in \mathbb{C}$, either $x \in \mathbb{D}_h$ or $x \notin \mathbb{D}_h$. In the first case, the axioms and rules guarantee that $\mathcal{H}$ will never wonder into the hole when started at x. In the second case, for no x' is it the case that $\mathcal{H}xx'$. Hence $\mathcal{H}$ avoids the hole.

7.4. Bisimilar Distributed Relations and Simulation Equivalence

As a use of bisimilar distributed relations, consider two components in an FPGA application with one a master and one a slave where the master controls the slave. Assume the master has two modes of operation. Control of the slave generates a control relation from the master to the slave. The fact that the master has two modes splits that control relation into two control relations. The security question is "can the slave determine what mode the master is in?". If the two control relations are bisimilar from the point of view of the slave, then the slave cannot tell which mode the master is using.

As a use of simulation equivalence, consider two components that may or may not perform in the same manner. One way to get at this question is to use two simulation relations and then determine whether the two simulation relations constitute a weak equivalence. If they do, then the two components may be swapped for each depending upon other considerations, e.g., energy footprint, chip real estate, etc.

8. Conclusion

The goal of this paper was to use Distributed Logic to express axioms and rules that govern model theoretic conditions of (bi)simulations and to show how bisimulation can be used in a version of homotopy type theory. We stayed closer to homotopy theory than homotopy type theory where the latter is now focused on an intensional theory such as proof theory. This paper is closer to extensional type theory.

Our language ReWire can be used to specify distributed systems such as Systems-on-a-Chip (SoCs) and can be used to specify Field Programmable Gate Arrays (FPGAs) versions of SoCs. ReWire can be seen as an intensional take on the modeling apparatus of Kripke structures. In particular, a state machine written in ReWire is an intensional version of a Kripke frame. To unwind the control in the state machine is to generate the Kripke model. The states themselves are functions accepting inputs and generating outputs and a next state. Hence the states and the local Kripke relations are specified together. The distributed Kripke relations are generated by the sending and receiving of signals among components in the FPGA application that implements the SoC specification.

In any specific application, we are not interested in a single local next state relation for a component but rather need subsets of this relation. Given a single Kripke relation, we can choose many subrelations to capture different aspects of the situation we are modeling with the frame. We may wish to treat a subcollection of states as something like a submachine of the larger component holding the local Kripke relation. Restricting that relation to the subcollection allows us this flexibility.

The entire collection of subrelations is a complete lattice, but typically all the relations of the lattice are not useful. One only needs a partial order. A logic incorporating this feature is partially-ordered modalities [27] where the main axioms are of the form

$$[{\scriptstyle h'}]P \supset [{\scriptstyle h}]P \text{ for } h' \geq h \text{ and } h', h \in V.$$

where V is a partial order and in any interpretation, $\mathcal{H} \subseteq \mathcal{H}'$. The partial order V comes from the domain of application. Adding a partial order structure to the endo-relations allows us to model submachines at a locality. This is an important concept in modeling the design of SoCs. We can also add a partial order on the distributed relations. This is also useful for modeling the design SoCs.

The endo-relation $\mathcal{H}$ at h gives the connection relation on the points of the Stone space at h, assuming one does not add a second endo-relation at h. A second endo-relation, with no subset condition between the two relations, would essentially mean two connection structures at a locality and we leave an exploration of this idea in a future paper other than remark it may become useful in modeling parallel machines with a single locality rather than use multiple localities.

9. Bibliography

References

[1] J. Barwise, J. Seligman, Information Flow: The Logic of Distributed Systems, CUP, 1997, Cambridge Tracts in Theoretical Computer Science 44 (1997).

[2] V. R. Pratt, Chu spaces from a representational viewpoint, Annals of Pure and Applied Logic 96 (1999) 319–333 (1999).

[3] J. A. Goguen, R. M. Burstall, Institutions: Abstract model theory for specification and programming, CSLI Research Reports 85-30 (1985) 1–73 (1985).

[4] G. Allwein, W. L. Harrison, Distributed modal logic, in: K. Bimbó (Ed.), J. Michael Dunn on Information Based Logic: Outstanding Contributions to Logic, Springer-Verlag, 2016, pp. 331–362 (2016).

[5] G. Allwein, W. L. Harrison, T. Reynolds, Distributed relation logic, Logic and Logical Philosophy 26 (1) (2017) 19–61 (2017).

[6] P. Blackburn, M. de Rijke, Y. Venema, Modal Logic, Cambridge University Press, 2001, Cambridge Tracts in Theoretical Computer Science, No. 53 (2001).

[7] J. Rutten, A calculus of transition systems (toward universal coalgebra), in: A. Ponse, M. de Rijke, Y. V. (eds.) (Eds.), Modal Logic and Process Algebra, a bisimulation perspective, CSLI Lecture Notes No. 53, Center for the Study of Language and Information, 1995, pp. 231–256 (1995).

[8] D. Sangiorgi, On the origins of bisimulation and coinduction, ACM Trans. Program. Lang. Syst. 31 (4) (2009) 15:1–15:41 (May 2009).

[9] J. A. Goguen, J. Meseguer, Security policies and security models, in: Proceedings of the 1982 IEEE Symposium on Security and Privacy, IEEE Press, 1982, pp. 11–20 (1982).

[10] M. Abadi, M. Budiu, U. Erlingsson, J. Ligatti, Control-flow integrity, in: Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS '05, ACM, New York, NY, USA, 2005, pp. 340–353 (2005).

[11] M. Abadi, M. Budiu, U. Erlingsson, J. Ligatti, Control-flow integrity principles, implementations, and applications, ACM Trans. Inf. Syst. Secur. 13 (1) (2009) 4:1–4:40 (Nov. 2009).

[12] N. Christoulakis, G. Christou, E. Athanasopoulos, S. Ioannidis, HCFI: Hardware-enforced control-flow integrity, in: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, CODASPY '16, ACM, New York, NY, USA, 2016, pp. 38–49 (2016).

[13] R. de Clercq, I. Verbauwhede, A survey of hardware-based control flow integrity (CFI), CoRR abs/1706.07257 (2017).

[14] W. L. Harrison, G. Allwein, Semantics-directed prototyping of hardware runtime monitors, in: Proceedings of the 29th International Symposium on Rapid System Prototyping (RSP), 2018, pp. 42–48 (2018).

[15] W. L. Harrison, G. Allwein, Language abstractions for hardware-based control-flow integrity monitoring, in: Proceedings of the 2018 International Conference on Reconfigurable Computing and FPGAs, 2018, p. (to appear) (2018).

[16] W. L. Harrison, A. Procter, I. Graves, M. Becchi, G. Allwein, A programming model for reconfigurable computing based in functional concurrency, in: Proceedings of the 11th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC 2016), IEEE, 2016, pp. 1–8 (2016).

[17] G. Birkhoff, J. D. Lipson, Heterogeneous algebras, Journal of Computational Theory 8 (1970) 115–133 (1970).

[18] G. Allwein, W. L. Harrison, D. Andrews, Simulation logic, Logic and Logical Philosophy 23 (3) (2014) 277–299 (2014).

[19] J. M. Dunn, G. Hardegree, Algebraic Methods in Philosophical Logic, Oxford Logic Guides 41, Oxford University Press, 2001 (2001).

[20] C. Kupke, A. Kurz, Y. Venema, Stone coalgebras, in: H. P. Gumm (Ed.), Coalgebraic Methods in Computer Science, Electronic Notes in Theoretical Computer Science, Vol. 82 of 1, 2003, pp. 170–190 (2003).

[21] P. J. Freyd, A. Scedrov, Categories, Allegories, North–Holland, 1990 (1990).

[22] D. Barker-Plummer, J. Barwise, J. Etchemendy, Language, Proof, and Logic, 2nd Edition, CSLI Publications, 2011 (2011).

[23] V. Gordanko, S. Passy, Using the universal modality: Gains and questions, Journal of Logic and Computation 2 (1992) 5–30 (1992).

[24] J. V. Benthem, A new modal Lindström theorem, Algebra Universalis 1 (1) (2007) 125–138 (2007).

[25] J. R. Munkres, Topology: A First Course, Prentice–Hall, New Jersey, 1975 (1975).

[26] R. Brown, Topology and Groupoids, BookSurge Publishing, 2006 (2006).

[27] G. Allwein, W. L. Harrison, Partially ordered modalities, in: Proceedings of the Advances in Modal Logic Conference, 2010, Springer-Verlag, 2010, pp. 1–20 (2010).