Semantics of Nondeterministic Construction

A thesis presented for the degree of
Master of Science
in
Logic, Computation, and Methodology

Jacob Neumann

Department of Philosophy
Carnegie Mellon University
August 2020

Abstract

Propositional Dynamic Logic (PDL) provides a formal language for the logic of (non)deterministic programs, and axiomatizations of PDL with respect to a standard class of relational models have been known for several decades. However, it has recently been shown in [5] that a basic fragment of PDL can also be modeled by a richer class of models: dynamic topological models. In addition to interpreting the language of PDL, dynamic topological models give semantics for modalities which admit an *epistemic interpretation*: we can read the formulas of this logic as formally expressing the dynamic-epistemic properties of agents performing actions in situations. We further this project by developing a theory of "program constructors", which provide a means to (deterministically or nondeterministically) combine simple actions into more elaborate ones. In addition to presenting a fascinating theory in its own right, the theory of program constructors allows us to formally model a wide array of phenomena within the dynamic topological framework, in particular nondeterministic binary choices (which we epistemically interpret as "coin-flipping"). In order to make sense of such objects, we undertake an extensive development of the mathematical theory of dynamic topological logic, elucidating a series of powerful notions which, in addition to allowing us to understand nondeterministic union/coin-flipping with precision, suggest a variety of fascinating further inquiries.

0 Introduction

In an unpublished note,[1] Immanuel Kant made the following claim: "The business of philosophers is not to give rules, but to analyze the covert judgments of common reason."[2] Setting aside the nebulousness of the phrase, "common reason", the underlying point is fairly straightforward: when going about their lives and interactions with their world, people seem to operate on some basic, pretheoretic conception of the way the world is and the ways it *could be*. This "conception" is not (and perhaps cannot be) made fully explicit, but nonetheless it seems to be the basis upon which each of us navigates our world and makes day-to-day choices. Only in select circumstances do we typically make use of explicit scientific reasoning to guide our actions – for the most part, we lean on this implicit, informal framework.[3] Therefore, the nature, structure, and function of this "conception" becomes a worthwhile topic of inquiry: if we understood with exactness the unspoken principles a person applies to make sense of (and make *use* of) her present situation, we would gain much better insight into the structure of her reality. Our purpose here will be to explicate the logical and mathematical structure of such a "conception" of the world (and its possibilities), understood from the standpoint of a given *agent*; thereby, we obtain a convincing (and concrete) mathematical model of the world that agent experiences. Studying the properties of this model will deliver to us a set of "judgments" – the principles *covertly* governing our agent's possibilities in the situation – *overtly* rendering them for precise analysis.

We'll begin by expounding a variety of mathematical logic suited to this task, which we'll call *Agent Dynamic Topological Logic* ("agent DTL"). Agent DTL consists of a formal language (called $\mathcal{L}_{\Box\bigcirc}$), a theory of models (called dynamic topological models, or DTMs), and a formal deductive calculus (called ADTL). The language will allow us to formally and exactly state propositions about the agent (and her abilities), the models will represent the mathematical structure of the agent's "conception" of the possibilities (and will give meaning to the $\mathcal{L}_{\Box\bigcirc}$ terms), and the deductive

[1] *Akademie* edition of Kant's Collected Works, Vol. 15, p. 180. This quote was likely written sometime around 1769-1771. An unpublished note from earlier in his career is obviously not an authoritative statement of Kant's philosophy, but I like the expression regardless.

[2] "Der philosophen Geschäfte ist nicht, Regeln zu geben, sondern die geheime Urtheile der gemeinen Vernunft zergliedern."

[3] Of course, it could well be argued that some features of scientific reasoning are inherent to this "informal framework" as well. That is certainly an interpretation which can be sustained throughout the present work – see the comments on "observation" in Sect. 1.

calculus will allow us to derive formal statements of the structural properties of these models – and thereby obtain formal statements of the structure of the agent's world.

Furthermore, this logic (introduced as "DTL" in [5]) represents the confluence of several threads in mathematical logic and theoretical computer science, and retains intimate connection to these disciplines. In particular, it combines dynamic topological logic[4] with propositional dynamic logic.[5] The former is itself a combination of two prominent topics in modal logic: topological semantics[6] and the logic of partial functions.[7] Each of these logics contributes a different aspect to agent DTL, and suggests further connections and research. We make an effort to indicate some of these connections and further applications of our method, though enumerating (let alone fully exploring) all of them is far beyond our scope.

We will focus our analysis on a certain activity peculiar to the human species: coin flipping. Though this is a rather simple and specific example, it will prove challenging to apprehend in its mathematical totality. Our reflections on the intended purpose of a "proper" coin flip will indicate for us the rough outline of how to proceed with such a mathematical analysis, which we can then make explicit. After giving an account of how the ability of the agent to flip a coin transforms the arrangement and structure of possible worlds, we give an axiomatization of how this transformation "works" in a simplified fragment of the agent DTL formal language. These axioms, when read as statements about agents interacting with coin-flipping situations, come out as plausible descriptions of the agent's abilities – vindicating our choice of interpretation.

This accomplished, we will turn our focus to fully articulating the logic of coin-flipping situations, using the agent DTL model theory. As mentioned, this is not easy. To achieve such a result, we will need to make a thorough dive into the theory of dynamic topological models.[8] In particular, it will be necessary to develop a notion intermediate to the traditional distinction in modal logic between *models* and *frames*, which we'll call *refined frames*.[9] Refined frames serve as the correct level of structural generality to capture the phenomenon of coin-flipping, a fact which we demonstrate by carrying out the appropriate analysis. We conclude by arguing that we have indeed obtained clarity on the covert judgments of common reason *about coin flips*, and speculating about further applications of our method.

[4] See e.g. [10], [4]

[5] [9] is the paper where this was introduced. The Stanford Encyclopedia of Philosophy provides a good introduction to PDL [16], whose notation we'll mimic.

[6] The classic paper on this is [11], but see also [2] or [3].

[7] The partial functions provide semantics for "Diamond-type" Kripke-style modalities (i.e. existentially quantifying over the 0 or 1 "accessible worlds"). We'll give both a philosophical interpretation of why these are interpreted as partial functions (rather than arbitrary relations), and see this requirement reflected in the deductive calculus.

[8] Which – even disregarding our philosophical motivation – proves interesting in its own right.

[9] Similar to (but less general than) the notion of a "general frame" – see [7, Section 5.5]

Contents

0.1 Mathematical Reference

This essay relies on a variety of mathematical notions, notations, and conventions. Most of what is needed will be explicitly introduced in the text, but we will assume some prior knowledge. The main ideas, terminology, and notation assumed is summarized below, for reference.

I assume familiarity with standard notions and techniques from mathematics, e.g. sets, binary relations, and functions. A **partial function** between sets X and Y, denoted $f : X \rightharpoonup Y$, consists of a relation $f \subseteq X \times Y$ where, for each $x \in X$, there is at most one $y \in Y$ such that $(x, y) \in f$. If there is such a y, we say f is *defined* at x and denote y by $f(x)$; otherwise we say "f is undefined at x" or "$f(x)$ is undefined". The set of those $x \in X$ such that f is defined is called the *domain* of f, whereas the set Y is called the *codomain*. We call f a **total function** (or just a "function") if its domain is all of X. Many of the usual notions defined for functions make sense for partial functions as well, such as injectivity and surjectivity.

Familiarity with classical propositional modal logic is assumed (the first several chapters of [7] provide more than adequate reference). I'll assume the reader knows and is comfortable with the syntax and relational ("Kripke-style") semantics for multi-modal propositional logic, the associated deductive calculus $\mathsf{(K)}$, and the meaning and demonstration of the soundness, (strong) completeness results, and the topics of definability and invariance. The distinction between *model-level* and *frame-level* formula satisfaction (and class definition) is key, as is the notion of a *bisimulation*. Much of this paper will be devoted to carrying out analogous development in a richer setting. In addition, familiarity with topological semantics for modal logic (see e.g. [2] or [3]) will prove helpful – see below for a note on the topological concepts we need. Finally, we'll frequently make reference to the idea – common to several branches of mathematical logic – of a *function symbol* with a specific *arity*. We don't systematically distinguish between function symbols which are written in prefix or infix position.

We make heavy use of binary relations. If $R \subseteq X \times Y$, we may indicate $(x, y) \in R$ by writing xRy or saying "x is related to y by R" or "x and y are R-related", as appropriate. Sometimes, we wish to think of binary relations $R \subseteq X \times Y$ as "multi-valued partial functions": for each $x \in X$, we write $R(x)$ to indicate the (possibly empty) set of all those $y \in Y$ such that xRy. When we want to emphasize this perspective, we might write $R : X \twoheadrightarrow Y$ to indicate $R \subseteq X \times Y$, especially if X and Y are understood to possess additional "structure" which R "respects".

- We'll say that R is **total** if $R(x)$ is nonempty for all $x \in X$.
- If $U \subseteq X$, then $R(U)$ denotes the set of those $y \in Y$ such that xRy for some $x \in U$. We call $R(U)$ the *image* of U under R.
- If $V \subseteq Y$, then $R^{-1}(V) = \{x \in X \ : \ xRy \text{ for some } y \in V\}$. We refer to this as the *preimage* of V under R.
- For $y \in Y$, $R^{-1}(y)$ is shorthand for the set $R^{-1}(\{y\})$. If $R^{-1}(y)$ is nonempty for all $y \in Y$, we say that R is **surjective**.

If $R \subseteq X \times X$ is an *equivalence relation* and $x \in X$, we'll write $[x]_R$ (or just $[x]$ if R is understood) instead of $R(x)$, and call it "the *R-equivalence class* of x". We write X/R for the *quotient of X by R*, the set of all R-equivalence classes, and write Q_R for the canonical projection function $X \to X/R$ sending x to $[x]_R$.

We also make use of notions and techniques from topology. [13] is a standard introduction to general topology. A **topology** τ on a set X is a collection of subsets of X called **open sets**, subject to the *axioms of topology*:

1. $X, \emptyset \in \tau$
2. If $U, V \in \tau$, then $U \cap V \in \tau$[10]
3. If I is an arbitrary set and $U_i \in \tau$ for all $i \in I$, then $\bigcup_{i \in I} U_i$ is also in τ.

The two simplest topologies on X are the *discrete topology* ($\tau = \mathcal{P}(X)$, i.e. every set is open) and the *indiscrete topology* ($\tau = \{\emptyset, X\}$). We'll indicate that τ is a topology on X by writing "(X, τ) is a topological space". Given such a topological space and some subset $A \subseteq X$, we write $\text{int}(A)$ (called the **interior** of A) to indicate the largest open set contained within A. An element x is in $\text{int}(A)$ iff there's some $U \in \tau$ such that $x \in U \subseteq A$. If $s : X \to Y$ (or $s : X \rightharpoonup Y$ or $s : X \nrightarrow Y$), we say that s is **continuous** with respect to topologies τ_X, τ_Y on X and Y (respectively) if, for all $V \in \tau_Y$, the preimage $s^{-1}(V)$ of V is in τ_X. Dually, we say that s is **open** if the image of any $U \in \tau_X$ is in τ_Y.

Topologies may be specified by *bases*. A **topological basis** (on a set X) is a collection $\mathcal{B} \subseteq \mathcal{P}(X)$ satisfying

1. For all $x \in X$, there's a $B \in \mathcal{B}$ such that $x \in B$.
2. If $B_0, B_1 \in \mathcal{B}$ and $x \in B_0 \cap B_1$, then there's a $B_2 \in \mathcal{B}$ such that

$$x \in B_2 \subseteq B_0 \cap B_1$$

Each basis $\mathcal{B}$ *generates* a topology $\tau(\mathcal{B})$. A set $U \subseteq X$ is open with respect to $\tau(\mathcal{B})$ (i.e. $U \in \tau(\mathcal{B})$) iff U can be written as

$$U = \bigcup_{i \in I} B_i$$

where each $B_i \in \mathcal{B}$. A simple proof shows that x is in the interior of $A \subseteq X$ with respect to $\tau(\mathcal{B})$ if and only if there's a $B \in \mathcal{B}$ with $x \in B \subseteq A$. Given topological spaces (X, τ_X) and (Y, τ_Y), we may define their **product topology** to be the topology on $X \times Y$ generated by the basis

$$\{U \times V \ : \ U \in \tau_X, V \in \tau_Y\}.$$

Note that the projection functions $\mathsf{pr}_1 : X \times Y \to X$ and $\mathsf{pr}_2 : X \times Y \to Y$ are continuous and open with respect to these topologies. Finally, given a topological space (X, τ_X) and an equivalence relation R on X, we define the **quotient topology** τ_X/R to be the topology on X/R given by

$$\tau_X/R = \{U \subseteq X/R \ : \ \{x \in X \ : \ [x] \in U\} \in \tau_X\}.$$

The quotient map Q_R is automatically continuous, and is open iff the following condition holds

$$\text{For all } U \in \tau_X, \ R(U) \in \tau_X.$$

Some of our central constructions will be achieved using *strings* and *streams*. Given a set Λ, write Λ^* for the set of all (finite-length) **Λ-strings**. All Λ-strings are either ϵ (the empty string) or $\lambda^\frown s$ for $\lambda \in \Lambda$ and $s \in \Lambda^*$. In the latter case, we call λ the *head* and s the *tail* of $\lambda^\frown s$.
A **Λ-stream** is an infinite-length Λ-string. Formally, we think of this as a function $\mathbb{N} \to \Lambda$, and use $\Lambda^\mathbb{N}$ to denote the set of all such streams. Similar to strings, we may extract the head and tail of a stream, and will write $S = \lambda^\frown S'$ to indicate that $S(0) = \lambda$ and $S(n) = S'(n-1)$ for all $n > 0$.

[10] By a simple inductive argument, this is equivalent to requiring that the intersection of finitely-many open sets is always guaranteed to be an open set.

Finally, we make occasional reference to *proper classes*, which are collections that are "too large to constitute a set". For instance, there cannot be a set of all topological spaces. The standard workaround is to speak of the *class* of topological spaces. The reader is encouraged to make sense of this according to their preferred foundational system. Though we remain intentionally nebulous on this point, our central notion (that of a *program constructor*) is indeed a *class function*, a function between proper classes. We also make use of this terminology to state soundness, completeness, and definability results: for instance, "Δ defines the class $\mathcal{K}$" just means that a structure (of the relevant kind) validates Δ if and only if it is an element of the class $\mathcal{K}$. $\mathcal{K}$ will generally be specified as those structures satisfying a given property, so class membership can be taken as a shorthand for satisfying the defining property.

0.2 Acknowledgement

It is no exaggeration to say that this project could not have existed without Adam Bjorndahl. This project developed in response to Adam's excellent work and insight, and his nurturing presence (and insightful comments and feedback) were vital to this thesis throughout its development. Thank you, Adam: it was a tremendous pleasure to work with you, and I hope this thesis serves as a worthwhile sample of the magnificent things you enable your students to achieve.

I am also greatly indebted to Kevin Kelly for his insights and guidance, and for continually inspiring and challenging me to think deeper. Kevin, I am quite lucky to have learned from you.

There were numerous people who contributed to this thesis, through insightful conversations, comments, and feedback, or whose presentations of various ideas (in some way) inspired what is written here. I'll undoubtedly fail to think of every such person, but the list includes at least: Steve Awodey, Patrick Blackburn, Steve Brookes, Clinton Conley, Simon Cullen, Ariel Davis, David Danks, Michael Erdmann, Jonas Frey, Florian Frick, Vipul Goyal, Harrison Grodin, Chris Grossack, Thejas Kadur, Dilsun Kaynar, Nathan King, Conny Knieling, Mara Harrell, Fernando Larrain, Tessa Murthy, Will Nalls, Paula Neeley, Aybüke Özgün, Frank Pfenning, Egbert Rijke, Vijay Ramamurthy, Wilfried Sieg, Elizabeth Viera-Patron, Dustin Updyke, Jeanne VanBriesen, Cameron Wong, Joseph Zielinski, Kevin Zollman, Colin Zwanziger, my classmates in the Spring 2018 Formal Epistemology Seminar, and my many students (especially those in the Spring 2020 'Modal Logic' course, and those in the Spring or Fall 2019 iterations of 'Hype for Types'). I am also incredibly thankful for my family and friends, who were with me and supported me throughout the writing of this thesis. Thank you!

1 Review: The Epistemic Interpretation

1.1 Agents and Situations

We begin by describing in more detail the phenomena we wish to formalize.

Agents find themselves in situations. In a situation, the agent perceives some features and properties of the situation, and perhaps utilizes the insights from these observations to guide their actions in the situation. By acting, the agent moves *through* the situation, and perhaps changes the truth of various propositions. For instance, if Alice is at a casino playing blackjack, then we would say that Alice found herself *in* the situation of "playing blackjack at the casino". The "situation" itself consists in the tables, the chairs, the cards, the chips, the people, etc. that Alice really interacts with,[11] but we are interested in how these things, in concert, create a dynamic-epistemic structure which Alice, so to speak, *navigates*.

Part of what makes the situation "playing blackjack" what it is, is the structure of what Alice is able to *do* in that situation: say Alice is dealt a 9 of spades and a 7 of clubs. At some point of the game (when it is her turn), Alice is required[12] to perform one of several available actions (in blackjack, the allowed actions are mostly hand gestures telling the dealer what to do), and performance of these actions results in different outcomes. For instance: tapping the table when it's her turn (known as "hitting") will make the dealer deal Alice another card, which could result in her losing the game (e.g. if the next card she gets is another 7, then, per the rules, she loses), or ending up in a more advantageous position (e.g. if the card she gets is a 3, then she's still in the game and is better-off). This is the "dynamic structure": as Alice goes through this situation, there's some fact of the matter about what's true at the present moment, but also about what will be true upon the execution of various actions. "Tap the table" is an action which is available to Alice at certain points of the situation, and by performing it, Alice may take her from one point of the situation (the beginning of her turn, at which point she is still in the game) to another (perhaps a state where she is no longer in the game). Throughout, we use "point", "state", or even "world" to refer to these moments within the situation, which the agent moves between by her actions.[13]

From a birds-eye view, we think of this dynamic structure of the situation as a collection of different *possible*[14] game states (different points the blackjack game *could* be at) *connected* by actions: if the agent executes a certain action (call the action π)[15] in the current point of the situation, then, by the time she's done with π, she'll find herself in a (perhaps) different point.[16] See Figure 1.1 for a visual depiction of such a setup. Part of what defines a situation is this internal structure: to *truly* specify how a situation (like playing blackjack) operates, we must provide indication of how the agent can move through that situation by executing various actions, and how executing those actions alters the truth of various propositions (like whether Alice is still in the game).

The situation must also encode what kinds of things Alice is able to *learn* about the circumstances she's in: part of what will inform her choice of what to do on her turn is what she can observe about her situation. For instance, when playing blackjack, the player's cards are dealt face-up (i.e. visible to all), but only one of the dealer's cards is dealt face-up. A good blackjack

[11] We need not make any hard commitments as to the ontological status of "situations". A reader skeptical of such entities may regard this merely as a manner of speaking about real-world phenomena.

[12] The actual details of how blackjack is played aren't important.

[13] We assume that these states are specified in a contextually-appropriate way.

[14] E.g. including all possible ways the cards could be dealt, and all possible points of such games

[15] We'll use $\pi, \sigma, \pi_0, \pi_1$, etc. to denote available actions

[16] We say "perhaps" because maybe we'll also include actions which don't change the state of the game at all. For instance, if Alice executes the action "scratch her head", then this has no meaning in the game of blackjack and therefore doesn't change the game state like "tap the table" does.

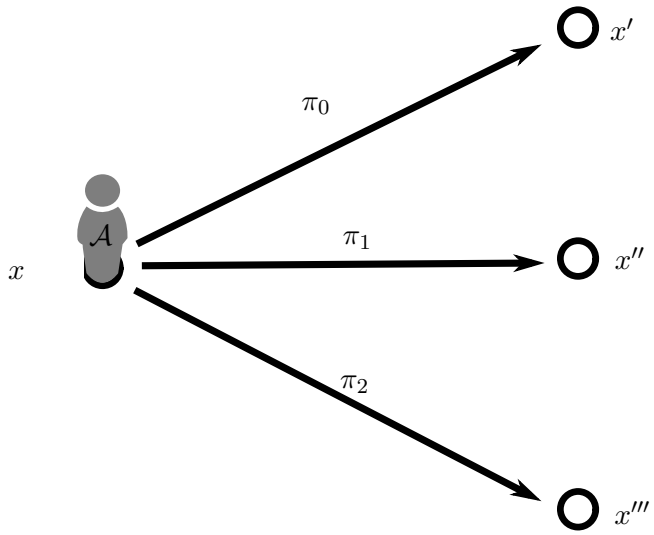


Figure 1.1: A situation S
If the agent $\mathcal{A}$ is at some point x and performs action π_0 (resp. π_1, π_2), she'll find herself at x' (resp. x'', x''')

player will use the information given to them (the cards they can see) to guide their strategy. So Alice is able to perceive things (by making appropriate observations, e.g. looking at the dealer's face-up card) and thereby come to know true propositions about the current point of the situation (for instance, that the dealer's face-up card is a 4). But Alice is *not* able to know (at the current point of the situation) that the dealer's other card is, say, a King of Diamonds: the card is face-down, so Alice's observational powers do not permit her to know what it is. In general, we'll think of our agent as having access to some limited "observational tools" (e.g. her vision and hearing) which she can employ to *learn*, that is, to come to know some true propositions about the current point of the situation. There may be propositions which are knowably true (i.e. it's true in the present situation, and moreover our agent is able to know this), or true-but-not-knowably-so (i.e. the proposition is true, but our agent's limited knowledge-gathering abilities do not allow her to know this), or knowably false, or false-but-not-knowably-so. This is the "epistemic structure" of the situation.

Note that we're restricting our attention to just what the agent is *able* to know, and won't endeavour (here) to model the actual state of the agent's knowledge.[17] We don't yet have the tools to articulate precisely what it means for an agent to "be able to know" something in a given situation, but we will. For now, it suffices to note that we're interested in what an agent is able to know, and want our account of situations to be able to encode such features.

The frontier between "acting" and "learning" in our framework deserves clarifying. For simplicity, we assume that *learning is contained within a given point.* In a blackjack game, Alice can make all possible observations quickly, and so merely *observing* does not take her to a different

[17]This idea is pursued in [5, Section 5]. There, the agent's *actual* state of knowledge is explicitly modeled as an additional parameter, which certain actions are capable of mutating. Our intention with the present work is that such an enrichment *could* be added to the models we design here, though the exact details are left to future work.

state.[18] Relatedly, we don't take "Alice could come to know φ" to be true in a point x if the only way Alice could come to know φ is by taking some action and moving to a different point x' where she can come to know that φ *was* true in x. Actions which change the situation do not count as "epistemic tools" in the foregoing sense. This is really the only choice of convention which will make an account of nondeterminism (in particular, coin-flipping) tenable: we want to be able to say "Alice cannot know in advance whether the coin will come up tails". In order to correctly capture what "know in advance" means, we need our sense of "knowability" to be *prior* to actions.[19]

There are numerous other philosophical and psychological features which make up the possibilities for an agent's experience of a situation (e.g. her beliefs, her desires, the abilities, knowledge, and intentions of other agents, etc.), but we will limit ourselves to just these. As we'll see, just these structural components (what's true, what can the agent do, and what can the agent know) will be more than enough to occupy our attention. Let's then proceed to articulate the logical-mathematical structure of situations with this structure. We begin by introducing some more formal notation.[20]

1.2 Epistemology and the Language of ADTL

For the duration of this thesis, the formal languages we'll be primarily working with are instances of $\mathcal{L}_{\Box\bigcirc}(\Sigma)$.

Definition 1.1

Let Σ be some given countable set, whose elements we'll call "actions" or "programs",[21] and let Φ be a given countable set of primitive (or "atomic") propositions. We define $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ by the grammar

$$\varphi, \psi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid \bigcirc_\sigma \varphi \mid \Box\varphi$$

where p varies over Φ and σ varies over Σ. We'll make use of the standard abbreviations, e.g. $\varphi \to \psi$ for $\neg(\varphi \wedge \neg\psi)$, $\top$ for $\neg(p \wedge \neg p)$, and $\Diamond\varphi$ for $\neg\Box\neg\varphi$.

$\mathcal{L}_{\Box\bigcirc}(\Sigma)$, suitably interpreted, is just the right language to express the above-mentioned properties of agents and situations. Consider an agent $\mathcal{A}$ in some point x of a situation S. As above, there are some actions available to $\mathcal{A}$ in x, each of which would take $\mathcal{A}$ to a (possibly) different point x' of S. $\mathcal{A}$ also has access to tools of inquiry and is thereby able to come to know the truth or falsity of various propositions in her current state. But, importantly, there are perhaps statements which are true at x which $\mathcal{A}$ lacks the ability to know. We're able to formulate statements about $\mathcal{A}$'s interaction with S (and possibilities of movement *through* S) *in the language* $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ via the following interpretation.

- The set Φ collects all the basic, extralogical propositions which can be true or false in the situation. So if $p \in \Phi$, then p is true at x just in case the corresponding proposition holds of

[18] In real life, there are often situations where knowledge-gathering takes a non-negligible amount of time. In particular, real agents often find themselves having to balance diligence versus decisiveness, as their possibilities for action evolve the longer they take gathering information. Modelling situations with a more complex interplay between knowability and possible action will require more work than we care to get into here, so we'll generally assume that – like in a game of blackjack – our agent can conduct her observational due diligence in a negligible amount of time, without changing the state (in particular, without changing the suite of available actions).

[19] Of course, this absolutely does not preclude the possibility of there being actions which, upon being performed, land the agent in a different, more transparent situation. "Learn to read" is such an action: upon performance of "learn to read", our agent finds themselves in a different situation (because in the prior situation they didn't know how to read), and in this resulting situation the world is significantly more transparent to them – there's so many more true propositions φ which the agent is, in this situation, able to know.

[20] All these definitions are taken (with some minor modifications) from [5].

[21] Or, perhaps more precisely: Σ is the set of *names* of actions/programs, in that elements of Σ *denote* actions.

that point of S. For instance, if p denotes the proposition "the dealer's visible card is a 6 of Clubs", then p holds in those points x where the dealer's visible card is a 6 of Clubs.

It's worth noting that we're taking these propositions as atomic, i.e. without any internal structure. Thus, in general there will be no relationship between when p holds and when q holds. The present analysis could perhaps be enriched to support, say, first-order terms and quantification – but we do not do so here. Relatedly, we could perhaps allow for some atomic propositions to be neither true nor false but *meaningless* at certain points. We also preclude this for simplicity, and assume that each atomic proposition (and therefore every formula) is either true or false at each point of the situation.

- We're able to close our statements under the classical propositional connectives, and understand their truth conditions accordingly. For instance, $\varphi \wedge \psi$ holds in x just in case φ holds in x and ψ holds in x.

- We assume that in x – and in any other possible state of S – every possible action is denoted by some element of Σ (but allow for elements of Σ to lack a denotation in some – or all – situations). So if $\sigma \in \Sigma$, then the formula $\bigcirc_\sigma \varphi$ is read "after σ, φ holds". This proposition is true at a point x just in case (a) it is possible to perform σ from x, and (b) doing so would result in a state s' where φ holds.

  If $\sigma \in \Sigma$ has no denotation at x, we simply understand this to mean that the action denoted by σ is not possible in x,[22] in which case every proposition $\bigcirc_\sigma \varphi$ is vacuously **false** at x: if executing σ from x is not even possible (i.e. there is no well-defined situation that σ results in), then it's certainly not the case that executing σ at x would result in a point where φ is true.[23] The three possibilities are depicted in Figure 1.2.

- Finally, we read the proposition $\Box\varphi$ as "φ is knowably true", "φ is measurably true", or "the agent *could* come to know[24] that φ is true". What it means for $\Box\varphi$ to be true at x is not merely that φ is true at x, but that $\mathcal{A}$'s tools of inquiry are powerful enough that $\mathcal{A}$ could come to know that $\mathcal{A}$ is true. Note that this notion is entirely relative to $\mathcal{A}$ and her knowledge-gathering tools: if $\mathcal{A}$ has weak learning powers, then her situation is more opaque and there are more formulas φ such that φ is true but $\Box\varphi$ is not. If $\mathcal{A}$'s investigative apparatus is more sophisticated, then she has greater capacity for learning and there are more formulas φ such that $\Box\varphi$ holds. Her knowledge-gathering abilities are allowed to vary based on the point: initially, Alice cannot see the dealer's second card, but later in the game it's revealed and she *can* come to know what it is. Our provision above – that this notion of

[22] Or, if we're thinking computationally, it could mean that executing σ in state x would not *terminate*,

[23] Saying that $\bigcirc_\sigma \varphi$ is false at x when σ is impossible is, admittedly, a choice which might not accord with some readers' interpretation of what "after σ, φ" ought to mean. Other conventions could be that "after σ, φ" is true (Lewis-Stalnaker), or simply meaningless (Strawson) if σ is impossible.

We chose "after σ, φ" to be false in our semantics simply because it's most convenient, and nothing seems to depend essentially on it. For what it's worth, the Lewis-Stalnaker version can be recovered in our semantics: if $\bullet_\sigma$ is defined like $\bigcirc_\sigma$, but is vacuously true when σ is undefined, then the following are tautologies.

$$\bullet_\sigma \varphi \leftrightarrow (\bigcirc_\sigma \varphi \vee \neg \bigcirc_\sigma \top) \leftrightarrow \neg \bigcirc_\sigma \neg \varphi$$

$$\bigcirc_\sigma \varphi \leftrightarrow (\bullet_\sigma \varphi \wedge \neg \bullet_\sigma \bot) \leftrightarrow \neg \bullet_\sigma \neg \varphi$$

So readers who prefer a Lewis-Stalnaker convention can view $\bigcirc_\sigma$ as an abbreviation for the right-hand formula (or the middle formula) of the latter tautology. The duality between $\bigcirc_\sigma$ and $\bullet_\sigma$ is yet another instance of the duality between bounded universal and bounded existential quantification, which is central to modal logic.

It's not immediately clear how much different our analysis would be if we followed Strawson's convention (and generally allowed formulas to lack a truth value in some worlds), but I suspect the differences wouldn't be essential.

[24] Or *will* come to know, if she makes the right observations

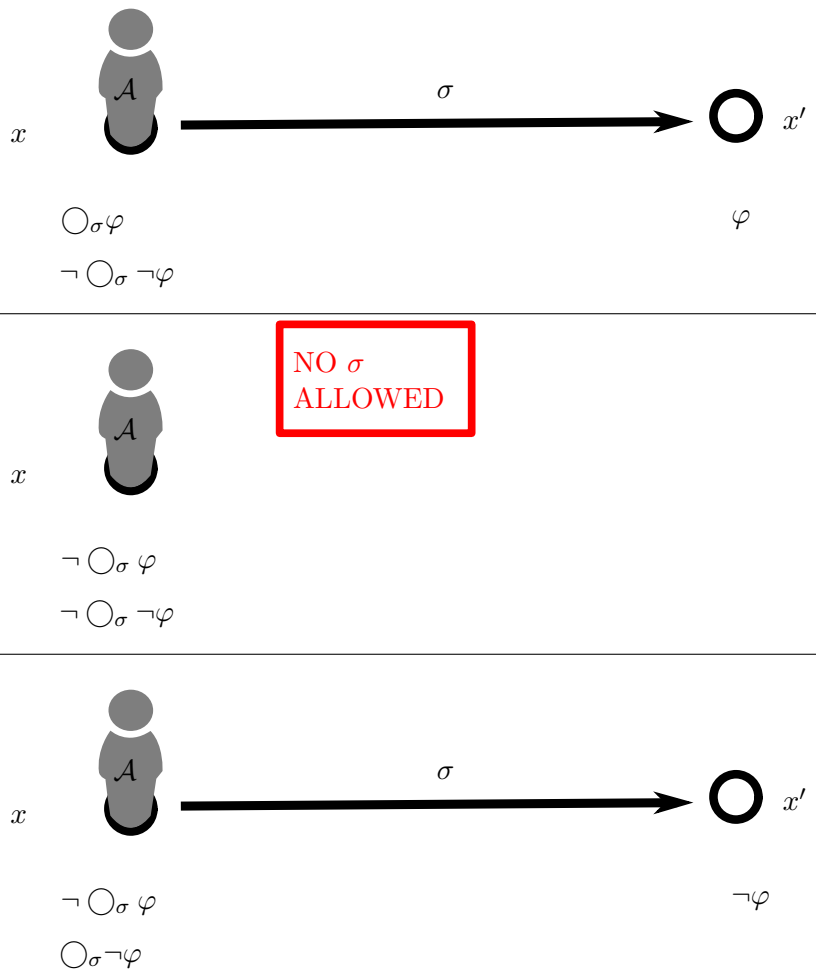


Figure 1.2: The three different combinations of truth values for $\bigcirc_\sigma\varphi$ and $\bigcirc_\sigma\neg\varphi$

"knowability" does not include $\mathcal{A}$ changing the state by undertaking actions – applies here as well.

The dual $\Diamond\varphi$ is shorthand for $\neg\Box\neg\varphi$, i.e. "it is not knowably true that $\neg\varphi$". We treat this as meaning "φ is consistent with any knowledge $\mathcal{A}$ could obtain", or "as far as $\mathcal{A}$ *could* know, φ could be the case".

Before we proceed to specify our mathematical models, let us use this new notation to say one further word about how we're understanding "knowability". Specifically, let us mention that we're realizing the knowability modality as a *contracting* operation on sets of worlds. To see what this means, suppose we had some proposition $\varphi \in \mathcal{L}_{\Box\bigcirc}(\Sigma)$. Let's say that φ is in fact true at the point x which $\mathcal{A}$ finds herself in presently. We might depict that as follows.

We've depicted the current state x (where $\mathcal{A}$ is) alongside various other possible states of the situation. Note that these are depicting "possible worlds", not (necessarily) adjacent physical locations. We've shaded in the "extension" of φ – those worlds where φ is true. When we say we're interpreting knowability as a contracting operation, we mean that the extension of $\Box\varphi$ – the set of points where φ is *knowably true* – is contained in the extension of φ. See Figure 1.4.

So, since $\mathcal{A}$ is in one of the $\Box\varphi$ points (one of the ones in the darker shaded region), that means

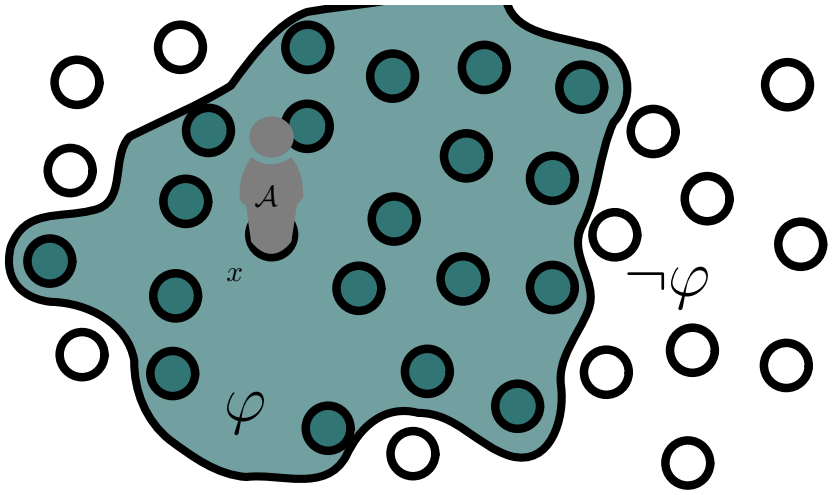


Figure 1.3: Extension of φ

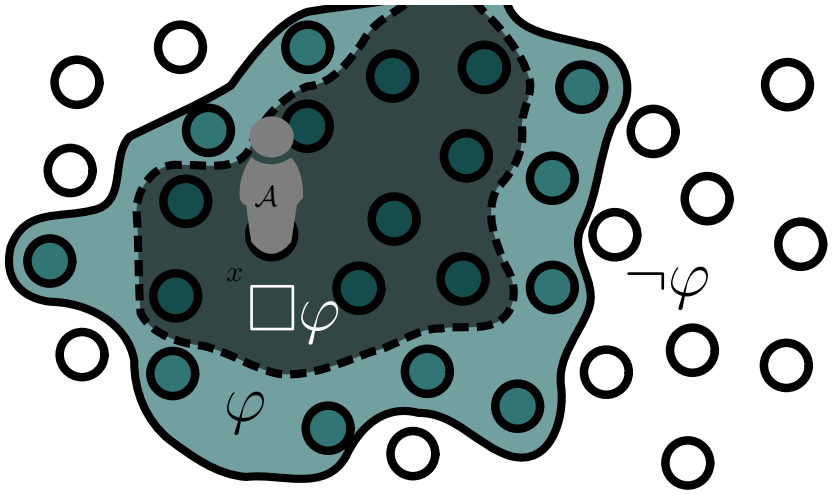


Figure 1.4: Extensions of φ and $\Box\varphi$

that φ is knowably true at x – in this situation, $\mathcal{A}$ can come to know that φ is true. The fact that the extension of $\Box\varphi$ is contained in the extension of φ just means that all $\Box\varphi$-states are φ-states: $\mathcal{A}$ can't come to *know* φ unless φ is, in fact, true.

The worlds in the extension of φ but not in the extension of $\Box\varphi$ (i.e. in the shaded region but not the darker shaded region) are worlds where φ is true, but $\mathcal{A}$ does not possess the tools to know this, i.e. φ is true-but-not-knowably-so. Notice that the extension of $\Box\varphi$ consists of those points which were not merely *in* the extension of φ, but *robustly within it.*[25] So the "φ is true but not knowably so" worlds are "on the border" of the extension of φ, barely inside. These spatial intuitions of "robustly within" or "on the border" are completely informal at this point (we will make it formal shortly), but serves as a useful (and reasonably apt) intuition for how we'll be thinking about knowability. We'll develop this connection more in the next subsection.[26]

To close out this subsection, we've depicted the meaning of $\bigcirc_\sigma\Box\varphi$ and $\Box\bigcirc_\sigma\varphi$ (see Figure 1.5 and Figure 1.6, respectively) as a demonstration (and for reference later when we're dealing with

[25] "Robustly" as opposed to "barely".

[26] We require that this "contraction" operation satisfy Kuratowski's Interior axioms

these formulas). The reader is encouraged to intuitively understand the difference between these two – their difference will go on to play an important role in our analysis (see Subsect. 1.5).

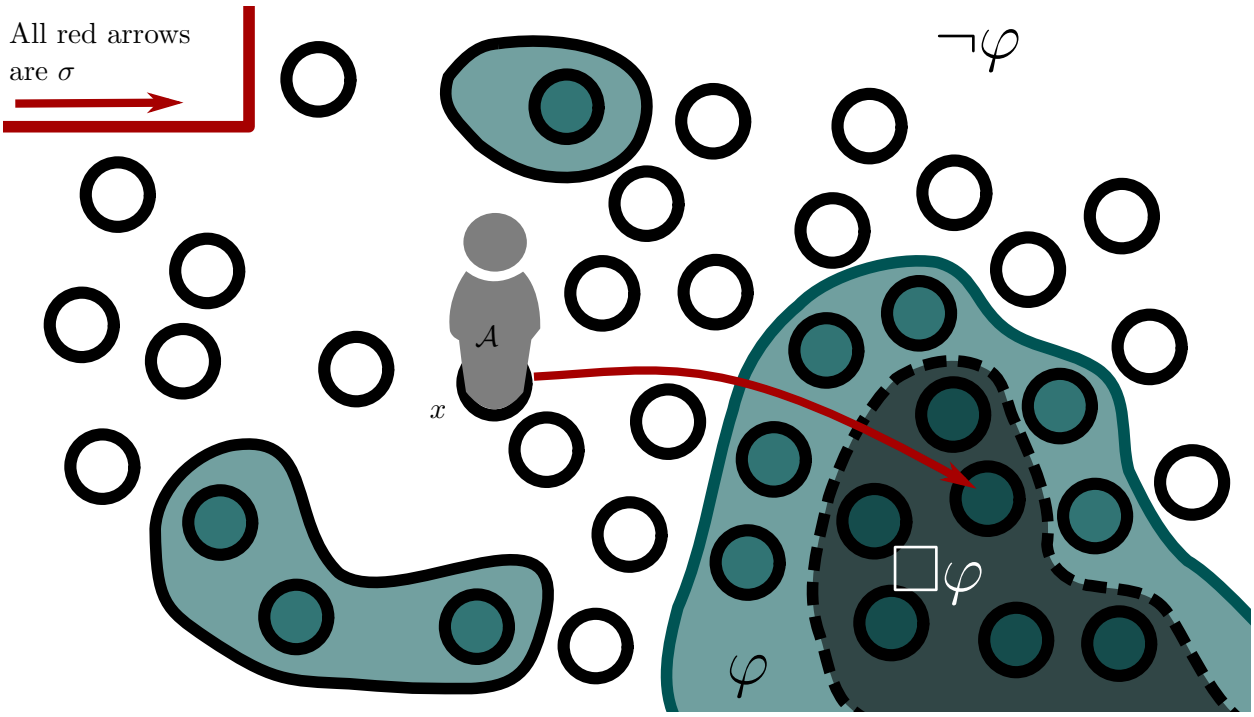


Figure 1.5: $\bigcirc_\sigma\Box\varphi$
"after σ, φ will be knowably true"

1.3 The Theory of Dynamic Topological Models

Our next task will be to develop the model theory of agent DTL, and argue it adequately reflects the epistemic intuitions above. In particular, this theory will make the depictions in the previous subsection (Figure 1.3, Figure 1.4, Figure 1.5, and Figure 1.6) more precise, and give explicit semantics for $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ formulas. Our ultimate task will be to determine the characteristic features of the models (articulated in $\mathcal{L}_{\Box\bigcirc}(\Sigma)$, i.e. object-language axiomatizations of the logic of these models), and then assess these conclusions against our intuitions once again.

$\mathcal{L}_{\Box\bigcirc}(\Sigma)$ is interpreted in mathematical structures known as Σ-*dynamic topological models*, or Σ-*DTMs*. Let's first define what Σ-DTMs are, and then explain how each component of the structure connects to the above considerations of agents in situations.

Definition 1.2

Let Σ be a set of program symbols. A **Σ-dynamic topological frame** ("Σ-frame", or "frame interpreting Σ") is a triple $\mathcal{F} = (X, \tau_X, \{f_\sigma\}_{\sigma\in\Sigma})$ where:

- X is a set (whose elements we'll refer to as the "states", "worlds", or "points" of $\mathcal{F}$)
- τ_X is a topology on X (whose elements we call "open sets")
- For each $\sigma \in \Sigma$, $f_\sigma : X \rightharpoonup X$ is a partial function defined on some (possibly empty) subset of X. We call f_σ the "interpretation of σ in $\mathcal{F}$".

If $\mathcal{F}$ is a Σ-frame and $V : \Phi \to \mathcal{P}(X)$ is a function sending propositional letters p to subsets of X, we say that V is a **valuation** on $\mathcal{F}$ and call $V(p)$ the "extension" of p in $(\mathcal{F}, V)$. In

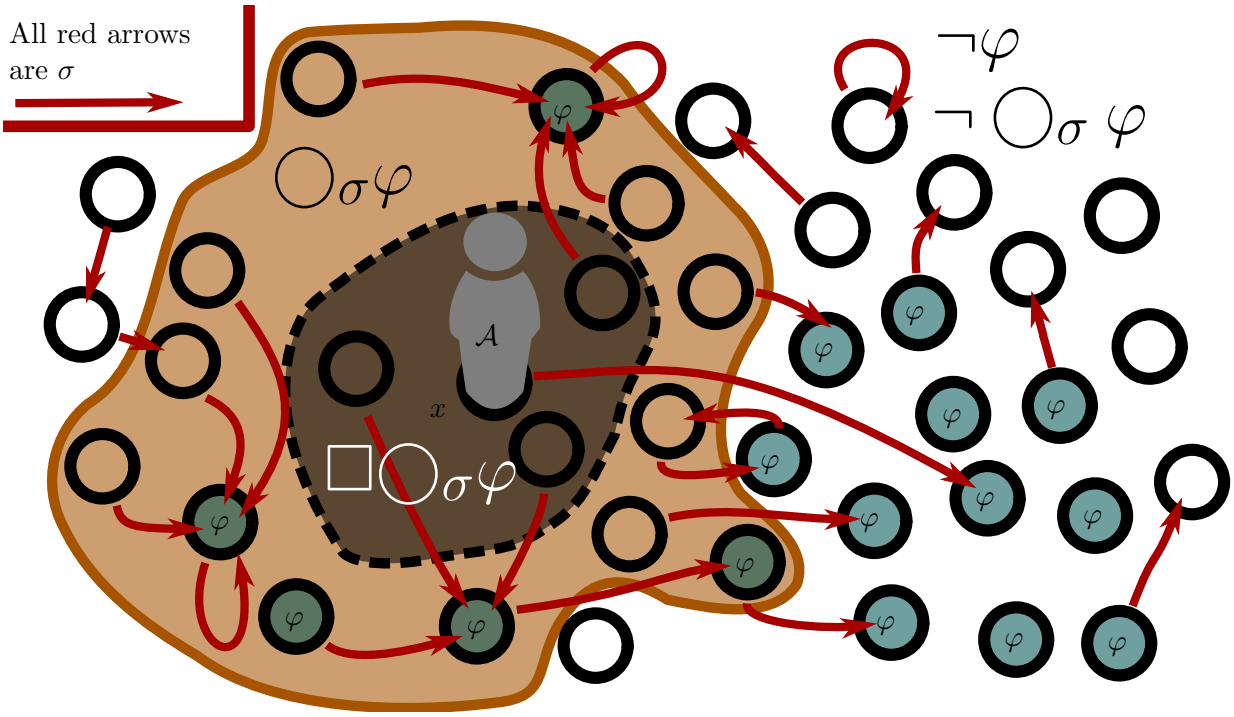


Figure 1.6: $\Box\bigcirc_\sigma\varphi$
"$\mathcal{A}$ can come to know (before executing σ) that executing σ will result in a world where φ is true". Notice that the orange region (the extension of $\bigcirc_\sigma\varphi$) consists of exactly those worlds which have an outgoing σ-arrow which terminates in a φ-world.

this case, $\mathfrak{M} = (X, \tau_X, \{f_\sigma\}, V)$ – also indicated by $\mathfrak{M} = (\mathcal{F}, V)$ – is called a **Σ-dynamic topological model** ("Σ-DTM", or "DTM interpreting Σ").

For a Σ-DTM $\mathfrak{M}$ (respectively: a Σ-frame $\mathcal{F}$), we write $|\mathfrak{M}|$ (resp. $|\mathcal{F}|$) for the set X underlying $\mathfrak{M}$ (resp. $\mathcal{F}$); for any $\sigma \in \Sigma$, we write $\|\sigma\|_{\mathfrak{M}}$ (resp. $\|\sigma\|_{\mathcal{F}}$) to denote the partial function $f_\sigma : |\mathfrak{M}| \rightharpoonup |\mathfrak{M}|$ interpreting σ in $\mathfrak{M}$ (resp. $\mathcal{F}$).[27]

Note that both the language $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ and the definition of a Σ-DTM are parametric over the "program set" Σ.[28] Indeed, we can discuss the interaction between DTMs with different program sets. For instance, notice that if $\Sigma \subseteq \Theta$, then every Θ-frame is also a Σ-frame (and every Θ-DTM is a Σ-DTM) – just forget that it has interpretations for $\Theta \setminus \Sigma$.

As promised, Σ-DTMs interpret the formulas of $\mathcal{L}_{\Box\bigcirc}(\Sigma)$.

Definition 1.3

Given a Σ-DTM $\mathfrak{M} = (X, \tau_X, \{f_\sigma\}, V)$, and a world $x \in X$, we interpret formulas of $\mathcal{L}_{\Box\bigcirc}(\Sigma)$

[27] Of course, $|(\mathcal{F}, V)| = |\mathcal{F}|$.

[28] They're also both parametric over the set Φ of primitive propositions, but we suppress this fact by generally assuming that Φ is fixed in the background.

according to the following semantics.

$$
\begin{aligned}
(\mathfrak{M}, x) &\models p && \iff x \in V(p) && (p \in \Phi)\\
(\mathfrak{M}, x) &\models \neg\varphi && \iff (\mathfrak{M}, x) \not\models \varphi\\
(\mathfrak{M}, x) &\models \varphi \wedge \psi && \iff (\mathfrak{M}, x) \models \varphi \text{ and } (\mathfrak{M}, x) \models \psi\\
(\mathfrak{M}, x) &\models \Box\varphi && \iff x \in \mathrm{int}([\![\varphi]\!])\\
(\mathfrak{M}, x) &\models \bigcirc_\sigma\varphi && \iff f_\sigma(x) \text{ is defined and } (\mathfrak{M}, f_\sigma(x)) \models \varphi && (\sigma \in \Sigma)
\end{aligned}
$$

The judgment $(\mathfrak{M}, x) \models \varphi$ is pronounced "$(\mathfrak{M}, x)$ validates φ". We write $[\![\varphi]\!]$ to indicate the set of worlds $x \in X$ such that $(\mathfrak{M}, x) \models \varphi$. And $\mathrm{int}(A)$ denotes the topological interior of $A \subseteq X$, according to the topology τ_X (see below).

We write $\mathfrak{M} \models \varphi$ and say "$\mathfrak{M}$ validates φ" just in case $(\mathfrak{M}, x) \models \varphi$ for all $x \in X$. If Δ is some set of $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ formulas, we write $\mathfrak{M} \models \Delta$ to indicate that $\mathfrak{M} \models \varphi$ for all $\varphi \in \Delta$.

We connect these semantics to the above discussion by claiming that a Σ-DTM $\mathfrak{M}$ is a *mathematical model of a situation*, and the $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ formulas which $\mathfrak{M}$ validates express features of that situation. Recall that we were concerned with the internal structure of the situation, and how the different "points" or "states" of the situation relate to each other. In particular, we wanted our models of situations to encode the possibility for the agent to move to a different state of the situation by performing actions, as well as the agent's capacity for coming to know various true propositions at different points. A Σ-DTM models exactly these features. The worlds $x \in X$ of a Σ-DTM $\mathfrak{M}$ represent the different states of the situation, the partial functions encode the ability of the agent to move through the situation by their actions, and the topological structure models the agent's capacity for knowledge. We'll see that the semantics for Σ-DTMs interprets $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ formulas in a way which reflects the above-established intuitions about what these formulas express philosophically, so the $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ formulas validated by the Σ-DTM meaningfully articulate the properties of dynamic-epistemic situations.

Let's begin by noting how the definition of a Σ-DTM accurately captures the structure of states within the situation and the structure of actions which move the agent between these states. Returning to our above example of Alice playing blackjack, we conceptualized the situation as a collection of "points" or "states" that Alice proceeded between by performing actions (e.g. if she were at point x, then perhps executing action σ could land her in a different world x'. Mathematically, this is most straightforwardly expressed as a set (the set of states in the situation) with a structure of partial functions (representing the effect – and possibility – of various actions at each state). So we assume that every possible action (at any point) is denoted by some element of Σ,[29] and then we interpret the action denoted by $\sigma \in \Sigma$ as a partial function f_σ from states to states. If $f_\sigma(x)$ is defined for some state x, we interpret that as σ being "possible" or "allowed"[30] at point x. And the resulting point $f_\sigma(x)$ is the state once the action is complete. For instance, if x is a state where it's Alice's turn and she has been dealt a 9 of spades and a 7 of clubs, then the action "tapping the table" – which we'll denote by σ – is allowed. If she does it, that will result in some new game state $x' = f_\sigma(x)$ where Alice has one more card than she did in x (the dealer dealt her one more card, as a consequence of her tapping the table). Let φ denote the proposition that Alice is still in the game. Suppose the card she'll be dealt if she taps the table will cause her to lose and no longer be in the game. So we'd say that $\neg\varphi$ held at x', because φ is not true there. And we'd also say that "after σ, $\neg\varphi$" held at x, because it is indeed the case that performing the action denoted by σ from x would result in a $\neg\varphi$ state: it results in x'. Fittingly, this is precisely the

[29] A plausible assumption, since Σ is allowed to be infinite.

[30] E.g. an allowable move for the agent to make, according to the rules of blackjack.

circumstances under which our Σ-DTM semantics for $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ would declare $\bigcirc_\sigma\neg\varphi$ to hold at x, affirming our choice of interpretation.

And, moreover, the topological semantics for $\Box$ accord with our interpretation of $\Box\varphi$ as "the agent could come to know φ", and with our understanding of Σ-DTMs as models of how agents reason their way through situations.[31] We indicated that our agents are able to make observations which inform them of the truth or falsity of some propositions about the current world. Recall that, in a game of blackjack, one of the dealer's cards is visible ("face-up") and the other is hidden, initially. However, at a later time the dealer's second card is revealed. At the point of the game x where Alice is deciding whether to motion the dealer to deal her another card or not, she can only see one of the dealer's cards – the second hasn't been revealed. But at some later point (call it x''), the dealer's other card is revealed and she *can* observe it. So in x the agent can come to know that the dealer's face-up card is, say, a 4. In x'' she can know this, and she can know (she *will* know if she just looks) that the dealer's other card is, say, the King of Diamonds. To incorporate this into our framework, we want to specify a criterion for which φ-states are moreover $\Box\varphi$ states. The topology will accomplish this beautifully.

How do we make sense of her "observing" at x that the dealer's face-up card is a 4? Well, what we want an observation to encode is *information for the agent about what possible state she's in.* We do so by making an "observation" a subset of the set X of all possible states. If the agent observes $O \subseteq X$, she knows that, whatever world she's presently at, it must be in O. This thereby rules out all the non-O worlds as worlds she might presently be at.[32] So, for instance, if she observes that the dealer's first card is a 4, then O is the set of all points where the dealer's first card is a 4. By making this observation, Alice has ruled out all the non-O worlds, and we would now say that she *knows* the dealer's first card is a 4. Since it can be known at x (i.e. the information is available at x) that the dealer's first card is a 4,

$$x \models \Box(\text{the dealer's first card is a 4}).$$

Notice that, more generally, this is true because $O \subseteq [\![\text{the dealer's first card is a 4}]\!]$: after making the observation O, every world Alice considers possible is a world where the dealer's first card is a 4. This will be our notion of what it means for Alice to "know" that a given proposition is true. We won't deal here with any concern about the agent's observations being faulty: we'll always assume that the present world x must be included in any observation O the agent makes. In other words, all her observations must be consistent with the actual world, and therefore she can only come to know things which are true.

However, there are propositions Alice *cannot* come to know, even though they are true: Alice cannot observe at x that the dealer's second card is the King of Diamonds. So even if it is indeed true that the dealer's second card is the King of Diamonds,

$$x \not\models \Box(\text{the dealer's second card is the King of Diamonds}).$$

At x, the card is face-down, and so Alice can only observe the back of the card (which carries no information). Thus, any observation she can make in that regard is consistent with the card being something else (i.e. *not* being the King of Diamonds), hence why she cannot *know* it is actually the King of Diamonds. Phrased more mathematically, for any observation O, we have that

$$O \not\subseteq [\![\text{the dealer's second card is the King of Diamonds}]\!]$$

So there are some subsets of X which are not valid observations the agent can make: there is no observation $O \subseteq X$ the agent can make at x ($x \in O$) which tells her that the dealer's second card is the King of Diamonds ($O \subseteq [\![\text{the dealer's second card is the King of Diamonds}]\!]$).

[31] The notion of "knowability" we advance here is particularly close to that of [6] and [**?**]

[32] If we identify a proposition with its extension, then "observing O" is "observing *that* O"

So, more generally, a world x validates $\Box\varphi$ iff there's an observation O the agent is allowed to make such that $x \in O \subseteq [\![\varphi]\!]$. But we must somehow constrain the set of subsets of X which count as "observations" in this sense. For a very specific example (like blackjack), we can enumerate all the different pieces of information the agent can come to obtain by observation and specify the allowed observations accordingly. But in general, we'd like some conditions which say a bit more about what minimal properties we want to require of these subsets of X in order for them to count as "observations". It results in an elegant theory if we require that the set $\mathcal{B}$ of all possible observations the agent could make satisfies the following two properties

(B1) For all $x \in X$, there's an $O \in \mathcal{B}$ such that $x \in O$

(B2) If $O_0, O_1 \in \mathcal{B}$ and $x \in O_0 \cap O_1$, then there's a $O_2 \in \mathcal{B}$ such that

$$x \in O_2 \subseteq O_0 \cap O_1$$

(B1) is something of a tautology: for every state there must be some observation possible. The easiest way to satisfy this is to just put $X \in \mathcal{B}$: the agent can make the trivial observation which does not rule out any possible states. (B2) is a bit trickier to interpret, and encodes as certain requirement about the manner in which the agent makes observations. It says that, for any pair of observations O_0, O_1 the agent could make, there's some other observation O_2 which is at least as strong as both ("strong" in the sense of ruling out more worlds, i.e. all worlds that are consistent with O_2 are consistent with both O_0 and O_1). That is plausible in the blackjack situation, for example, because the agent is able to observe all the available information (their two cards, and the dealer's exposed card)[33] essentially instantaneously, so we can think of there being a single observation the agent could make at x which represents everything they could know at x, satisfying (B2). To make better sense of (B2) would require more of a detour than we care to take here. For our purposes, we will accept (B2) because, as mentioned, it allows for an elegant mathematical description of knowability. As we'll see, the notion we develop from this will turn out to be plausible description of the logic of knowability, so our acceptance of (B2) will not prove misguided.

If we have a collection $\mathcal{B}$ satisfying these two axioms, then $\mathcal{B}$ is what's known as a *topological basis*. As is developed in any standard account of topology, topological bases are all you need to specify a topology: each topological basis $\mathcal{B}$ "generates" a topology $\tau(\mathcal{B})$ (see the Mathematical Reference, Subsect. 0.1). So when we say that a Σ-DTM $\mathfrak{M}$ comes equipped with a topology, we can think of that as saying that $\mathfrak{M}$ comes equipped with a *topological basis*, specifying what observations the agent is able to make. However, it turns out that the notion of knowability we develop below depends on the topology the basis generates, not the basis itself. In other words, for two such "observation bases" $\mathcal{B}$ and $\mathcal{B}'$ which generate the same topology ($\tau(\mathcal{B}) = \tau(\mathcal{B}')$), the resulting notion of "knowability" is the same. So, when we reference a basis $\mathcal{B}$ below, it suffices to pick *any* basis which generates the topology of the DTM in question.

So why is $\Box\varphi$ true on the interior of $[\![\varphi]\!]$, according to the above-given semantics? Well, from the above discussion about the information content of an agent's observations and what it means for her to *know* something, we obtain the following principle.

> An agent can come to know φ at x iff there exists an observation O the agent could make such that $x \in O \subseteq [\![\varphi]\!]$

[33] Ignore for the moment the possibility of there being other players (the other players' cards are also visible, which is also information available to the agent). Also ignore the possibility that the agent could know more about the situation, e.g. that all the Queens appeared in the previous hand and would therefore be unlikely to appear in the current hand. We could perhaps expand the scope of our example to consider these.

And so – assuming the set $\mathcal{B}$ of observations the agent can make satisfies (B1) and (B2), we have the following result:

Proposition 1.1
For $\mathcal{B}$ satisfying (B1) and (B2), the following are equivalent for any $x \in X$ and any $A \subseteq X$

- There exists $O \in \mathcal{B}$ such that $x \in O \subseteq A$
- x is in the interior of A with respect to the topology $\tau(\mathcal{B})$

So if $\mathcal{B}$ is the topological basis corresponding to observations our agent can make, then she can come to know φ at x iff $x \in \text{int}([\![\varphi]\!])$, where interior is taken with respect to the generated topology $\tau(\mathcal{B})$. This is why we said that it doesn't matter what basis $\mathcal{B}$ we use so long as it generates the same topology: observe that two such bases $\mathcal{B}$ and $\mathcal{B}'$ would give the same notion of interior, and therefore the same notion of "knowability". In summary: we equip DTMs with topologies, because topologies encode what kinds of thing the agent is able to know by making observations (in the foregoing sense). The notion of "knowability" we get out is topological interior: our agent can come to know φ on exactly those points which are on the interior of $[\![\varphi]\!]$.

Let us also note that interior is a "contracting operation", in the sense that $\text{int}(A) \subseteq A$ for all $A \subseteq X$. This makes good on our discussion earlier of how $[\![\Box\varphi]\!]$ should be contained in $[\![\varphi]\!]$. With that, we have argued for the plausibility of understanding Σ-DTMs as models of what an agent can do and know inside a situation. The reason we wanted such models is to study their properties, and to obtain theorems (in the language $\mathcal{L}_{\Box\bigcirc}(\Sigma)$) articulating how the situation works. We see that Σ-DTMs can validate or refute such formulas, and so let us now turn our attention to understanding *which* formulas Σ-DTMs validate, obtaining $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ formulas to check against our intuitions of agents in situations. Our system for obtaining these formulas is the deductive calculus ADTL.

1.4 The Deductive Calculus of Agent DTL

The deductive calculus ADTL is a system which declares certain $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ formulas to be "theorems" and the rest "nontheorems". As usual in mathematical logic, the definition of what constitutes a theorem of ADTL is recursively defined as the outcome of a "proving" process.

Definition 1.4
The system ADTL consists of the following *axiom schemes* and *inference rules* (summarized in the table).

- Classical propositional logic: all (or enough) tautologies of classical propositional logic (over $\mathcal{L}_{\Box\bigcirc}(\Sigma)$) and the inference rule of *Modus Ponens*;
- $\mathsf{S4}_{\Box}$: all instances ($\varphi, \psi \in \mathcal{L}_{\Box\bigcirc}(\Sigma)$) of the (K) scheme, the (T) scheme, and the (4) scheme, and the inference rule of *Necessitation*;
- All instances ($\sigma \in \Sigma$, $\varphi, \psi \in \mathcal{L}_{\Box\bigcirc}(\Sigma)$) of the ($\neg$-PC) scheme, the ($\wedge$-C) scheme, and the rule of *Monotonicity*.

	ADTL Axioms and Inferences	
(CPL)	enough tautologies of CPL	
(MP)	from $\varphi \to \psi$ and φ, infer ψ	Modus Ponens
(K)	$\Box(\varphi \to \psi) \to \Box\varphi \to \Box\psi$	Distribution
(T)	$\Box\varphi \to \varphi$	Reflexivity
(4)	$\Box\varphi \to \Box\Box\varphi$	Transitivity
(Nec)	from φ, infer $\Box\varphi$	Necessitation
($\neg$-PC)	$\bigcirc_\sigma\neg\varphi \leftrightarrow (\neg \bigcirc_\sigma \varphi \wedge \bigcirc_\sigma\top)$	$\neg$-Partial Commutativity
($\wedge$-C)	$\bigcirc_\sigma(\varphi \wedge \psi) \leftrightarrow \bigcirc_\sigma\varphi \wedge \bigcirc_\sigma\psi$	$\wedge$-Commutativity
(Mon)	from $\varphi \to \psi$, infer $\bigcirc_\sigma\varphi \to \bigcirc_\sigma\psi$	Monotonicity

We say a formula $\varphi \in \mathcal{L}_{\Box\bigcirc}(\Sigma)$ is a **theorem** of ADTL, and write $\vdash \varphi$ if there exists a *proof* of φ: a sequence of formulas $\varphi_1, \ldots, \varphi_n$ where $\varphi_n = \varphi$ and for each $i = 1, \ldots, n$, either

- $\varphi_i = \varphi_j$ for some $j < i$,
- φ_i is an instance of one of the axiom schemes of ADTL, or
- φ_i can be inferred from $\varphi_1, \ldots, \varphi_{i-1}$ according to the rules of inference of ADTL.

These axioms, I claim, hold up under an epistemic interpretation. For instance, (T) asserts that if φ is knowably true, then φ is indeed true. (Nec) asserts that the agent can come to know any tautology (not that she *does* know, necessarily, but that she *can*). ($\neg$-PC) states that the statement "doing σ will result in a $\neg\varphi$-world" is equivalent to "doing σ won't result in a φ-world, but will result in some world (i.e. is possible)". (Mon) states that if it's a tautology (i.e. is true everywhere) that φ implies ψ, then it's a tautology that "after σ, φ" implies "after σ, ψ". And so on.

It turns out that these axioms are enough to fully capture the logic of Σ-DTMs.

Theorem 1.2 (Kremer and Mints, Bjorndahl)
ADTL is a sound and complete axiomatization of $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ with respect to the class of Σ-dynamic topological models: for all $\varphi \in \mathcal{L}_{\Box\bigcirc}(\Sigma)$,

$$\vdash \varphi \qquad \text{if and only if} \qquad \mathfrak{M} \models \varphi \text{ for all } \mathfrak{M}.$$

Theorem 1.2 affirms our epistemic interpretation. We chose to work with Σ-DTMs with an eye towards modelling how an agent interacts with a situation, e.g. the choice of partial functions to model the "state transition" nature of an acting agent, and the use of topology to articulate the capacities of the agent for knowledge. With the deductive system of ADTL, we have a characterization of *all* the tautologies of the theory of Σ-DTMs, and, it seems, all the theorems seem plausible from our interpretation. Henceforth, we will take this as license to reason about Σ-DTMs as if they were epistemic situations, and talk about how an agent would move through such a situation, and what options were available to them in it.

With that, we can (so to speak) "take the interpretation and run with it". As we'll see, we can develop the agent DTL-epistemology connection further, and encounter various dynamic-epistemic phenomena (i.e. phenomena arising from the interaction of a knowledgeable agent with their situation) whose internal logic can be expressed using agent DTL, via this interpretation. We'll begin by reviewing one such phenomenon identified by [5]: *nondeterminism*. We'll then conclude our introductory section by looking at propositional dynamic logic and its "program constructors". The task of developing PDL-style program constructors in the setting of dynamic topological logic (and interpreting them epistemically) will be our main goal.

1.5 Determinism and Continuity

In our example of Alice playing blackjack, the chief dilemma she faced was an inability to predict the outcomes of her actions: if Alice ordered another card from the dealer, she had no way of knowing in advance whether the new card would be advantageous for her or would result in her losing the game. This is, in a sense, the essence of gambling: the casino has crafted a situation for Alice wherein she cannot know the outcome of her actions, and she must decide how to act in face of such uncertainty. We emphasize the point that we're considering the epistemic situation *only from Alice's perspective*: while a real casino will achieve uncertainty by dealing from a shuffled deck (which is "random" in some sense), that's not actually necessary to create such an epistemic situation for Alice: it's perfectly possible that, say, the dealer could already know what the next card is. There could even be some specific pattern to how the cards are coming out (the casino may have, as a matter of fact, fixed the sequence of cards that get dealt). As long as *Alice* cannot know what the next card is, she faces an uncertain choice.[34] It is this kind of "unpredict*able* action" that we'll deem "nondeterministic" (and its negation – actions whose outcomes Alice *can* fully account for – that we'll call "deterministic").

The content of the present subsection can be summarized as follows: three notions – continuity (in the theory of Σ-DTMs), determinism (in the world of agents and situations), and the partial commutativity of $\Box$ and $\bigcirc_\sigma$ (in the proof calculus of ADTL) – all correspond in logical structure. Indeed, we intend to argue that continuity is precisely the mathematical model of determinism (from the standpoint of an agent and their capacities in a particular situation). Moreover, the claim "σ is deterministic" can be formalized as a single axiom scheme in the object language $\mathcal{L}_{\Box\bigcirc}(\Sigma)$, and this axiom scheme picks out precisely those DTMs whose interpretation of σ is continuous.

The study of continuous functions is central to the field of topology. If (X, τ_X) and (Y, τ_Y) are topological spaces and $f : X \to Y$ is a function between them, then we call f **continuous** just in case $f^{-1}(U') \in \tau_X$ for all $U' \in \tau_Y$. In words: if the preimage of each open $U' \subseteq Y$ under f is open with respect to τ_X. This articulates a sense of the function "reflecting" topological structure: any open subset of the image can be pulled back to an open set of the domain. We also should note that nothing about the definition above relies on f being a *total* function, that is, on $f(x)$ defined for *every* x. It still makes sense to talk of continuity when f is allowed to be a partial function: $f^{-1}(U')$ is just to be defined as the set of those $x \in X$ such that $f(x)$ is defined and $f(x) \in U'$, a straightforward generalization of our definition for total functions. And so we can define continuity of partial functions in the same way we did for total functions ($f^{-1}(U') \in \tau_X$ for all $U' \in \tau_Y$); it therefore makes sense to wonder whether a function f_σ (interpreting $\sigma \in \Sigma$ in a Σ-DTM $\mathfrak{M}$) is continuous or not with respect to the topology of $\mathfrak{M}$.

A natural question to ask is whether we can define (in the object language $\mathcal{L}_{\Box\bigcirc}(\Sigma)$) the class of Σ-frames $\mathcal{F}$ such that $\|\sigma\|_{\mathcal{F}} : |\mathcal{F}| \rightharpoonup |\mathcal{F}|$ is continuous for all (or some) $\sigma \in \Sigma$. It turns out we *can*: as shown in [5, Proposition 1], the scheme

$$\bigcirc_\sigma \Box \varphi \to \Box \bigcirc_\sigma \varphi$$

defines the class of Σ-frames $\mathcal{F}$ where $\|\sigma\|_{\mathcal{F}}$ is continuous. In other words, the scheme formally characterizes continuity (and we can require continuity for a specific set of σ by just asserting the scheme for those σ). So we have connected the topological notion of continuity to a particular form of $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ formula. Now we complete the legs of our overall analogical "triangle" and connect these two to considerations in epistemology.

Let's consider the meaning of the axiom scheme

$$\bigcirc_\sigma \Box \varphi \to \Box \bigcirc_\sigma \varphi \qquad (*)$$

[34] We could perhaps say that nondeterminism is a necessary, but not sufficient condition for "randomness".

under the epistemic interpretation given above. First note that we depicted the antecedent and consequent of (*) in Figure 1.5 and Figure 1.6, so those pictures may help serve intuitions for our discussion. Now, let's read (*) under the given epistemic interpretation. We get: "if, after executing σ it is knowably the case that φ holds, it is knowably the case *before executing* σ that doing so will result in a φ world". In other words, the agent doesn't need to perform σ to learn whether doing so will result in a φ world. If we call the agent's present state x and x' the state she would find herself upon completing the action σ (assuming there is such a world), then (*) says that our agent cannot know in x' that φ is true unless this information was accessible to her back in x: "after σ, φ will be knowably the case" being true in x implies that "it is knowably the case that after σ, φ" is true in x. If (*) holds at x for all φ (i.e. x validates the scheme, instantiated for every formula φ), this actually serves as a plausible notion for σ being epistemically **deterministic** at x: what it rules out is the possibility of there being any "uncertainty" which can only be resolved by an execution of σ itself: for any φ you're able to know at x', you're able to know at x that performing σ would land you in a φ-world. We could perhaps phrase the above as "the impacts of σ are *transparent* to our agent": nothing about the execution of σ should surprise a sufficiently-diligent agent, because there's no formulas φ which could be known to be true after the execution of σ ($\bigcirc_\sigma \Box \varphi$) which our agent could not have known *before* executing σ that executing σ would result in a φ world ($\Box \bigcirc_\sigma \varphi$). The inclusion of the antecedent (that φ must indeed be knowable at x') allows for there to be generally-unknowable propositions: if there's some φ which the agent cannot even know φ upon reaching x', then her inability to know "after σ, φ" at x shouldn't count against the determinism of σ.

As stated above, $f_\sigma : X \rightharpoonup X$ is continuous with respect to τ_X just in case $f_\sigma^{-1}(U) \in \tau_X$ for any $U \in \tau_X$. Here's an equivalent way of stating it (see Prop. 4.1): for any $A \subseteq X$, if $f_\sigma(x) = x'$ and $x' \in \text{int}(A)$, then $x \in \text{int}(f_\sigma^{-1}(A))$. Interpreting DTMs as models of situations: if, inside the situation at state x, the action σ will take the agent to a state x' where she could observe that she's in A,[35] then she can, in fact, make sufficient observations in x (*prior* to executing σ) that allows her to know that executing σ *will* take her to a world in A. This is *not* true in the blackjack example: suppose x is the beginning of Alice's turn, and x' is the state after she gets an additional card which causes her to lose. In x', Alice can observe that she has lost the game. But there is no observation she can make in x which guarantees that this will happen: she can't observe anything about what the next card will be, so – as far as she can know – getting dealt another card *won't* result in her losing. That is just the nondeterminism of the situation (from Alice's perspective). The "uncertainty" inherent to a nondeterministic action can be formally made sense of as the discontinuity of the interpreting function (equivalently, the refutation of the corresponding axiom scheme), as claimed. What we have found, then, is that determinism, continuity, and the continuity axiom scheme all interrelate and characterize each other's logical structure.

1.6 PDL Program Construction

The final ingredient which we'll need for our analysis is relational propositional dynamic logic. Propositional Dynamic Logic (henceforth, PDL) is a simple variety of relational modal logic developed primarily to reason about nondeterministic computer programs. In particular, PDL can be utilized to do Hoare-Logic-style reasoning about preconditions and postconditions of programs (i.e. what's true before and after executing a computer program). We'll build on this intuition by identifying the formal language of PDL with a fragment of the language of agent DTL, and thereby transfer the interpretation (which, as we'll see, works out). We will not extensively introduce the

[35] Recall $x' \in \text{int}(A)$ means that there exists an observation O with $x \in O \subseteq A$ – an observation consistent with x' which guarantees the present world is in A.

technical details of relational PDL, except insofar as we need it as background for our current goal.[36]

PDL is a logic conducted in the language $\mathcal{L}_{PDL}(\Sigma)$, which is given by

$$\varphi, \psi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid [\sigma]\,\varphi.$$

Here, as elsewhere, p varies over some set Φ of primitive propositions and σ over some set Σ of primitive program symbols. We use all the same notational shorthands as before, but additionally write $\langle\sigma\rangle\varphi$ for $\neg\,[\sigma]\,\neg\varphi$. At first glance, this may appear to be alternate notation for the $\{\bigcirc_\sigma\}_{\sigma\in\Sigma}$ fragment of $\mathcal{L}_{\Box\bigcirc}(\Sigma)$, with $[\sigma]\,\varphi$ playing the role of $\bigcirc_\sigma\varphi$. Not quite. The semantic relationship between these languages is somewhat more subtle, as we shall see.

A (Σ-)**relational model**[37] M is a triple $(X, \{R_\sigma\}_{\sigma\in\Sigma}, v)$, where X is a set, $R_\sigma \subseteq X \times X$ is a binary relation for each $\sigma \in \Sigma$ and $v : \Phi \to \mathcal{P}(X)$ is a valuation function. The relation R_σ provides a semantics for $[\sigma]$ in the usual "Kripke-style" way: $(M, x) \models [\sigma]\varphi$ iff $(M, x') \models \varphi$ for all x' such that $xR_\sigma x'$. So we're justified in regarding such a model as just a model of standard relational modal logic, with a collection of $\Box$-style modalities indexed by Σ. This is made precise with the following result.

Definition 1.5

Let PDL_0 consist of

- All (or enough) tautologies of classical propositional modal logic (over the language $\mathcal{L}_{PDL}(\Sigma)$)
- The (K) axiom for each modality $[\sigma]$[38]
- The inference rule of *Modus Ponens*
- For each modality $[\sigma]$, the inference rule of *Necessitation*: from φ, infer $[\sigma]\,\varphi$.

A **theorem** of PDL_0 is an $\mathcal{L}_{PDL}(\Sigma)$ formula which may be proved from the axioms and rules of deduction of PDL_0, analogously to how we defined for ADTL above.

Proposition 1.3

PDL_0 is a sound and complete axiomatization of $\mathcal{L}_{PDL}(\Sigma)$ with respect to the class of Σ-relational models.

Showing Prop. 1.3 is a standard proof in modal logic, utilizing a canonical model to refute nontheorems of PDL_0.[39] However, it's important to note that this proof relies on a particular assumption: that there are no relationships "baked in" between the different relations. Notice that PDL_0 treats the modalities $[\sigma]$ and $[\sigma']$ entirely independently: there are no axioms governing how the truth conditions of $[\sigma]$ and $[\sigma']$ interact, because we interpret them by completely arbitrary relations R_σ,$R_{\sigma'}$, and thus there *isn't* any particular structural relationship between R_σ and $R_{\sigma'}$.[40]

[36] A reader unfamiliar with PDL is encouraged to read [16] for more background. We use similar-enough notation (and explicitly note our use of it enough) that no great confusion should arise.

[37] It is due to this notion that we refer to "*relational* PDL" as such. We're ultimately working towards a contrast with *dynamic-topological PDL*, which is what we seek to introduce.

[38] Replace the $\Box$ in the statement of (K) in Defn. 1.4 with each $[\sigma]$.

[39] This is a generalization of [7, Theorem 4.23]. See the proof of Theorem 3.7, Claim 1 for more on this.

[40] There are certain relationships between the interpreting relations that we might be interested in (e.g. that R_σ is always interpreted to be a subset of $R_{\sigma'}$), and, as any standard account of modal logic will explore, there are certain relationships that can be formally stated in the object language $\mathcal{L}_{PDL}(\Sigma)$. We can impose on our models the requirement that $R_\sigma \subseteq R_{\sigma'}$, but this will lead to additional formulas being validated (in this instance: requiring

Relational PDL has a particular manner for imposing additional structure, which will become our central theme: **program construction**. So far (and particularly for Prop. 1.3), we've treated the set Σ of "program names" as some undifferentiated set of primitives with no internal structure. But PDL does assume that Σ has a particular form. Specifically, the set Σ of programs used in standard PDL is given as the closure of a set of primitives under several connectives. Relational PDL includes 3 ways for combining program symbols:[41]

- *Sequencing*: given programs σ_0,σ_1, form the program $\sigma_0;\sigma_1$ consisting of "do σ_0, then do σ_1".
- *Nondeterministic union*: executing the program $\sigma_0 \cup \sigma_1$ consists of nondeterministically selecting either σ_0 or σ_1 and performing that action (e.g. "flip a coin. If heads, do σ_0. If tails, do σ_1")
- *Nondeterministic iteration*: executing σ^* consists of performing the action σ some nondeterministic number of times (e.g. "generate a random natural number n, and then do σ n times")

Standard presentations of PDL also include a way of taking a formula φ and forming the "test program" $\varphi?$ which "asserts" φ. Together, these form the standard core of PDL, and particular interpretation is given to these "constructed programs". In this section, we'll elaborate the relational semantics for nondeterministic union, before we go on to give dynamic-topological semantics. A similar development of sequencing and test programs is given in [5], and nonterministic iteration will be beyond the scope of our work here.[42]

Definition 1.6

Let Π be some set of primitive program symbols. The set $\Pi^{\cup}$ is given to be the closure of Π under combination using $\cup$:

$$\sigma ::= \pi \mid \sigma_0 \cup \sigma_1. \qquad (\pi \in \Pi)$$

The language $\mathcal{L}_{PDL}(\Pi^{\cup})$ (also written as just $\mathcal{L}^{\cup}_{PDL}$ when Π is understood) is given by:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid [\sigma]\varphi. \qquad (p \in \Phi,\ \sigma \in \Pi^{\cup})$$

We refer to the programs denoted by elements of Π as "**primitive programs**", and programs denoted by any other element of $\Pi^{\cup}$ as "**constructed programs**".

More generally, if c is some n-ary function symbol, then we write Π^c to denote the closure of Π under combination with c, i.e. the least Σ such that $\Pi \subseteq \Sigma$ and

$$\Sigma = \Sigma \cup \{c(\sigma_1, \ldots, \sigma_n) \ : \ \sigma_1, \ldots, \sigma_n \in \Sigma\}.$$

Accordingly, we'll write $\mathcal{L}^c_{PDL}$ to denote the PDL language over this program set, and $\mathcal{L}^c_{\Box\bigcirc}$ for the corresponding agent DTL language.

$R_\sigma \subseteq R_{\sigma'}$ makes $[\sigma']\varphi \to [\sigma]\varphi$ a tautology). Such formulas are not theorems of PDL_0, so our completeness result will not continue to hold for the class of models where we've made these kinds of restrictions (PDL_0 is **not** complete with respect to the class of "Σ-relational models where $R_\sigma \subseteq R_{\sigma'}$"). So if we want our relational models to possess a richer structure, but also want to maintain the completeness of our deductive calculus, then we'll need to add corresponding axioms to PDL_0. This kind of tension between making structural requirements and maintaining completeness is ubiquitous in mathematical logic.

[41] These connectives bear strong relation (and resemblance) to standard operations on regular expressions. This is, of course, no accident. For this reason, the term "regular" is often used in reference to program sets of this form. We won't make use of this terminology (or connection), but it is worth noting.

[42] Though we give semantics for sequencing and nondeterministic iteration as program constructors in Subsect. 2.3.

This definition showcases how we'll use the notation to keep track of our languages: the subscript indicates what modalities are operating on the set of *formulas* (either PDL for the PDL modalities $[-]$ and $\langle - \rangle$, or $\Box\bigcirc$ to indicate we're working with $\Box$, $\Diamond$, and many $\bigcirc$s) while the superscript indicates the available connectives among *programs* (in this case: just the binary connective $\cup$).

So we have some syntactic relationships among the program names: if σ and σ' are program names, then there's a program name $\sigma \cup \sigma'$, $\sigma \cup (\sigma' \cup \sigma)$, and so on. But in order for the syntax to have meaning, the semantics must back up these relationships. As mentioned above, in an arbitrary $\Pi^{\cup}$-relational model there is no particular relationship between R_{σ_0}, R_{σ_1}, and $R_{\sigma_0 \cup \sigma_1}$. But we have an intended meaning for $\sigma_0 \cup \sigma_1$ – the nondeterministic union of σ_0 and σ_1 – and we desire the semantics of $\sigma_0 \cup \sigma_1$ to reflect this. So we turn our attention to what we'll call *union-models*.

Definition 1.7

Let Π be fixed. A **union-model** M is a $\Pi^{\cup}$ relational model $(X, \{R_\sigma\}_{\sigma \in \Pi^{\cup}}, v)$ such that

$$(x, x') \in R_{\sigma_0 \cup \sigma_1} \quad \Longleftrightarrow \quad (x, x') \in R_{\sigma_0} \text{ or } (x, x') \in R_{\sigma_1}$$

This condition is phrased more compactly as $R_{\sigma_0 \cup \sigma_1} = R_{\sigma_0} \cup R_{\sigma_1}$.

Defn. 1.7 implements nondeterministic union, but in a different sense of the word "nondeterministic" than the richer notion of nondeterminism we developed above. In relational PDL, $R_\sigma(x)$ is the set of all worlds which *could* be an outcome of σ at x. So if there are multiple possible outcomes, then the outcome is "nondeterministic". So relational PDL gives semantics for nondeterministic union accordingly: the outcomes of $\sigma_0 \cup \sigma_1$ are all the outcomes of σ_0 and all the outcomes of σ_1. This notion of "nondeterminism" (there being multiple possible outcomes) doesn't have any more elaborate internal structure, whereas our epistemic notion of nondeterminism arises from a subtle interaction between the agent's knowledge and the possibilities. We'll see, ultimately, that the two can mirror each other nicely.

So there are two perspectives we could take on Defn. 1.7. One perspective – the one suggested by the phrasing of the definition itself – is that we may view union-models as special $\Pi^{\cup}$-relational models, namely the ones whose interpretations of constructed programs obey the above-specified semantic relationship between R_{σ_0}, R_{σ_1} and $R_{\sigma_0 \cup \sigma_1}$ for arbitrary σ_0, σ_1. But we may also view it in this way: in order to define a union-model, it suffices to supply a Π-relational model. The other programs, the constructed programs, have their interpretations completely determined by the above. So we could even imagine a model transformation sending a Π-relational model M to the unique union-model M^{Union} "induced" by M, i.e. the union-model with the same set of worlds, same valuation, and same interpretation of Π programs as M, but which additionally interprets $\Pi^{\cup}$ programs via the *definition* $R_{\sigma_0 \cup \sigma_1} := R_{\sigma_0} \cup R_{\sigma_1}$. The main thing we wish to note here is the overall method, namely the semantic "transformation" turning models interpreting Π into models interpreting $\Pi^{\cup}$ by "copying" the primitive interpreations and then utilizing structural recursion to interpret constructed programs. Later on, this will be our central approach, though it will certainly prove more difficult in the dynamic-topological setting.

Now that we've given semantics for interpreting $\cup$-programs, we conclude our discussion of relational PDL by extending our PDL_0 axioms to axiomatize union-models. As mentioned above, imposing structural constraints (like $R_{\sigma_0 \cup \sigma_1} = R_{\sigma_0} \cup R_{\sigma_1}$) cause additional formulas – not provable from PDL_0– to be validated. In this case, those formulas are instances of the (U) axiom scheme.

Definition 1.8

Let (U) be the axiom scheme

$$[\sigma_0 \cup \sigma_1]\,\varphi \leftrightarrow [\sigma_0]\,\varphi \wedge [\sigma_1]\,\varphi,$$

where, unless stated otherwise, φ ranges over $\mathcal{L}^{\cup}_{PDL}$ and the σs over $\Pi^{\cup}$. We'll also use in its equivalent "dual" form:

$$\langle \sigma_0 \cup \sigma_1 \rangle \varphi \leftrightarrow \langle \sigma_0 \rangle \varphi \vee \langle \sigma_1 \rangle \varphi.$$

Proposition 1.4

The axiom scheme (U) defines the class of "union-frames": given a pair $(X, \{R_\sigma\}_{\sigma \in \Pi^{\cup}})$ where X is a set and $R_\sigma \subseteq X \times X$ for all $\sigma \in \Pi^{\cup}$,[43] the following are equivalent:

1. $R_{\sigma_0 \cup \sigma_1} = R_{\sigma_0} \cup R_{\sigma_1}$ for all $\sigma_0, \sigma_1 \in \Pi^{\cup}$
2. For all valuation functions $v : \Phi \to \mathcal{P}(X)$,

$$(X, \{R_\sigma\}, v) \models [\sigma_0 \cup \sigma_1]\varphi \leftrightarrow [\sigma_0]\varphi \wedge [\sigma_1]\varphi$$

for all $\sigma_0, \sigma_1 \in \Pi^{\cup}$ and all $\varphi \in \mathcal{L}^{\cup}_{PDL}(\Pi)$.

In summary: we have a process for taking arbitrary Π-relational models (representing the interpretations of primitive programs on the state space), and endowing them with a new construction, nondeterministic union. This construction is enacted both *syntactically* (by closing Π under $\cup$ to get $\Pi^{\cup}$) and *semantically* (by putting $R_{\sigma_0 \cup \sigma_1} = R_{\sigma_0} \cup R_{\sigma_1}$). The relationship between the syntactic and semantic extensions is governed by the (U) axiom scheme: (U) picks out precisely those $\Pi^{\cup}$-relational structures (i.e. those frames interpreting the extended syntax) which arise via the semantic extension.

We'll often refer to (U) as the "classical PDL axiom for $\cup$" (or variations of this phrase), to distinguish it from the axiomatization our *dynamic topological* interpretation of PDL, to which we now turn our attention.

1.7 Motivation: Dynamic Topolocal Program Construction

Let's now make the connection between PDL and DTL explicit. We said earlier that the PDL modalities were designed to reason about the behavior of computer programs. As Fischer and Ladner explain in the introduction to the original presentation of PDL[9], the intended interpretation of $\langle \sigma \rangle \varphi$ is that "σ ***can*** terminate with φ holding upon termination" (emphasis added). The word "can" here refers to the possible nondeterminism of the program σ: there could be a variety of possible outcomes to σ, but at least one of them is a φ-world. In the relational semantics, this is our notion of nondeterminism: R_σ is a relation (and is not required to constitute a partial function), so for each world x and each program σ, there are perhaps several worlds x' such that $xR_\sigma x'$. So it is taken as primitive that there are multiple possible outcomes to any given action, and our notion of nondeterminism does not possess any further logical structure.

But we have a more sophisticated notion of nondeterminism at play: the epistemic interpretation of nondeterminism. If σ is "nondeterministic" under this account, that didn't mean that there were *actually* multiple possible worlds that could result from executing σ from x. Indeed, we assumed for our models that there was some fact of the matter: either σ is not defined at x, or else there is some unique x' which is *the* world resulting from executing σ at x. Semantically, this corresponded to the stipulation that $f_\sigma(x)$ is a *partial function*, not just an arbitrary relation. The notion of nondeterminism, recall, arose from the inability of an agent to know beforehand what properties hold of the world resulting from executing σ.

In terms of the formal language: suppose both of the formulas $\Diamond\bigcirc_\sigma\varphi$ and $\Diamond\bigcirc\neg\varphi$ are true at x. What this says is that the agent **cannot** come to know whether (or not) φ will be true after the

[43] We'll call such a pair $(X, \{R_\sigma\})$ a "$\Pi^{\cup}$ relational frame".

execution of σ, at least not prior to actually executing σ. This is rather analogous to the situation in relational PDL where a single world x would be related to (at least) two distinct other woulds x_0 and x_1 by R_σ (i.e. $xR_\sigma x_0$ and $xR_\sigma x_1$) but x_0 and x_1 disagree on whether φ is true. If $\Diamond\bigcirc_\sigma\varphi$ and $\Diamond\bigcirc\neg\varphi$ both hold, then the agent, regardless of what state of knowledge they're in, must count it as a possibility that executing σ would result in a φ world, and also as a possibility that executing σ would result in a $\neg\varphi$ world. In this way, the formula $\Diamond\bigcirc_\sigma\varphi$ plays a similar role as $\langle\sigma\rangle\varphi$: it expresses the possibility that executing σ would result in a φ world ("possibility" in the primitive sense for PDL, and "possibility, as far as the agent can know" for epistemic DTL).

We can make this connection between $\langle\sigma\rangle$ and $\Diamond\bigcirc_\sigma$ official: view $\mathcal{L}_{PDL}(\Sigma)$ as a fragment of $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ by *defining* $\langle\sigma\rangle\varphi \equiv \Diamond\bigcirc_\sigma\varphi$. Then, it becomes possible to interpret $\mathcal{L}_{PDL}(\Sigma)$ on Σ-DTMs. What formulas of $\mathcal{L}_{PDL}(\Sigma)$ are validated by Σ-DTMs? Well, the answer is not surprising.[44]

Proposition 1.5

For any set Π, PDL_0 is a sound and complete axiomatization of $\mathcal{L}_{PDL}(\Pi)$ with respect to the class of all Π-DTMs: for all $\varphi \in \mathcal{L}_{PDL}(\Pi)$,

$$\vdash_{\mathsf{PDL}_0} \varphi \quad \Longleftrightarrow \quad \mathfrak{M} \models \varphi \text{ for all } \Pi\text{-DTMs } \mathfrak{M}$$

The same axioms which fully captured the logic of Σ-relational models fully captures the logic (that can be expressed in the more limited language $\mathcal{L}_{PDL}(\Sigma)$) of Σ-DTMs. One of the main results of [5] was to articulate the connection between the primitive PDL theories of relational PDL models and dynamic topological models regarded as models of the PDL language given in Prop. 1.5.

With the connection between relational PDL and agent DTL in mind, let us mention program construction again . Suppose we had a Π-relational model M in which $xR_{\pi_0}x_0$ and $xR_{\pi_1}x_1$, where $x_0 \neq x_1$. If we extend M to a union-model M^{Union} by the definition $R_{\pi_0\cup\pi_1} = R_{\pi_0} \cup R_{\pi_1}$, then

$$xR_{\pi_0\cup\pi_1}x_0 \qquad \text{and} \qquad xR_{\pi_0\cup\pi_1}x_1$$

Hence why "nondeterministic union" is so named: even if π_0 and π_1 were "deterministic" (in the sense of having at most one possible outcome world when executed from x), their union, $\pi_0 \cup \pi_1$ is nondeterministic (in the sense of having multiple possible outcomes). Furthermore, consider such a case where x_0 and x_1 are not only distinct, but validate different $\mathcal{L}_{PDL}(\Pi)$ formulas as well. Without loss of generality, say,

$$(M, x_0) \models \varphi \qquad \text{and} \qquad (M, x_1) \not\models \varphi.$$

And therefore we get that $(M, x) \models \langle\pi_0 \cup \pi_1\rangle\varphi$ and $(M, x) \models \langle\pi_0 \cup \pi_1\rangle\neg\varphi$.

Now to our key question: how can we carry out program construction in the domain of dynamic topological logic? If I have an arbitrary Π-DTM $\mathfrak{M}$, is there a way to transform it to a $\Pi^\cup$-DTM $\mathfrak{M}^{\mathsf{Union}}$ where the interpretation of $\pi_0 \cup \pi_1$ in $\mathfrak{M}^{\mathsf{Union}}$ bears the proper relationship to the interpretations of σ_0 and σ_1? Specifically, if $(\mathfrak{M}^{\mathsf{Union}}, w) \models \langle\sigma_0\rangle\varphi$ and $(\mathfrak{M}^{\mathsf{Union}}, w) \models \langle\sigma_1\rangle\neg\varphi$, can we have it so that

$$(\mathfrak{M}^{\mathsf{Union}}, w) \models \langle\sigma_0 \cup \sigma_1\rangle\varphi \wedge \langle\sigma_0 \cup \sigma_1\rangle\neg\varphi,$$

and thereby faithfully mimic relational PDL's nondeterministic union in agent DTL?

We saw with relational models that we could define M^{Union} to have the same state space and $\{R_\pi\}_{\pi\in\Pi}$ as M, and could add interpretations for constructed programs via $R_{\sigma_0\cup\sigma_1} = R_{\sigma_0} \cup R_{\sigma_1}$. But we cannot just copy this definition DTL: if we want $\|\sigma_0 \cup \sigma_1\|_{\mathfrak{M}^{\mathsf{Union}}}$ to be a partial function

[44] A version of Prop. 1.5 – with the additional assumption of *seriality* – is proved as [5, Thm. 2]. The completeness proof presented here is a novel one, more in line with the implementation of the program constructor $\mathsf{U}\infty$ in Subsect. 2.3.

(which is part of the definition of what it means for $\mathfrak{M}^{\mathsf{Union}}$ to constitute a $\Pi^{\cup}$-DTM), then it does not suffice to put

$$\|\sigma_0 \cup \sigma_1\|_{\mathfrak{M}^{\mathsf{Union}}} = \|\sigma_0\|_{\mathfrak{M}} \cup \|\sigma_1\|_{\mathfrak{M}} \,.$$

Rather, we must encode the nondeterminism in a more subtle way: by structuring the topology (and with it, the agent's epistemic capabilities) so that the interpretations of $\sigma_0 \cup \sigma_1$ can be discontinuous (nondeterministic) even when the interpretations of σ_0 and σ_1 are continuous (deterministic). This turns out to be quite an involved task, and will occupy our attention for the remainder of this work.

Chapter 1

Basic Theory of Program Constructors

To inform what kind of *mathematical* setup is required to enrich dynamic topological models with nondeterministic program constructors, it is helpful to consider the *philosophical* meaning of such a procedure. If a DTM can be interpreted as modeling a dynamic epistemic situation, then what kind of scenario is modeled by a DTM endowed with program constructors? What does it mean if the agent is able to execute actions of the form $\sigma_0 \cup \sigma_1$? We take up this question now, and we'll find that the answer will lead us to the abstract definition of "program constructor" for dynamic topological models.

As mentioned above, nondeterministic union will be our primary example. However, the definition we provide will be much more general, and we provide several other examples. Our goal here is not simply to introduce a way to define nondeterministic union in the dynamic-topological setting, but furthermore to use this task as an occasion to significantly develop the theory of DTL. To this end, the present chapter (as well as Chapter 2) will be presented in a greater level of generality than is strictly necessary, and will be accompanied with enough mathematical theory to facilitate this more general view. We develop the specific details for nondeterministic union (and a simplified version of union we'll call "OR"), and leave similar developments of other program constructors to future work.

2 Program Constructors

2.1 Intuition

To begin, consider the following situation.[1]

> It's December 14th, 1903, and two brothers, Wilbur and Orville Wright, are on the cusp of becoming the first people to successfully build and fly an airplane.[2] They have been working for years on their prototype flying craft, and today might be the day they make history. To conduct a test flight, one of them needs to be in the plane and one of them on the ground. If today's flight is successful, then whichever brother was in the plane that day will forever be *the first person ever to fly a plane*, whereas the other brother will merely watch history's first flight from the ground (despite his roughly-equal contribution to the effort). Unable to decide which of them gets to (maybe) become the

[1] Taken (with some artistic license) from [1] (www.nps.gov/wrbr/learn/historyculture/thefirstflight.htm).
[2] That's heavier than air, self-propelled, capable of making sustained flights, etc.

> first pilot in human history, they flip a coin. Wilbur wins the toss, and gets to operate the craft on today's test flight.

This is a story of human utilization of technology, in two ways. In addition to the obvious use of technology by the Wright brothers (using tools and machines to fashion a flying contraption), they are utilizing a much older, much more familiar technology: a coin flip. As we'll see, the latter is actually much more curious: the brothers use hammers to manipulate metal, knives and saws to manipulate wood, but the coin they use to manipulate *their own ability to predict the future*.

Part of the purpose of a coin flip is to produce an arbitrary result, with no "reasoning" behind the outcome. If I'm flipping a coin to resolve a binary decision, I'm doing so because I've exhuasted other considerations for resolving the question and am looking for the decision to be made for me. But, for some applications of coin flips (e.g. when a hotly-contested competition ends in a tie, and the coin is brought in to break the tie), we demand something more: the decision should be arbitrary. If a referee tosses a coin to decide the winner of a deadlocked game, then nobody can (reasonably) question whether the referee's personal biases in favor of one team affected the decision: everyone understands that basic precautions (e.g. seeing that the coin comes from an unbiased source, allowing the teams to inspect the coin, conducting the flip for all to watch, etc.) make it virtually impossible to rig a coin flip. Therefore, even if the referee wanted to fix the coin flip in favor of one team or the other, these kinds of constraints seem to make it impossible to execute such a fix. The result is arbitrary. Clarifying this notion of "arbitrariness" in its totality (and related questions of whether it's possible to "fix" a given result to a process) will lead us to much more complex questions about causality, counterfactuals, and the like, which we don't want to get into here. But we can articulate *epistemic nondeterminism* as at least one (apparently necessary) aspect of a properly-conducted coin flip: at the very least, nobody could have known what the result would be before the coin was flipped. If someone was able to possess such knowledge (e.g. the referee was able to *know for a fact* that the coin would come up heads), then clearly something is afoot. Thus, while epistemic nondeterminism (viewed as a property of the coin-flipper's capacity for knowledge in such a situation) does not appear to be a *sufficient* characterization of what makes a coin flip a coin flip, it is a *necessary* property.

To see this, let's return to the example of the Wright brothers. If either brother knew enough to predict the outcome of the flip (the flip deciding which brother flies today), then the entire exercise would be effectively pointless. For instance, let's say Wilbur tossed the coin, but gave his brother Orville the pick of "heads" or "tails". If Orville somehow knew in advance which way the coin would come up, [3] then the intention behind flipping the coin – arbitrarily deciding which of them gets to fly today – is silently defeated: Orville must have fixed the decision somehow (how else would he *know* the result beforehand?), and it is absolutely not arbitrary. On the other hand, if *Wilbur* knows the outcome in advance,[4] the decision is again not arbitrary. In neither of these cases are the brothers faithful to their original impetus for the coin flip. And certainly if the brothers find themselves in some complex epistemic setup atop these more basic "foreknowledge" situations,[5] then the flip itself is still pointless. The coin has its intended function ***only if*** it creates a particular arrangement of knowledge, specifically one where neither brother is able to predict the outcome of the coin flip.

The previous paragraph could be more succinctly expressed as: *coin flips are supposed to be epistemically nondeterministic*. It's worth emphasizing that it's *epistemic* nondeterminism at play

[3] Say, he anticipated the situation and quietly substituted Wilbur's fair coin with a coin with two "heads"

[4] Suppose he is cheating with the coins, e.g. he has a two-headed coin and atwo-tailed coin, and will flip one based on whether Orville calls heads or tails, thereby allowing him to ensure absolutely that he'll win the toss

[5] E.g. Wilbur knows that Orville knows that Wilbur knows the outcome of the flip in advance, but Orville does not know that Wilbur knows that Orville knows that Wilbur knows the outcome.

here: it's certainly not clear that a coin flip is "nondeterministic" in any metaphysical sense,[6] or at least we don't need it to be. The above story about the brothers and their engagement with each other and with the coin makes total sense, even if the coin toss itself is "ultimately" deterministic. It presents no obstacle to the foregoing analysis if there is ultimately some fact of the matter about which side of the coin comes up; we just require that the brothers be ignorant of this fact. Or rather, we require that the brothers be *necessarily* ignorant about this: that they be *unable* to know which way the coin will come up.[7] Henceforth, when we refer to a "coin toss", we will allow (indeed, assume) that there is some ultimate fact of the matter about which way the coin comes up, but we will assume that the toss is "honest": either outcome is a possibility (from the epistemic standpoint of the agent in question), and the agent cannot know in advance which will transpire.

Let's talk about this using the agents-in-situations language from before. We'll understand "decisions by coin flip" in the following manner: our agent[8] will always be deciding between two *courses of action*. More precisely, our agent will be considering two actions σ_0 and σ_1. Then coin flipping manifests as a binary operation *on actions*: if our agent has access to a coin, then she is allowed to instead perform the action $\sigma_0 \cup \sigma_1$, which consists of "flip a coin. If it comes up heads, do σ_0. If tails, σ_1". For our purposes, we do not allow the agent to 'disobey' the coin: we do not allow her to, say, flip the coin and then decide to ignore it and do σ_0 anyways, or forget the whole thing and default to some unrelated σ_2, or various other modifications. I see no reason that this more complex behavior *couldn't* be modelled in (some modification of) the framework we'll develop, but we'll exclude this for simplicity. When the agent performs $\sigma_0 \cup \sigma_1$, she flips the coin and faithfully carries out either σ_0 or σ_1, according to the outcome of the flip.

The introduction of a nondeterministic device (in this case, the coin) allows the agent to nondeterministically combine actions. She can take any actions (deterministic or not) σ_0 and σ_1, and form the action $\sigma_0 \cup \sigma_1$. The resulting action $\sigma_0 \cup \sigma_1$ will be nondeterministic: the agent doesn't *and cannot* know whether the flip will come up heads or tails, at least until she does this flip. Consequently, she cannot know in advance whether $\sigma_0 \cup \sigma_1$ will end up being σ_0 or σ_1, and thus cannot know exactly which outcomes to expect from this compound action (except in degenerate cases, e.g. $\sigma_0 = \sigma_1$). $\sigma_0 \cup \sigma_1$ could end up being σ_0, and produce whatever results σ_0 produces, or it could be σ_1 and have σ_1's results. These are the main structural components to a coin-flipping situation: $\sigma_0 \cup \sigma_1$ ultimately consists of either σ_0 or σ_1, but there's no way to know which one in advance.

Our mathematical development of this idea will rely on the following observation. When a coin is flipped, there is some fact of the matter about which way the coin will come up: the coin is subject to physical laws which fully determine its outcome. But our agent is not in a position to *know* this outcome beforehand.[9] But, as the agent conceives of it, it's equally possible that the coin flip has

[6] One could perhaps tell a coherent story about how the macroscopic outcome of the coin flip is determined by the outcomes of a fantastic number of microscopic quantum events, and thereby be able to claim that there is no single, fixed outcome of the flip until it actually occurs. We don't make an attempt to evaluate such accounts.

[7] This is a simplifying assumption. A modification of the present account where there's more internal structure to the epistemology of the coin flip – structure the relevant agent(s) can know about and perhaps exploit – certainly sounds interesting, and could be fruitful further work.

[8] We'll restrict ourselves to considering a single agent, who is both the flipper of the coin *and* the one reasoning about possible outcomes. The ability of an agent (e.g. Orville) to reason about *another* agent (e.g. Wilbur) flipping a coin is an added layer of complexity which we don't address here.

[9] If the flip is conducted "honestly" (e.g. no double-headed coin), it would be tremendously hard to predict its outcome and would undoubtedly require special equipment. Perhaps if the agent had measured all the minute differences in density of the coin and had set up a machine to measure the coin's height, velocity, angular momentum, etc. at the moment of the flip, then perhaps it could crunch the numbers fast enough (e.g. using Newtonian mechanics) to predict the outcome. This is just to further emphasize the point that our notion of nondeterminism is dependent on the agent's epistemic abilities: if the agent has access to such a coin-flip-predictor machine, then coin flipping is indeed deterministic. We assume they do not.

the opposite outcome. We make sense of this by stipulating two worlds of the situation (call them w and w') which are identical in every respect, except for the outcome of the next coin flip performed by the agent. More precisely, $\sigma_0 \cup \sigma_1$ will end up being σ_0 in w but σ_1 in w', but this is the only difference. In particular, there's no observation the agent could make to indicate they're in w but not in w' (or vice versa). So when we say that "the agent cannot know whether the flip will come up heads or tails", this just means that they find themselves in such an w or w' (but can't know which). We might say that w is a "0-world" and w' a "1-world", since a coin flip comes up heads in w (tails in w'), so executing an action of the form $\sigma_0 \cup \sigma_1$ gets interpreted to σ_0 (respectively, σ_1). Accordingly, interpreting $\sigma_0 \cup \sigma_1$ in DTMs will consist of making "duplicates" of worlds, and making one these twin worlds a 0-world and the other a 1-world, and rigging up the topology to encode the fact that the agent can't tell them apart. Indeed, we will need to further multiply the state space, in particular to account for multi-flip "nested" actions, e.g. $(\sigma_0 \cup \sigma_1) \cup (\sigma_2 \cup \sigma_3)$ and the like.

One final comment we'll add to this is that $\sigma_0 \cup \sigma_1$ should be at least as epistemically nondeterministic as σ_0 and σ_1, which are themselves allowed to be nondeterministic. Returning to the Wright brothers example, it turns out that December 14th, 1903 *wasn't* the day. On that day, Wilbur won the toss but the flight was not a success. So Orville, the loser of the coin toss, eventually became history's first pilot: after the brothers repaired the craft, it was Orville's turn to fly on December 17th. *That* was the historic flight. So consider everything from Wilbur's perspective: let σ_0 be the action "attempt to fly the craft on Dec. 14", σ_1 be the action "stay on the ground and help your brother fly on Dec. 14", and let φ be the proposition that Wilbur is the first pilot in human history. Then Wilbur did indeed execute $\sigma_0 \cup \sigma_1$. The fact of the matter was that the coin came up heads, so Wilbur did σ_0 (flew on December 14). Unbeknownst to him, the outcome was him *not* becoming history's first pilot. But this wasn't ultimately because of the coin flip: he won the coin flip! Wilbur wanted to do σ_0 because he was ignorant of σ_0's impacts: he thought "after σ_0, φ" held (he thought that flying on December 14th would make him the first pilot in human history). But actually, it was the case that after σ_0, $\neg\varphi$. So, under this reading, the following formulas are validated:

$\bigcirc_{\sigma_0}\neg\varphi$	(Flying on Dec. 14 won't make Wilbur the first pilot)
$\Diamond \bigcirc_{\sigma_0} \varphi$	(As far as Wilbur could know, flying on Dec. 14 *could* make him the first pilot)
$\bigcirc_{\sigma_0}\psi \to \Diamond \bigcirc_{\sigma_0\cup\sigma_1} \psi$	(The coin could come up heads)
$\bigcirc_{\sigma_1}\psi \to \Diamond \bigcirc_{\sigma_0\cup\sigma_1} \psi$	(The coin could come up tails)
$\bigcirc_{\sigma_0}\psi \to \bigcirc_{\sigma_0\cup\sigma_1}\psi$	(The coin *did* come up heads.)

The latter three formulas include a metavariable ψ which could be any proposition. So, for instance, we labelled $\bigcirc_{\sigma_0}\psi \to \Diamond \bigcirc_{\sigma_0\cup\sigma_1} \psi$ as "The coin could come up heads" because the possibility of heads (as far as Wilbur can know) means that the actual result of σ_0 is a possible result of $\sigma_0 \cup \sigma_1$ (again, as far as Wilbur can know). And so if ψ is true upon the execution of σ_0 (whatever ψ is), then, as far as Wilbur can know, ψ could be true upon the execution of $\sigma_0 \cup \sigma_1$. Likewise for the formula below it. So the conclusion is that there's some epistemic nondeterminism due to the coin flip itself, but the actions being union-ed together may themselves be nondeterministic. With that, we've established all the requisite intuitions and can proceed to incorporate all these considerations into a mathematical framework.

2.2 Definition

In relational PDL, we saw that we could augment Π-relational models to $\Pi^{\cup}$-relational models (specifically, union-models) using the definition $R_{\pi_0 \cup \pi_1} = R_{\pi_0} \cup R_{\pi_1}$. The $\Pi^{\cup}$-relational model M^{Union} produced from $M = (X, \{R_\pi\}_{\pi \in \Pi}, v)$ via this definition had the same set of points, same valuation, and same interpretation of Π as M; it just additionally interpreted ∪-programs. We saw that the same cannot be done so easily with Π-DTMs: in order to produce a transformation sending a Π-DTM $\mathfrak{M}$ to a $\Pi^{\cup}$-DTM $\mathfrak{M}^{\mathsf{Union}}$ which interprets $\sigma_0 \cup \sigma_1$ as nondeterministic union, we need a more mathematically-elaborate setup.

As mentioned above, we will do this by duplicating the state space: in the coin-flip example, we had the worlds w and w' which were alike in every way, except for the result of a coin flip. Moreover, the nondeterminism of a coin flip was achieved by making these worlds indistinguishable. So if we want $\mathfrak{M}^{\mathsf{Union}}$ to consist of the same situation as $\mathfrak{M}$ but with a (nondeterministic) coin flip available, then we could (say) make the states of $\mathfrak{M}^{\mathsf{Union}}$ to be pairs (x, γ) where x is a world of $\mathfrak{M}$ (saying what $\mathfrak{M}$-state the agent is in), and γ is either 0 or 1, encoding the outcome of the next coin flip. If the agent undertakes an action which does not involve flipping the coin (i.e. an action denoted by an element of Π), then we ignore γ. But we can interpret $\sigma_0 \cup \sigma_1$ to be σ_0 in $(x, 0)$ and σ_1 in $(x, 1)$. From there, it's just a matter of ensuring that $(x, 0)$ and $(x, 1)$ are "indistinguishable" (topologically, and in terms of satisfying $\mathcal{L}_{\Box\bigcirc}(\Pi)$ formulas), which, recall, was crucial to thinking of this as a model of coin-flipping. Let us codify this process in much greater generality, and later develop this idea of $\mathfrak{M}^{\mathsf{Union}}$ as a specific instance.[10]

Definition 2.1 (Model-to-Model Program Constructor)
Let $n \in \mathbb{N}$ be given. An (n-ary) **program constructor** C interpreting an n-ary function symbol c consists of a topological space (Γ, τ_Γ) and a *rule*[11] which assigns to each Π-DTM $\mathfrak{M} = (X, \tau_X, \{f_\pi\}_{\pi \in \Pi}, V)$ a Π^c-DTM[12] $\mathfrak{M}^C$ such that

- $\left|\mathfrak{M}^C\right| = X \times \Gamma$: the states/worlds of $\mathfrak{M}^C$ are pairs (x, γ) for x a world of $\mathfrak{M}$ and $\gamma \in \Gamma$;
- the topology of $\mathfrak{M}^C$ is the product topology of τ_X and τ_Γ;
- primitive programs don't touch Γ: for all $\pi \in \Pi$, all $x \in X$, and all $\gamma \in \Gamma$,

$$\|\pi\|_{\mathfrak{M}^C}(x, \gamma) = (f_\pi(x), \gamma)$$

- and the valuation ignores Γ: for all $p \in \Phi$, all $x \in X$, and all $\gamma \in \Gamma$,

$$(x, \gamma) \in [\![p]\!]_{\mathfrak{M}^C} \quad \Longleftrightarrow \quad x \in [\![p]\!]_{\mathfrak{M}}.$$

We'll often refer to $\mathfrak{M}^C$ as the "C-augmentation" of $\mathfrak{M}$, and refer generically to "C-augmented DTMs". We'll often refer to Γ as the "state space of C" (and – especially if τ_Γ is the indiscrete topology – the "hidden state space"), and refer to elements $\gamma \in \Gamma$ as "constructor states" or "hidden states".

Let's indicate the function of each part of this definition, but instead of talking specifically of "coin flips" (and their symbolic stand-in, ∪), we'll articulate this in terms of more general

[10] We will actually have several "program constructors" which encode coin-flipping with binary digits like this. To avoid confusion, none of them will be denoted $\mathfrak{M}^{\mathsf{Union}}$.

[11] We could, if desired, explicitly define this "rule" as a class function on the proper class of all DTMs (or as a class function from the class of Π-DTMs into the class of Π^c-DTMs for some fixed Π). This might aid our understanding of the phenomenon of program construction in more formal detail, but I don't believe we rely on such "foundational" considerations in any substantive way to deliver any technical results.

[12] Recall Π^c is the closure of Π under combination with the n-ary function symbol c, as mentioned in Defn. 1.6.

"nondeterministic devices" which take several actions $\sigma_1, \ldots, \sigma_n$ as arguments, and combines them into some action $c(\sigma_1, \ldots, \sigma_n)$. What "doing $c(\sigma_1, \ldots, \sigma_n)$" consists of will depend (in general) on the interpretations of $\sigma_1, \ldots, \sigma_n$ in the present state, as well as properties of the present state (especially the Γ state). The reader may find it helpful to review some of the examples below in conjunction with this explanation, and see some basic ways we can instantiate this (admittedly quite abstract) definition.

- X is the state space of the original situation, and Γ is the private state space of the "device" we're introducing into this situation. So, as mentioned above, states in the "augmented" situation consist of pairs (x, γ): x says what state of the original scenario our agent is in, and γ indicates what state the device is in. This is why the state space of this new DTM is $X \times \Gamma$, the set of all such pairs.

  Throughout, we'll informally describe the set $X \times \Gamma$ as "Γ-many copies of each world of X" and describe (x, γ) as "a copy of x in $\mathfrak{M}^C$". This is to reflect our intuition that $\mathfrak{M}^C$ interprets the same situation as $\mathfrak{M}$ (only with constructed programs now also possible), and a world (x, γ) of $\mathfrak{M}^C$ represents the world x of $\mathfrak{M}$ where additionally the "device" is in state γ. We'll invest some effort to showing how the logic of augmented DTMs reflects this relationship.

- Likewise with the topology: the product topology of τ_X and τ_Γ is generated by open sets of the form $U \times V$, for $U \subseteq X$ open in τ_X and $V \subseteq \Gamma$ open in τ_Γ. Roughly, this corresponds to the assertion that "observations" in our augmented situation $\mathfrak{M}^C$ rule out some $\mathfrak{M}$-states $(X \setminus U)$ and rule out some Γ-states $(\Gamma \setminus V)$, in that the only worlds considered possible are those $(x, \gamma) \in X \times \Gamma$ where both $x \in U$ **and** $\gamma \in V$. This establishes the basis of the topology for our augmented situation, and then the preceding comments apply.

  Defn. 2.1 is stated to allow for program constructors to have varying levels of 'epistemic opacity', as measured by the coarseness of the topology. For the purposes of this work, we only consider program constructors with indiscrete topologies, i.e. $\tau_\Gamma = \{\emptyset, \Gamma\}$. This corresponds to the agent having *no* ability to reason about what the Γ-state is from within the scenario: no observation could rule out any Γ-state. In the coin-flip example above, we were working with $\Gamma = \{0, 1\}$, corresponding to the two possible outcomes of a flip. For such a program constructor, we use the indiscrete topology to indicate that the agent does not and *cannot* have any idea whether the Γ state is 0 or 1, i.e. whether the next flip will be a heads.

- The functions encode precisely the intuitions we had about how a device might utilize a private state space. If the current state is (x, γ) and we're seeking to execute $\pi \in \Pi$, then we don't need the private state space – we can execute π in the same manner as in the original DTM, and leave the private state untouched. However, if we need to execute a constructed program $c(\sigma_1, \ldots, \sigma_n)$, then this is precisely what the augmentation is for. The definition does not constrain at all what $\|c(\sigma_1, \ldots, \sigma_n\|_{\mathfrak{M}^C}$ can be: all that the definition requires is that there's *some* way of defining it[13]. But usually (as we'll see in the examples below) the interpretation of $c(\sigma_1, \ldots, \sigma_n)$ will depend systematically on the state and the interpretations of $\sigma_1, \ldots, \sigma_n$.

- Finally, the valuation, as noted, does not regard the Γ state, and decides the value of primitive propositions in terms of the $\mathfrak{M}$-state. In the examples we develop below, we don't want $\mathcal{L}_{PDL}$ formulas (and, in particular, primitive propositions) to be able to pick up hidden state variation. Later in this chapter (and in the next), we shall pay a great deal of attention to this point.

[13] Which can, in general, depend on the DTM $\mathfrak{M}$ it's applied to, the $\mathfrak{M}$-state, the Γ-state, and the arguments $\sigma_1, \ldots, \sigma_n$ in arbitrary – and perhaps pathological – ways.

The point of this definition is not to constrain too much the class of objects we're considering. Undoubtedly, the definition of what constitutes a "program constructor" is sufficiently broad to admit all manner of bizarre cases and poorly-behaved instances (much like other such abstract definitions, e.g. of a topological space). Accordingly, we will be limited in how much we can say about program constructors in full generality (although, as the following development will hopefully show, there is a good deal we can say). The intention with this statement is rather to establish a basic framework and terminology for approaching this topic. The real interest in our analysis will be the class of special cases we begin to develop – my hope (and suspicion) is that carving out special classes and compelling examples of program constructors will prove to be a quite fruitful avenue of inquiry. Let us initiate this study by laying out the examples which we'll focus on for the remainder of the present work.

2.3 Examples

We devote the rest of this section to listing several examples. We only will develop a few of these examples in greater detail, but likely each of them would prove interesting if examined further. Throughout, let $\mathfrak{M} = (X, \tau_X, \{f_\pi\}, V)$ be an arbitrary Π-DTM.

Example 2.1

The 0-ary program constructor SKIP (augmenting Π-DTMs to Π^{skip}-DTMS, where $\Pi^{\mathsf{skip}} = \Pi \cup \{\mathsf{skip}\}$) has trivial state space ($\Gamma = \{\star\}$, $\tau_\Gamma = \{\emptyset, \Gamma\}$), and interprets:

$$\|\mathsf{skip}\|_{\mathfrak{M}^{\mathsf{SKIP}}}(x, \star) = (x, \star)$$

SKIP is a *deterministic* program constructor, because all constructed programs (i.e. just skip itself) are deterministic if their arguments are (skip takes 0 program arguments). We often suppress the unnecessary $\star$ parameter, and just act as if $\mathfrak{M}$ and $\mathfrak{M}^{\mathsf{SKIP}}$ have the same state space (in which case $\|\mathsf{skip}\|_{\mathfrak{M}^{\mathsf{SKIP}}}(x) = x$).

For another program constructor C, we might write $C + \mathsf{SKIP}$ to denote the program constructor which behaves exactly like C (and has the corresponding state space and such), but produces a $\Pi^c \cup \{\mathsf{skip}\}$-DTM instead of a Π^c-DTM, and interprets skip as the identity function.[14]

Example 2.2

Pulling from [5, Section 4], we introduce the program constructor SEQ of *sequencing*. We write Π^{seq} for the program set given by the grammar

$$\sigma ::= \pi \mid \sigma; \sigma'$$

The program $\sigma; \sigma'$ is pronounced "σ, *then* σ'". As with SKIP, SEQ is a deterministic program constructor (in the sense of not needing a Γ state space), so we can formally put $\Gamma = \{\star\}$ but informally pretend that $\mathfrak{M}^{\mathsf{SEQ}}$ and $\mathfrak{M}$ have the same state space. For any $\mathfrak{M}$ and any $x \in |\mathfrak{M}|$,

$$\left\|\sigma; \sigma'\right\|_{\mathfrak{M}^{\mathsf{SEQ}}}(x) = (\left\|\sigma'\right\|_{\mathfrak{M}^{\mathsf{SEQ}}} \circ \|\sigma\|_{\mathfrak{M}^{\mathsf{SEQ}}})(x)$$

Note the order of composition is such that we do σ first, then σ'. If either $\|\sigma\|_{\mathfrak{M}^{\mathsf{SEQ}}}(x)$ is undefined, or $\|\sigma'\|_{\mathfrak{M}^{\mathsf{SEQ}}}(\|\sigma\|_{\mathfrak{M}^{\mathsf{SEQ}}}(x))$ is undefined, then $\|\sigma; \sigma'\|$ is undefined at x.

Example 2.3

There are two nondeterministic union program constructors, $\mathsf{U}\omega$ and $\mathsf{U}\infty$, which augment Π-DTMs to $\Pi^{\cup}$-DTMs.

$\mathsf{U}\omega$ is defined by

[14] This will come in handy later.

- $\Gamma = \{0,1\}^*$, the set of all finite-length strings of 0s and 1s. We write ϵ for the empty string and, where necessary, write $\gamma_1 ^\frown \gamma_2$ or $\gamma_1\gamma_2$ for the concatenation of two such strings.
- τ_Γ is the indiscrete topology, $\{\emptyset, \Gamma\}$
- In $\mathfrak{M}^{\mathsf{U}\omega}$, the constructed program $\sigma_0 \cup \sigma_1$ is interpreted by

$$\begin{aligned} &\|\sigma_0 \cup \sigma_1\|_{\mathfrak{M}^{\mathsf{U}\omega}}(x, \epsilon) \text{ is undefined} \\ &\|\sigma_0 \cup \sigma_1\|_{\mathfrak{M}^{\mathsf{U}\omega}}(x, 0s) = \|\sigma_0\|_{\mathfrak{M}^{\mathsf{U}\omega}}(x, s) \\ &\|\sigma_0 \cup \sigma_1\|_{\mathfrak{M}^{\mathsf{U}\omega}}(x, 1s) = \|\sigma_1\|_{\mathfrak{M}^{\mathsf{U}\omega}}(x, s) \end{aligned}$$

$\mathsf{U}\omega$ has the advantage of being more tractable (its state space Γ is countable, for instance, which will make it easier to capture its essential logic), but the presence of the ϵ (meaning that $\sigma_0 \cup \sigma_1$ could be undefined even if σ_0 and σ_1 are both defined) can strain our interpretation of $\cup$ as coin-flipping. This possibility perhaps makes more sense if we think of $\cup$ as "querying a source of random bits" (e.g. implemented on a computer), which could conceivably *run out of randomness*.[15]

The opposite choice of tradeoffs is made with $\mathsf{U}\infty$:

- $\Gamma = \{0,1\}^{\mathbb{N}}$, the set of all infinite-length strings of 0s and 1s.
- τ_Γ is the indiscrete topology, $\{\emptyset, \Gamma\}$
- In $\mathfrak{M}^{\mathsf{U}\infty}$, the constructed program $\sigma_0 \cup \sigma_1$ is interpreted by

$$\begin{aligned} &\|\sigma_0 \cup \sigma_1\|_{\mathfrak{M}^{\mathsf{U}\infty}}(x, 0s) = \|\sigma_0\|_{\mathfrak{M}^{\mathsf{U}\infty}}(x, s) \\ &\|\sigma_0 \cup \sigma_1\|_{\mathfrak{M}^{\mathsf{U}\infty}}(x, 1s) = \|\sigma_1\|_{\mathfrak{M}^{\mathsf{U}\infty}}(x, s) \end{aligned}$$

So we do not need an ϵ case – there are always enough bits (which makes things somewhat cleaner, mathematically and philosophically). The downside is that Γ is now uncountable, and will prove difficult to fully capture using a finitary modal logic.

Example 2.4

In addition to $\mathsf{U}\omega$ and $\mathsf{U}\infty$ (which interpret all of $\Pi^\cup$, including arbitrarily-nested $\cup$'s), we introduce a family of program constructors we call OR, which only allow for finite-nesting of $\cup$s. To remind ourselves of this, we use the symbol or instead of $\cup$.

More formally: for each $n \in \mathbb{N}$, we define the program set $\Pi^{\mathsf{or}\ n}$ by recursion as follows.

$$\begin{aligned} \Pi^{\mathsf{or}0} &= \Pi \\ \Pi^{\mathsf{or}(n+1)} &= \Pi^{\mathsf{or}\ n} \cup \left\{\sigma\ \mathsf{or}\ \sigma' \ :\ \sigma, \sigma' \in \Pi^{\mathsf{or}\ n}\right\} \end{aligned}$$

So, for instance, if $\pi, \pi', \pi'', \pi''' \in \Pi$, then $\Pi^{\mathsf{or}1}$ contains $\pi \,\mathsf{or}\, \pi'$ but does not contain $(\pi \,\mathsf{or}\, \pi') \,\mathsf{or}\, \pi''$. The latter *is* contained in $\Pi^{\mathsf{or}\ n}$ for $n \geq 2$, since the or's are doubly-nested. We'll simply write Π^{or} for $\Pi^{\mathsf{or}1}$.

The OR program constructor, which augments Π-DTMs to Π^{or}-DTMs, is given by:

- $\Gamma = \{0,1\}$,

[15] Or perhaps the agent attempts to flip the coin, but the flip itself fails (e.g. the agent doesn't catch the coin and it rolls into the sewer)?

- $\tau_\Gamma = \{\emptyset, \Gamma\}$ (the indiscrete topology),
- For primitive programs π_0, π_1,

$$\|\pi_0 \text{ or } \pi_1\|_{\mathfrak{M}^{\mathsf{OR}}}(x, 0) = \|\pi_0\|_{\mathfrak{M}^{\mathsf{OR}}}(x, 0)$$
$$\|\pi_0 \text{ or } \pi_1\|_{\mathfrak{M}^{\mathsf{OR}}}(x, 1) = \|\pi_1\|_{\mathfrak{M}^{\mathsf{OR}}}(x, 1)$$

Note that $\pi_0 \,\mathsf{or}\, \pi_1$ is always defined whenever π_0 and π_1 are (unlike $\mathsf{U}\omega$), but nesting of or's is disallowed, since we only have one "random bit" to resolve coin flips with.

We also define $\mathsf{OR1}, \mathsf{OR2}, \mathsf{OR3}, \ldots$, which allow for bounded nesting. For a set A and a natural number n, define $A^{\leq n}$ to be the set of all strings of elements of A, of length $\leq n$. For instance,

$$\{0,1\}^{\leq 2} = \{\epsilon, 0, 1, 00, 01, 10, 11\}$$

This allows us to define OR n for $n \geq 1$:

- $\Gamma = \{0,1\}^{\leq n}$
- $\tau_\Gamma = \{\emptyset, \Gamma\}$
- For any $\sigma_0, \sigma_1 \in \Pi^{\mathsf{or}(n-1)}$,

$$\|\sigma_0 \text{ or } \sigma_1\|(x, \epsilon) \text{ is undefined}$$
$$\|\sigma_0 \text{ or } \sigma_1\|(x, 0s) = \|\sigma_0\|(x, s)$$
$$\|\sigma_0 \text{ or } \sigma_1\|(x, 1s) = \|\sigma_1\|(x, s)$$

Note that OR and $\mathsf{OR1}$ are different: $\mathsf{OR1}$ also has ϵ as a Γ-state, and therefore $\|\sigma_0 \,\mathsf{or}\, \sigma_1\|_{\mathfrak{M}^{\mathsf{OR1}}}$ can be undefined even if $\|\sigma_0\|_{\mathfrak{M}^{\mathsf{OR1}}}$ and $\|\sigma_1\|_{\mathfrak{M}^{\mathsf{OR1}}}$ are defined, unlike with OR.

Example 2.5

We may define a unary program constructor STAR, which augments Π-DTMs to Π^{star}-DTMs, where Π^{star} is given by

$$\sigma ::= \pi \mid \sigma^*.$$

The intended interpretation of σ^* is "do σ a nondeterministic number of times", and is often known as "nondeterministic iteration" for this reason. Analogously to $\cup$, we can define several different program constructors giving semantics for this, with different properties about nesting. We'll define the ∞ version, $\mathsf{STAR}\infty$ here; the other versions ($\mathsf{STAR}\omega$, STAR, $\mathsf{STAR1}$, $\mathsf{STAR2}$, etc.) can be given analogously.

$\mathsf{STAR}\infty$ is defined by:

- $\Gamma = \mathbb{N}^{\mathbb{N}}$, the set of all (countably) infinite-length strings of natural numbers.
- τ_Γ is the indiscrete topology, $\{\emptyset, \Gamma\}$
- In $\mathfrak{M}^{\mathsf{STAR}\infty}$, the constructed program σ^* is interpreted by

$$\|\sigma^*\|_{\mathfrak{M}^{\mathsf{STAR}\infty}}(x, N^\frown s) = \|\sigma\|^N_{\mathfrak{M}^{\mathsf{U}\infty}}(x, s) = \underbrace{(\|\sigma\|_{\mathfrak{M}^{\mathsf{U}\infty}} \circ \|\sigma\|_{\mathfrak{M}^{\mathsf{U}\infty}} \circ \cdots \circ \|\sigma\|_{\mathfrak{M}^{\mathsf{U}\infty}})}_{N}(x, s)$$

In words: in order to interpret σ^*, pull the first element $N \in \mathbb{N}$ from the Γ-state (which is an endless stream of natural numbers), and perform σ, N times (if defined) from (x, s) – the state after we've removed N from the head of the stream.

3 Basic Theory of Program Constructors

3.1 General Program Constructors

Now that we have established the framework of program constructors and introduced our main examples, we turn our attention to understanding *how* such objects operate. Since program constructors are things which transform DTMs, and DTMs are models of a formal language, our analysis will naturally focus on how this transformation is reflected in the interpretation of this language. In particular, we want to understand how $\mathfrak{M}^C$'s interpretation of $\mathcal{L}^c_{\Box\bigcirc}$ is a function of (a) the structure of how C operates, and (b) how $\mathfrak{M}$ interprets $\mathcal{L}_{\Box\bigcirc}$. There are a variety of ways of grasping this, which we will be interested to explore.

As mentioned above, the notion of a "program constructor" is broad enough to admit a quite diverse array of constructions. Therefore, if we want to say anything meaningful about "how C operates", we will need to focus on specific program constructors, or at least more specific *classes* of program constructors. Subsect. 3.2 and Subsect. 3.3 will focus on the theory of $\mathsf{U}\omega$ and $\mathsf{U}\infty$ specifically (and Chapter 2 will mainly use those examples and OR). However, the logical properties of a C-augmented DTM $\mathfrak{M}^C$ rely heavily on its underlying Π-DTM, $\mathfrak{M}$. In this subsection (and more throughout Chapter 2) we will develop this connection in detail.

Let's begin by establishing a definition which will prove convenient.

Definition 3.1

Let Σ be a set and $\mathfrak{M} = (X, \tau_X, \{f_\sigma\}, V)$ a Σ-DTM.

- The $\mathcal{L}_{\Box\bigcirc}(\Sigma)$-**theory** of $\mathfrak{M}$, written $\mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{M})$, is defined by:

$$\mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{M}) := \{\varphi \in \mathcal{L}_{\Box\bigcirc}(\Sigma) \ : \ \mathfrak{M} \models \varphi\}$$

- For any $x \in X$, the $\mathcal{L}_{\Box\bigcirc}(\Sigma)$-**theory** of x, written $\mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{M}, x)$, is defined by:

$$\mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{M}, x) := \{\varphi \in \mathcal{L}_{\Box\bigcirc}(\Sigma) \ : \ (\mathfrak{M}, x) \models \varphi\}$$

We make several modifications on this notation:

- When some Π is understood as fixed in the background, $\mathrm{Th}_{\Box\bigcirc}(\mathfrak{M})$ will mean $\mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{M})$, and likewise for $\mathrm{Th}_{\Box\bigcirc}(\mathfrak{M}, x)$.
- Superscripts indicate constructed programs: we'll write $\mathrm{Th}^c_{\Box\bigcirc}(\mathfrak{M})$ to mean $\mathrm{Th}_{\Box\bigcirc}(\Pi^c; \mathfrak{M})$.
- 'PDL' subscript indicates use of $\mathcal{L}_{PDL}$ instead of $\mathcal{L}_{\Box\bigcirc}$: we'll write $\mathrm{Th}_{PDL}(\mathfrak{M})$ for the set of those $\varphi \in \mathcal{L}_{PDL}$ such that $\mathfrak{M} \models \varphi$.

This definition doesn't really introduce any new material to our theory, but rather supplies us a convenient notation. Many of the claims below will be phrased as an equality between the theories of two different models, or two different points within one model, or points in different models. The theory of a model (or of a point of a model) tells us everything the language "has to say" about that point. If, for instance, two very different models have the same theory in some language, then we might say that the language "cannot tell them apart" or "cannot express their difference". As is usual in mathematical logic (generally speaking), the ability of various formal languages to articulate the similarities and differences between different mathematical structures will be a key focus.

We use this notation to state some basic results of program constructor theory.

Proposition 3.1

For every Π-DTM $\mathfrak{M}$, every world $x \in |\mathfrak{M}|$, every program constructor C, and every program constructor state $\gamma \in \Gamma$,

$$\mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{M}, x) = \mathrm{Th}_{\Box\bigcirc}\left(\Pi; \mathfrak{M}^C, (x, \gamma)\right)$$

Notice that this is about the *primitive* theory of these DTMs, i.e. those formulas which express the properties of how Π-programs work.[16] What this says is that the language $\mathcal{L}_{\Box\bigcirc}(\Pi)$ cannot express the difference between a world x of $\mathfrak{M}$ and any of the "copies" (x, γ) of x in $\mathfrak{M}^C$. Thinking of $\mathfrak{M}^C$ as the same situation as $\mathfrak{M}$ (and (x, γ) the same state as x) but with more available actions, this makes sense: if we're not considering any of the additional actions (no $\mathcal{L}_{\Box\bigcirc}(\Pi)$ formula says anything about constructed programs), then there should be no difference.

As corollaries, we can get that this "theoretical equivalence" also holds globally for these DTMs.

Corollary 3.1.1

For every Π-DTM $\mathfrak{M}$ and every program constructor C,

$$\mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{M}) = \mathrm{Th}_{\Box\bigcirc}\left(\Pi; \mathfrak{M}^C\right).$$

And furthermore we can easily see that (x, γ) and (x, γ') are no different when it comes to $\mathcal{L}_{\Box\bigcirc}(\Pi)$.

Corollary 3.1.2

For all $\mathfrak{M}, C, x, \gamma, \gamma'$,

$$\mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{M}^C, (x, \gamma)) = \mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{M}^C, (x, \gamma'))$$

This latter result illuminates better how augmented situations work: if the agent is not using the device we introduce, then it doesn't matter what state the device is in. In Subsect. 2.1, we said we'd make sense of coin-flipping as the agent being in one of two states which are alike in every respect (and indistinguishable to the agent) except for the outcome of coin flips. This result is precisely what we meant by "alike in every respect": they have the same primitive theories.

Finally, let us connect this study of program constructors to the discussion of Subsect. 1.5 about determinism and continuity. Recall that actions π interpreted as continuous partial functions were deemed deterministic, whereas nondeterminism corresponded to discontinuity. To begin understanding how this notion interacts with the process of program construction, we prove that program constructors don't make *primitive* programs more or less deterministic than they already were.

Proposition 3.2

Let $\mathfrak{M}$ be a Π-DTM, C a program constructor, and $\pi \in \Pi$.

$$\|\pi\|_{\mathfrak{M}} \text{ is continuous (open)} \quad \text{iff} \quad \|\pi\|_{\mathfrak{M}^C} \text{ is continuous (open)}$$

Though we don't develop it further here, the property of "openness" is an interesting one for the interpreation of programs to have. [5] interprets openness epistemically as "perfect recall", and occasionally requires it for various constructions. Part of our reason for including it here is to show that the program construction is, in general, compatible with such requirements. We also include it because the interesting duality between continuity and openness – see the proof of Prop. 3.2 in the appendices.

So the results of this subsection can be summarized as "program constructors faithfully copy the interpretation of primitive programs". For all intents and purposes, $\mathfrak{M}$ and $\mathfrak{M}^C$ interpret primitive

[16] Indeed, $\mathfrak{M}$ is a Π-DTM, so those are the only kinds of programs it interprets.

programs the same way.[17] This is true of every program constructor, without qualification. With this established, our focus turns to understanding how specific program constructors interpret constructed programs against this "faithful copy" of the primitive dynamics.

3.2 Special Program Constructors

We described the intended functionality of a coin-flip as follows: doing "flip a coin. If heads, do σ_0; if tails, do σ_1" (i.e. doing $\sigma_0 \cup \sigma_1$) ultimately consists of either doing σ_0 or doing σ_1, but the agent cannot know in advance which one. We can split this description into two criteria:

1. $\sigma_0 \cup \sigma_1$ is ultimately either σ_0 or σ_1
2. The agent cannot know which one (as far as she can know, both are possible).

Our claim is that $\mathsf{U}\omega$ and $\mathsf{U}\infty$ produce DTMs which satisfy these desiderata. We start with the second one.

Proposition 3.3
Let C be either $\mathsf{U}\omega$ or $\mathsf{U}\infty$ and suppose $\mathfrak{M}$ is some Π-DTM, w a world of $\mathfrak{M}^C$, φ an $\mathcal{L}^{\cup}_{\Box\bigcirc}$ formula, and $\sigma_0, \sigma_1 \in \Pi^{\cup}$ such that $(\mathfrak{M}^C, w) \models \bigcirc_{\sigma_0}\varphi$ but $(\mathfrak{M}^C, w) \models \bigcirc_{\sigma_1}\neg\varphi$. Then,

$$(\mathfrak{M}^C, w) \models \Diamond \bigcirc_{\sigma_0 \cup \sigma_1} \varphi \wedge \Diamond \bigcirc_{\sigma_0 \cup \sigma_1} \neg\varphi.$$

Let's interpret this epistemically: suppose agent $\mathcal{A}$ is in a situation, and moreover has a coin which she can flip to make arbitrary binary decisions. Then, if it's the case that doing σ_0 would result in φ being true but doing σ_1 would result in φ being false, then, as far as $\mathcal{A}$ can know, it's possible that doing $\sigma_0 \cup \sigma_1$ could result in φ being true, or could result in φ being false. This is essentially our second criterion. We added the antecedent about σ_0 and σ_1 differing in their results because we needed this distinction to make the statement properly: if there's no such φ, then these actions have completely identical outcomes (at least that $\mathcal{L}^{\cup}_{\Box\bigcirc}$ can express), and so it doesn't really make a difference which one $\mathcal{A}$ does. But in this case (which is both more common and more interesting), there's an articulable difference between the outcomes of the two actions. In such a case, the agent can have no idea which one to expect. This is precisely the epistemic "opacity" we expect out of a coin-flip.

Furthermore, this makes good on the promise that coin flips ought to be nondeterministic. Suppose that φ is not only true after σ_0, but *knowably so*:

$$(\mathfrak{M}^C, w) \models \bigcirc_{\sigma_0} \Box\varphi.$$

But suppose $\sigma_0 \cup \sigma_1$ ends up being σ_0. So then it would be the case that

$$(\mathfrak{M}^C, w) \models \bigcirc_{\sigma_0 \cup \sigma} \Box\varphi. \tag{*}$$

So, still assuming that $(\mathfrak{M}^C, w) \models \bigcirc_{\sigma_1}\neg\varphi$, we obtain from Prop. 3.3 and some basic propositional reasoning that

$$(\mathfrak{M}^C, w) \models \Diamond \bigcirc_{\sigma_0 \cup \sigma_1} \neg\varphi.$$

From there, we can use the fact that $\|\sigma_1\|_{\mathfrak{M}^C}(w)$ is defined, plus some reasoning from ADTL (or just semantic reasoning about DTMs) to see that

$$(\mathfrak{M}^C, w) \not\models \Box \bigcirc_{\sigma_0 \cup \sigma} \varphi.$$

[17] We will later make this more explicit as a kind of semantic equivalence called a "surjective Π-bisimulation".

Putting this together with (*), we get

$$(\mathfrak{M}^C, w) \not\models \bigcirc_{\sigma_0 \cup \sigma_1} \Box \varphi \rightarrow \Box \bigcirc_{\sigma_0 \cup \sigma} \varphi$$

which is to say that $\sigma_0 \cup \sigma_1$ is nondeterministic at w. Notice our reliance on the assumption that $(\mathfrak{M}^C, w) \models \bigcirc_{\sigma_1} \neg \varphi$: if there is no $\mathcal{L}^{\cup}_{\Box\bigcirc}$ difference between the outcome of σ_0 and the outcome of σ_1 (e.g. if $\sigma_0 = \sigma_1$, or if they happen to result in indistinguishable states), then $\sigma_0 \cup \sigma_1$ could end up being deterministic.

Let's pivot back to our first criterion: $\sigma_0 \cup \sigma_1$ consists of either σ_0 or σ_1, not some unrelated action. There are various ways to express that claim, but here's one.

Lemma 3.4

Let C be either $\mathsf{U}\omega$ or $\mathsf{U}\infty$. Then, for any Π-DTM $\mathfrak{M}$, any $\sigma_0, \sigma_1 \in \Pi^{\cup}$ and any $\varphi \in \mathcal{L}^{\cup}_{PDL}$,

$$\mathfrak{M}^C \models \Diamond \bigcirc_{\sigma_0} \varphi \vee \Diamond \bigcirc_{\sigma_1} \varphi \rightarrow \Diamond \bigcirc_{\sigma_0 \cup \sigma_1} \varphi$$

Lemma 3.5

Let C be either $\mathsf{U}\omega$ or $\mathsf{U}\infty$. Then, for any Π-DTM $\mathfrak{M}$, any $\sigma_0, \sigma_1 \in \Pi^{\cup}$ and any $\varphi \in \mathcal{L}^{\cup}_{PDL}$,

$$\mathfrak{M}^C \models \Diamond \bigcirc_{\sigma_0 \cup \sigma_1} \varphi \rightarrow \Diamond \bigcirc_{\sigma_0} \varphi \vee \Diamond \bigcirc_{\sigma_1} \varphi$$

Together, what these claims express is that the possibilities (as far as the agent can know) for the outcome of $\sigma_0 \cup \sigma_1$ are given by the possible outcomes of σ_0 and σ_1. Notice that we are not claiming the equivalence of $\bigcirc_{\sigma_0 \cup \sigma_1} \varphi$ and $\bigcirc_{\sigma_0} \varphi \vee \bigcirc_{\sigma_1} \varphi$, but are requiring they be equivalent *up to the agent's possibility for knowledge*. In $\mathsf{U}\omega$- or $\mathsf{U}\infty$-augmented DTMs, notice that the interpretation of $\sigma_0 \cup \sigma_1$ at some world w does not actually consist of executing σ_0 or σ_1 at w, but rather at some nearby world w' with the same $\mathfrak{M}$-state but an updated Γ-state. To account for this difference, we must make this statement up to the agent's possible knowledge. This is also why we use the less expressive $\mathcal{L}^{\cup}_{PDL}$ language: as we'll see, $\mathcal{L}^{\cup}_{PDL}$ cannot express Γ-state variation – making a result like this possible.

However, we have a different motivation for putting this statement in the $\mathcal{L}^{\cup}_{PDL}$ language. Notice that these results, together, prove the validity of (U) axiom scheme (over $\mathcal{L}^{\cup}_{PDL}$) on $\mathsf{U}\omega$- and $\mathsf{U}\infty$-augmented DTMs. What we'll show now is that this axiom scheme completely captures the $\mathcal{L}^{\cup}_{PDL}$ logic of such models.

3.3 PDL Soundness and Completeness of Union

So far, we have provided ample indication that $\mathsf{U}\omega$ and $\mathsf{U}\infty$, which were designed from an examination of coin-flipping, satisfy formal properties which seem to match our ideas about this dynamic-epistemic phenomenon. An analogous development of, say, OR, would yield some similar results. To conclude this chapter, we will connect this result to relational PDL, and show that $\mathsf{U}\omega$ and $\mathsf{U}\infty$ serve as dynamic-topological implementations of nondeterministic union.

To state this result, it will be helpful to be able to refer to the *class* of C-augmented DTMs. This is given by the following definition.

Definition 3.2

Given some program constructor C, we will write DTM^C to refer to the class of all Π^c-DTMs of the form $\mathfrak{M}^C$ for some Π-DTM $\mathfrak{M}$.[18]

[18] As mentioned in Subsect. 0.1, readers who do not wish to think about proper classes may instead take statements involving DTM^C to merely be convenient shorthands for quantified claims about the C-augmented DTMs themselves, and not need to admit the 'class' of such structures as an object in its own right.

We will show a soundness and completeness result with respect to this class. Soundness and completeness results connect a particular axiom system to a given class of models: saying "Ax is a sound and complete axiomatization of $\mathcal{L}$ with respect to $\mathcal{K}$" (where Ax is some deductive system producing theorems in the language $\mathcal{L}$, and $\mathcal{K}$ is a class of models giving semantics for $\mathcal{L}$) just means that the set of *theorems* of the system Ax (or at least the ones expressible in the language $\mathcal{L}$) are precisely those $\mathcal{L}$-formulas which are valid on all models in $\mathcal{K}$. In this case, Ax captures the essential logic (expressible in $\mathcal{L}$) of models in $\mathcal{K}$.

As we saw in Lemma 3.4 and Lemma 3.5, the characteristic axiom of union-models, $\mathsf{(U)}$, is valid on all DTMs augmented with $\mathsf{U}\omega$ or $\mathsf{U}\infty$. Combined with Prop. 1.5, this gives us that the deductive system $\mathsf{PDL}_0 + \mathsf{(U)}$ – i.e. PDL_0 with every instance of $\mathsf{(U)}$ added as an axiom – is **sound** with respect to the classes $\mathsf{DTM}^{\mathsf{U}\omega}$ and $\mathsf{U}\infty$. What we now also claim is the converse, **completeness**.

Theorem 3.6
$\mathsf{PDL}_0 + \mathsf{(U)}$ is a sound and complete axiomatization of $\mathcal{L}^{\cup}_{PDL}$ with respect to $\mathsf{DTM}^{\mathsf{U}\omega}$

Theorem 3.7
$\mathsf{PDL}_0 + \mathsf{(U)}$ is a sound and complete axiomatization of $\mathcal{L}^{\cup}_{PDL}$ with respect to $\mathsf{DTM}^{\mathsf{U}\infty}$

The critical thing to note here is the language: this claim (at least the completeness part) is *NOT* true over the language $\mathcal{L}^{\cup}_{\Box\bigcirc}$ – there are many $\mathcal{L}^{\cup}_{\Box\bigcirc}$ formulas which are true on all $\mathsf{U}\omega$-augmented DTMs but not provable from $\mathsf{PDL}_0 + \mathsf{(U)}$.[19] What's interesting about this result is that, as far as the PDL language can express, both $\mathsf{U}\omega$ and $\mathsf{U}\infty$ faithfully carry out the logic of relational nondeterministic union in the dynamic-topological setting, thus fulfilling one of our central goals.

Finally, let us say one word about the proof of these theorems (they have a common proof, which needs only trivial adjusting to suffice for either).[20] As we've mentioned, the soundness result is contained in what we've already proved. Conversely, recall that completeness (in the case of Theorem 3.6) is the following statement:

$$\mathfrak{M}^{\mathsf{U}\omega} \models \varphi \text{ for all } \mathfrak{M} \quad \Longrightarrow \quad \vdash_{\mathsf{PDL}_0+\mathsf{(U)}} \varphi.$$

We prove the contrapositive: that any nontheorem of $\mathsf{PDL}_0 + \mathsf{(U)}$ is refuted on some DTM of the form $\mathfrak{M}^{\mathsf{U}\omega}$. To produce this "countermodel", we leverage a relational completeness result. In particular, it's the case that any nontheorem of $\mathsf{PDL}_0 + \mathsf{(U)}$ is refuted on some *union-model* M. Then we do a somewhat strange thing: we *forget* that M interprets $\cup$-programs, and consider it just as a Π-relational model. We do this so we can transform it into an equivalent Π-DTM $\mathfrak{M}$ (we have a transformation which does this, from the proof of completeness for Prop. 1.5). The idea is this: if $\mathsf{U}\omega$ (or $\mathsf{U}\infty$) is genuinely playing the same role as the "relational program construction" given by the equation $R_{\sigma_0 \cup \sigma_1} = R_{\sigma_0} \cup R_{\sigma_1}$, then adding an interpretation of $\cup$ to $\mathfrak{M}$ by augmenting it to $\mathfrak{M}^{\mathsf{U}\omega}$ (or $\mathfrak{M}^{\mathsf{U}\infty}$) will end up achieving the same refutation of φ as we had in M. Lots of details must be checked to verify this, but it all works out.

Conclusion of Chapter 1

In the earlier part of this work (particularly Sect. 1 and Subsect. 2.1), we established three primary ingredients which combine to give the present inquiry:

[19] This is true for reasons which have little to do with program construction: for instance, $\mathsf{PDL}_0 + \mathsf{(U)}$ does not prove $\Box\varphi \to \Box\Box\varphi$. But even remedying this to, say, $\mathsf{ADTL} + \mathsf{(U)}$ still doesn't completely capture the logic of $\mathsf{U}\omega$ or $\mathsf{U}\infty$. We will need more elaborate tools to get something like that.

[20] As we'll explore in the next chapter, the fact that these two program constructors – which are quite distinct – have the exact same axiomatization (and that there is no essential difference in the proofs of the soundness and completeness results) speaks to the profound limitations of the PDL language: it just isn't expressive enough to articulate the differences between $\mathsf{U}\omega$ and $\mathsf{U}\infty$. The full $\mathcal{L}^{\cup}_{\Box\bigcirc}$ language, however, certainly can express the difference.

1. Intuitions about agents navigating situations (especially coin-flipping situations)
2. Dynamic topological logic (including a theory of models and a deductive calculus)
3. Relational PDL and its process for interpreting constructed programs.

The central aim of this chapter was to establish a series of interconnections between these topics, specifically to develop a framework for doing PDL-style program construction in agent DTL, in such a way that admitted a plausible reading as the structure of an agent's possibilities navigating a situation with the help of a (possibly nondeterministic) device. The general framework we expounded – the abstract definition of a *program constructor* – provides a basic terminology and suite of basic results (Subsect. 3.1) to approach such a task. Within this framework, we were able to describe multiple program constructors which (in various ways) carry out the logic of nondeterministic union in the setting of agent DTL. The results of the previous section (especially the soundness and completeness results) highlighted how program construction generally (and $\mathsf{U}\omega$ and $\mathsf{U}\infty$ specifically) managed to stay true to both our ideas about agents and the relational PDL process it was mimicking.

From here, talk of agents and situations will take more of a backseat, and we will focus more on developing the mathematical theory. Our impetus will be the shortcomings of the soundness and completeness results: while these did well to tie agent DTL program construction to PDL program construction, they manifestly fail to distinguish $\mathsf{U}\omega$ from $\mathsf{U}\infty$: the PDL language (specifically the (U) axiom scheme alone) are just not enough to capture how these constructors work in fine detail. Building enough mathematical theory to achieve such a characterization now becomes our main focus.

Chapter 2

Advanced Theory of Program Constructors

Introduction

Our aim so far has been to develop the theory of this "agent DTL" as a framework for reasoning about epistemology. But at this point, we turn our focus to expounding the pure mathematical theory of this dynamic topological logic. So we'll briefly untether ourselves from epistemic intuitions and just go where the math leads. At the end of the day, we'll be able to bring this inquiry back to the more familiar intuitions from the before. So far, we've covered several topics which are essential to a mathematical development of this variety of dynamic topological logic, but there's much to be explored. This chapter exists to document some interesting parts of this vast and interesting field of inquiry – much will still be left to investigate after we're done.

We organize our development of agent DTL around a quest to understand program constructors better, and particularly to express their properties in the object language. We saw at the end of the previous chapter that we can axiomatize the $\mathcal{L}^{\cup}_{PDL}$ theories of $\mathsf{U}\omega$-agumented DTMs and $\mathsf{U}\infty$-augmented DTMs. But $\mathcal{L}^{\cup}_{PDL}$ is a relatively inexpressive fragment of the full language $\mathcal{L}^{\cup}_{\Box\bigcirc}$ that such DTMs can interpret. In particular, it exemplifies the limitations of $\mathcal{L}^{\cup}_{PDL}$ that it cannot express the difference between $\mathsf{U}\omega$ and $\mathsf{U}\infty$. What we want to do is bring the whole $\mathcal{L}^{\cup}_{\Box\bigcirc}$ language to bear, and try to characterize the behavior of the program constructor in this much more expressive object language. This will also bring us back to our intuitive understanding of agent DTL, because the formulas of $\mathcal{L}^{\cup}_{\Box\bigcirc}$ admit an epistemic reading. In particular, $\mathcal{L}^{\cup}_{\Box\bigcirc}$ formulas express the dynamic-epistemic properties of coin-flipping situations from the perspective of the coin-flipping agent. If we manage to obtain a characterization of, say, $\mathsf{U}\infty$ in the language $\mathcal{L}^{\cup}_{\Box\bigcirc}$, then we can interpret those formulas, and see whether they make sense as descriptions of how coin-flipping situations work. But before we can do this, we must clarify what "characterize" means: when does a set $\Delta \subseteq \mathcal{L}^{c}_{\Box\bigcirc}$ *characterize* a program constructor C, and what does it tell us about C-augmented models and frames if it does? Making sense of this will prove to be quite a task – we will reject several candidate notions of "characterize" before finding the right one – and will serve as the central goal we're working towards for this chapter.

The most immediate thing we might try is defining the class of C-augmented DTMs as a subclass of the class of all Π^{c}-DTMs. If we could, then the defining formulas would articulate the "essential" properties of how the program constructor works, delivering us the result we wanted. While it's not clear how possible this kind of result is to obtain for arbitrary program constructors C (it might work nicely for some C), the outlook is pessimistic for the program constructors we're concerned with. As an illustrative example, we ask the question: can the class $\mathsf{DTM}^{\mathsf{OR}}$ of

OR-augmented DTMs be defined (as a subclass of the class of *all* Π^{or}-DTMs) using the language $\mathcal{L}_{\Box\bigcirc}(\Pi^{\mathsf{or}})$? Even if we weaken the requirements of this question, we'll find that the answer is **no**. If we instead inquire whether we can define $\mathsf{Frame}^{\mathsf{OR}}$ of OR-augmented *frames*, the answer is **no**. If we inquire whether we can define $\mathsf{Frame}^{\mathsf{OR}}$-BISIM of frames *bisimilar* to an OR-augmented frame (for a notion of "bisimilar" which we'll develop), the answer is – I believe – **no**. Moreover, the proofs of these negative results will indicate that there's a pattern here: defining classes of models (or frames) in this way is simply ill-suited to study program constructors (or at least all of our examples of program constructors), and likely won't lead to useful results.[1] If we want to characterize our program constructors, we need to look elsewhere.

We then lay the groundwork for what will turn out to be a more successful notion of "characterization", informed by our failures to define classes of OR-augmented structures. This consists of expounding a novel theory about dynamic topological frames, which we call *refined frame theory*. Refined frame theory allows us more freedom in how we articulate dynamic-topological structure using $\mathcal{L}_{\Box\bigcirc}(\Sigma)$. In particular, refined frame theory will turn out to be a quite fitting tool set to study the structure and function of program constructors. After proving several invariance results and developing some useful techniques for working with refined frames, we define an appropriate notion of *characterization*. Characterization results will be generally obtainable,[2], and will provide very insightful descriptions of our program constructors in the object language. We conduct a characterization of OR, and provide some indication of a possible approach to characterizing the more elaborate constructors Uω and U∞. Attempts to characterize the latter two are beset by further difficulties (which we make some effort to describe), which undoubtedly will serve as impetus for interesting future study.

4 Undefinability

4.1 Bisimulation

First, let us introduce a notion we'll use throughout. Recall the statements of Prop. 3.1 and Corollary 3.1.1, which articulated that $\mathfrak{M}$ and $\mathfrak{M}^C$ have the same $\mathcal{L}_{\Box\bigcirc}(\Pi)$ theories, both pointwise and globally. To prove such results, we could always proceed by structural induction on $\mathcal{L}_{\Box\bigcirc}(\Pi)$ formulas, proving inductively for each ψ that $\mathfrak{M}$ (or a world of $\mathfrak{M}$) validates ψ iff the corresponding worlds in $\mathfrak{M}^C$ do. But this turns out to be the kind of argument we frequently need to employ, so good mathematical practice would be to abstract that reasoning away for easy application in numerous places. The concept we need in order to do this is *bisimulation*. Bisimulations are a standard tool in modal logic and in theoretical computer science[3], and can be readily adapted to our purposes.

Definition 4.1 (Bisimulation)
Let Θ, Θ' be sets, and $\Sigma \subseteq \Theta \cap \Theta'$. If $\mathfrak{M} = (X, \tau_X, \{f_\sigma\}, V_X)$ is a Θ-DTM and $\mathfrak{N} = (Y, \tau_Y, \{g_\sigma\}, V_Y)$ is a Θ'-DTM, then a binary relation $s \subseteq X \times Y$ is said to constitute a **Σ-bisimulation** between $\mathfrak{M}$ and $\mathfrak{N}$ if the following conditions are satisfied:

[1] This also provides a fairly clear indication that this string of negative results is not peculiar to OR. Though we do not pursue this topic explicitly here, my suspicion is that similar tricks could be used to defeat the prospect of obtaining these kinds of results for Uω, U∞, etc.

[2] Though perhaps somewhat difficult and lengthy to prove

[3] See [15] and [8] for a treatment of bisimulations in relational modal logic, [17, pg. 17] in topological modal logic, and [12] and [14] in the dynamic logic of programs.

- (Base) For every $p \in \Phi$, every $x \in X$ and every $y \in s(x)$[4],

$$x \in V_X(p) \quad \Longleftrightarrow \quad y \in V_Y(p)$$

- (Forth) If $x \in U \in \tau_X$ and xsy, then there is a $U' \in \tau_Y$ such that $y \in U' \subseteq s(U)$
- (Back) If $y \in U' \in \tau_Y$ and xsy, then there is a $U \in \tau_M$ such that $x \in U \subseteq s^{-1}(U')$[5]
- (Respects Σ) For each $\sigma \in \Sigma$,
  - (σ-Forth) If xsy and $f_\sigma(x)$ is defined, then $g_\sigma(y)$ is defined and $f_\sigma(x)$ is related to $g_\sigma(y)$ by s.
  - (σ-Back) If xsy and $g_\sigma(y)$ is defined, then $f_\sigma(x)$ is defined and $f_\sigma(x)$ is related to $g_\sigma(y)$ by s.
- (Total) s is total: the domain of s is X.

We'll write $s : \mathfrak{M} \twoheadrightarrow_\Sigma \mathfrak{N}$ to indicate that s is a Σ-bisimulation from $\mathfrak{M}$ to $\mathfrak{N}$. We'll say that $\mathfrak{M}$ and $\mathfrak{N}$ are **Σ-bisimilar** (in symbols, $\mathfrak{M} \twoheadrightarrow_\Sigma \mathfrak{N}$) if there is some Σ-bisimulation between them.

We'll write $s : (\mathfrak{M}, x) \twoheadrightarrow_\Sigma (\mathfrak{N}, y)$ to indicate $s : \mathfrak{M} \twoheadrightarrow_\Sigma \mathfrak{N}$ and xsy, in which case we say "x is Σ-bisimilar to y".

The essential idea of a bisimulation is to connect worlds of $\mathfrak{M}$ with worlds of $\mathfrak{N}$ which are "structurally equivalent", in that the two worlds have – as far as $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ can express – indistinguishable surroundings. And it is for this reason that they'll turn out to have identical $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ theories. The different clauses of the definition guarantee that the different components of the dynamic-topological structure are "similar". Let's explore them in turn.

The (Base) condition ensures that the two models in question "agree on primitive propositions": bisimilar worlds must validate the exact same primitive propositions.[6] (Back) and (Forth) are the usual conditions from topological modal logic, which express that the bisimulation respects the topological structure in the proper way. Indeed, (Forth) and (Back) articulate the relational analogues of standard notions of topological similarity.

Proposition 4.1

Suppose $s \subseteq X \times Y$ is a binary relation between two topological spaces (X, τ_X) and (Y, τ_Y). (1) and (2) are equivalent to each other, and (3) and (4) are equivalent to each other.

(1) s is *continuous*: for every $U' \in \tau_Y$, its preimage $s^{-1}(U') = \{x \in X \ : \ xsy \text{ for some } y \in U'\}$ is in τ_X

(2) s satisfies (Back): If $y \in U' \in \tau_Y$ and xsy, then there is a $U \in \tau_X$ such that $x \in U \subseteq s^{-1}(U')$.

(3) s is *open*: for every $U \in \tau_X$, its image $s(U) = \{y \in Y \ : \ xsy \text{ for some } x \in U\}$ is in τ_Y

(4) s satisfies (Forth): If $x \in U \in \tau_X$ and xsy, then there is a $U' \in \tau_Y$ such that $y \in U' \subseteq s(U)$.

[4] Recall $s(x)$ denotes the set of those $y \in Y$ such that xsy

[5] Recall $s^{-1}(U')$ is the set of those $x \in M$ such that xsy for some $y \in U'$

[6] We'll later articulate this same point by saying that the two valuations "respect" the equivalence established by the bisimulation.

Keep in mind that these are "relational" notions of openness and continuity: $s(U)$, the image of U under s, is all those points y related to some element of U (which generalizes the usual notion of the image of a set under a function). For some purposes, it will be helpful to use the earlier statement of (Back) and (Forth); in other contexts, thinking of (Forth) as openness and (Back) as continuity will be what we want.

Similarly, the (σ-Forth) and (σ-Back) conditions ensure that bisimilar models have the same dynamic structure – that the simulation both *preserves* and *reflects* the system of partial functions.

Despite the stringency of some of these definitions, we actually do have an immediate variety of interesting and relevant examples of bisimulations between dynamic topological models. In paticular, this will include our original impetus: program constructors.

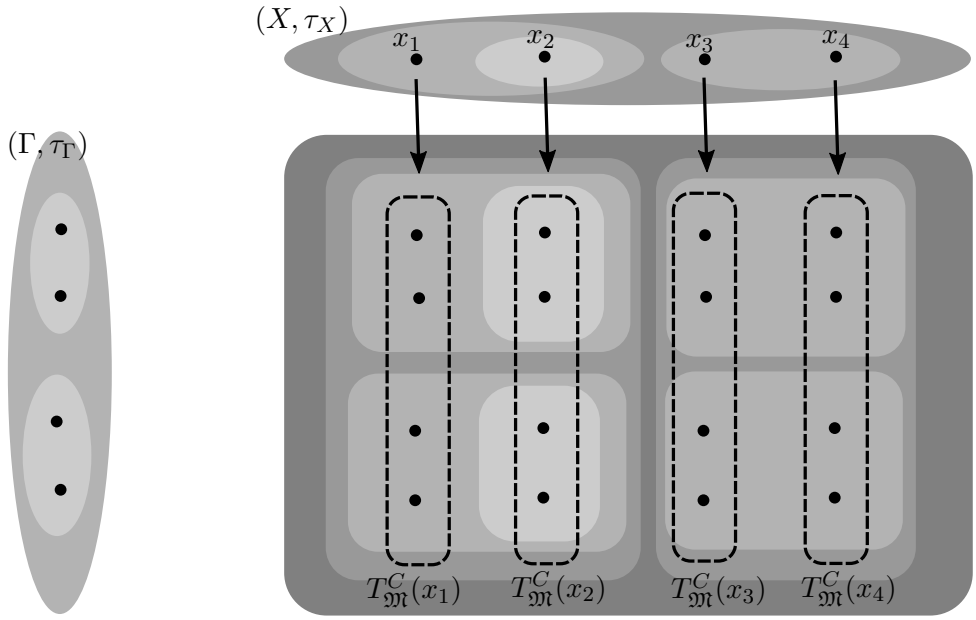
Example 4.1

Let $\mathfrak{M} = (X, \tau_X, \{f_\pi\}, V)$ be a DTM and C a program constructor with state space Γ. Define the relation $T^C_{\mathfrak{M}}$ between X and $X \times \Gamma$ by

$$x \; T^C_{\mathfrak{M}} \; (x', \gamma) \quad \text{iff} \quad x = x'$$

In other words: each $x \in X$ is related to every element of $X \times \Gamma$ whose first coordinate is x (this is depicted in the figure below). $T^C_{\mathfrak{M}}$ is a Π-bisimulation:[7]

$$T^C_{\mathfrak{M}} : \mathfrak{M} \twoheadrightarrow_\Pi \mathfrak{M}^C$$



So we've established a bisimulation between $\mathfrak{M}$ and $\mathfrak{M}^C$. Now, we begin to put this concept to work expediting the proofs of claims like Prop. 3.1.

[7] Note that $T^C_{\mathfrak{M}}$ *cannot* be a Π^c-bisimulation, since its domain is merely a Π-DTM.

Theorem 4.2
If $(\mathfrak{M}, x) \twoheadrightarrow_\Sigma (\mathfrak{N}, y)$, the following equality holds:

$$\mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{M}, x) = \mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{N}, y).$$

So what this theorem says is that the existence of a Σ-bisimulation guarantees the $\mathcal{L}_{\Box\bigcirc}(\Sigma)$-equivalence of the points it connects. When we want to instead prove that the two DTMs in question have the same "global" theory, we need our bisimulation to also be *surjective*.

Definition 4.2
Let $s : \mathfrak{M} \twoheadrightarrow_\Sigma \mathfrak{N}$.

- We say that s is **surjective** if the image of s[8] is all of $|\mathfrak{N}|$, [9] in which case we write $s : \mathfrak{M} \simeq_\Sigma \mathfrak{N}$. We indicate the existence of such an s by writing $\mathfrak{M} \simeq_\Sigma \mathfrak{N}$.
- If $s : \mathfrak{M} \simeq_\Sigma \mathfrak{N}$ is furthermore an injective function $s : |\mathfrak{M}| \to |\mathfrak{N}|$, then s is called a **isomorphism**. In this case, we say $\mathfrak{M}$ and $\mathfrak{N}$ are Σ-isomorphic, which we denote $\mathfrak{M} \cong_\Sigma \mathfrak{N}$.

Corollary 4.2.1
If $\mathfrak{M} \simeq_\Sigma \mathfrak{N}$,

$$\mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{M}) = \mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{N}).$$

Since the $T^C_{\mathfrak{M}}$ bisimulation is indeed surjective, we can use Corollary 4.2.1 to prove Corollary 3.1.1. In addition to abstracting away the reasoning behind these results, bisimulations will also play a central role in our overall task of studying the theory of program constructors. We shall see why in a moment.

4.2 Model

So, as mentioned above, our goal is to assign to a program constructor C some collection of $\mathcal{L}^c_{\Box\bigcirc}$ formulas which articulate the properties of the program constructor. We are in search of a meaningful way to do that. A natural approach is to examine the class DTM^C of all those DTMs $\mathfrak{M}^C$ produced by the program constructor – which is a subclass of the class of all Π^c-DTMs. Obviously, the formulas satisfied by every DTM in DTM^C[10] tell us something about how the program constructor works. So a simple candidate notion for a "characterization" of C is just the set of formulas $\Delta = \left\{\varphi \in \mathcal{L}^c_{\Box\bigcirc} \ : \ \mathfrak{M}^C \models \varphi \text{ for all } \mathfrak{M}\right\}$. But this feels inadequte as a "characterization": there are – in general – plenty of DTMs $\mathfrak{N}$ which are not of the form $\mathfrak{M}^C$ but which still validate all of Δ. The idea of class *definition* seeks to improve upon this.

Definition 4.3
A set $\Delta \subseteq \mathcal{L}_{\Box\bigcirc}(\Sigma)$ **defines** a class $\mathcal{K}$ of Σ-DTMs if

$$\mathfrak{M} \in \mathcal{K} \quad \text{iff} \quad \mathfrak{M} \models \Delta$$

A class $\mathcal{K}$ is said to be **definable** if there exists a Δ which defines $\mathcal{K}$.

So if Δ defines a class DTM^C, then a Π^c-DTM $\mathfrak{N}$ is of the form $\mathfrak{M}^C$ iff $\mathfrak{N} \models \Delta$. So if we obtain such a Δ, then Δ must express the essential features of how C works: the models that validate Δ are exactly those produced by C. The only question is, then, whether DTM^C is definable. The immediate answer is *no*, but for a reason which is quick to fix.

[8] The set $s(\mathfrak{M}) := \{y \in |\mathfrak{N}| \ : \ xsy \text{ for some } x \in |\mathfrak{M}|\}$
[9] Some authors use the term "total" to mean "total" (in our sense) *and* surjective.
[10] E.g. every DTM in $\mathsf{DTM}^{\mathsf{U}\infty}$ validates every instance of (U).

Note 4.1
If $\mathcal{K}$ is a $\mathcal{L}_{\Box\bigcirc}(\Sigma)$-definable class of Σ-DTMs, then $\mathcal{K}$ is closed under surjective[11] Σ-bisimulation:

$$\mathfrak{M} \simeq_\Sigma \mathfrak{N} \text{ for some } \mathfrak{N} \in \mathcal{K} \quad \Longrightarrow \quad \mathfrak{M} \in \mathcal{K}.$$

To see this, observe that $\mathfrak{N} \in \mathcal{K}$ and Δ defines $\mathcal{K}$, so $\mathfrak{N} \models \Delta$. But, by Corollary 4.2.1 and the fact that $\mathfrak{M} \simeq_\Sigma \mathfrak{N}$, we get $\mathfrak{M} \models \Delta$. Thus $\mathfrak{M} \in \mathcal{K}$.

So therefore, if we want to have any hope of DTM^C being definable, we need to close it under Π^c-bisimulation.

Definition 4.4
Given any program constructor C, the class $\mathsf{DTM}^C\text{-}\mathsf{BISIM}$ consists of all those Π^c-DTMs $\mathfrak{N}$ such that $\mathfrak{N} \simeq_{\Pi^c} \mathfrak{M}^C$ for some Π-DTM $\mathfrak{M}$.

However, this too proves impossible to define.

Theorem 4.3
The class $\mathsf{DTM}^{\mathsf{OR}}\text{-}\mathsf{BISIM}$ is not definable.

Let us say a brief word on how to prove such a result, which will illuminate better what this result says. A (reasonably) straightforward way to demonstrate that a class $\mathcal{K}$ of Σ-DTMs is *not* definable by *any* set of $\mathcal{L}_{\Box\bigcirc}(\Sigma)$-formulas is to provide two models $\mathfrak{M}$ and $\mathfrak{N}$ such that:

(a) $\mathfrak{M} \in \mathcal{K}$,

(b) $\mathfrak{N} \notin \mathcal{K}$, but such that

(c) $\mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{M}) \subseteq \mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{N})$.

The existence of this setup makes it impossible for any set Δ of $\mathcal{L}_{\Box\bigcirc}(\Sigma)$-formulas to define $\mathcal{K}$. To see why, we argue that if Δ is assumed to define $\mathcal{K}$, we can use (a), (b), and (c) to obtain a contradiction. The two relevant aspects of this situation are depicted in Figure 4.1 and Figure 4.2.

"Δ defines $\mathcal{K}$" means precisely that $\mathcal{K}$ is exactly the class of all Σ-DTMs validating Δ. Thus, if $\mathfrak{N} \notin \mathcal{K}$, then it must be the case that $\mathfrak{N} \not\models \Delta$. This is the situation depicted in Figure 4.1. On the other hand, since $\mathfrak{M} \in \mathcal{K}$, it must be the case that $\mathfrak{M} \models \Delta$. That is, $\Delta \subseteq \mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{M})$. But $\Delta \subseteq \mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{M}) \subseteq \mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{N})$ and we thus get $\mathfrak{N} \models \Delta$. This is depicted in Figure 4.2. Thus we have a contradiction, and the assumption that Δ defined $\mathcal{K}$ must fall. So a class $\mathcal{K}$ where this setup is *possible* is necessarily undefinable. This is the approach we use to prove Theorem 4.3.

Specifically, we define a Π^{or}-DTM $\mathfrak{N}$ such that $\mathrm{Th}^{\mathsf{or}}_{\Box\bigcirc}(\mathfrak{P}^{\mathsf{OR}}) \subseteq \mathrm{Th}^{\mathsf{or}}_{\Box\bigcirc}(\mathfrak{N})$ for some Π-DTM $\mathfrak{P}$., but such that $\mathfrak{N}$ is not Π^{or}-bisimilar to *any* OR-augmented DTM.[12]. What this means is that $\mathcal{L}^{\mathsf{or}}_{\Box\bigcirc}$ is not discerning enough to distinguish whether or not a given Π^{or}-DTM is "genuinely" (bisimilar to) an OR-augmented DTM. Thus we have

(a) $\mathfrak{P}^{\mathsf{OR}} \in \mathsf{DTM}^{\mathsf{OR}}\text{-}\mathsf{BISIM}$. Indeed, $\mathfrak{P}^{\mathsf{OR}}$ is bisimilar to an OR-augmented DTM: itself.

(b) $\mathfrak{N} \notin \mathsf{DTM}^{\mathsf{OR}}\text{-}\mathsf{BISIM}$ because $\mathfrak{N}$ is not bisimilar to any OR-augmented DTM.

(c) $\mathrm{Th}^{\mathsf{or}}_{\Box\bigcirc}(\mathfrak{P}^{\mathsf{OR}}) \subseteq \mathrm{Th}^{\mathsf{or}}_{\Box\bigcirc}(\mathfrak{N})$

[11] Surjectivity isn't actually needed here, but we include it for simplicity.
[12] Precisely: there is no Π-DTM $\mathfrak{P}$ such that $\mathfrak{N} \simeq_{\Pi^{\mathsf{or}}} \mathfrak{P}^{\mathsf{OR}}$

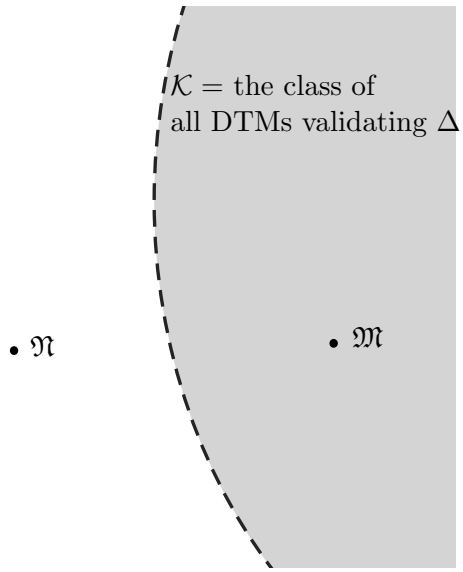


Figure 4.1:
Δ defines $\mathcal{K}$, (a) $\mathfrak{M} \in \mathcal{K}$ and (b) $\mathfrak{N} \notin \mathcal{K}$

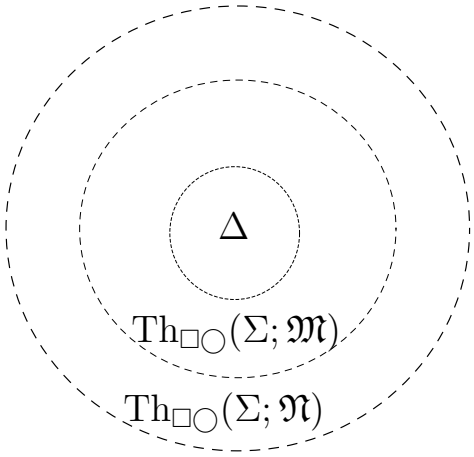


Figure 4.2:
$\mathfrak{M} \models \Delta$ and $\mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{M}) \subseteq \mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{N})$

Put these together and we obtain the contradiction. Intuitively, the $\mathfrak{N}$ we construct is good enough of a "counterfeit OR-augmented DTM" to fool $\mathcal{L}^{\mathsf{or}}_{\Box\bigcirc}$ into thinking it is really OR-augmented, when it in fact is quite different structurally.

Briefly, this is done by exploiting the fact that all $\mathcal{L}^{\mathsf{or}}_{\Box\bigcirc}$ formulas are of *finite length*. You see, the key feature of OR-augmented DTMs is that there are pairs of worlds which are identical as far as *primitive* programs go, but are quite different as far as *constructed* programs go. In an OR-augmented DTM, the worlds $(x, 0)$ and $(x, 1)$ are identical primitively, but differ on $(\pi_1 \ \mathsf{or} \ \pi_2)$ programs ("in one world, a coin flip comes up heads, in the other tails"). That is, for every world w of an OR-augmented DTM, there is another world w' which is exactly the same except that it interprets or-programs in the opposite way. **This condition cannot be enforced in the object language**: finite-length formulas cannot guarantee that two worlds agree on infinitely-many formulas. We build our "counterfeit" DTM precisely by fooling $\mathcal{L}^{\mathsf{or}}_{\Box\bigcirc}$ into *thinking* we satisfied this requirement, when we, in fact, have not. I believe this (or similar) techniques could be utilized to prove that, for any of the other constructors we introduced (at least the nondeterministic ones), the class of (Π^c-DTMs bisimilar to) DTMs augmented with that constructor *also cannot be defined*.

We could, of course, expand the class $\mathcal{K}$ we're interested in to include these "really good counterfeits", but it's not clear what we'd arrive at. My view is that the counterfeit OR-augmented DTM $\mathfrak{N}$ is sufficiently unlike "real" OR-augmented DTMs that trying to encompass $\mathfrak{N}$ into our characterization of OR would be a mistake. Furthermore, as mentioned in the previous paragraph, this issue does not appear to be unique to OR, but seems to affect all the program constructors we're interested in. On this basis, we conclude that defining classes of models by sets of formulas will not deliver us the results we want. We will instead seek out a different notion.

4.3 Frame

A reader well-versed in modal logic will know that defining classes of models (in the foregoing sense) is less common than trying to define classes of *frames*. Reviewing the topic of *frame definability* in relational modal logic will likely prove helpful background for understanding the content of the present section – we refer the reader to sources linked in Subsect. 0.1 (or any standard development

of modal logic) for this. We'll proceed to make the analogous definitions for *dynamic-topological* frames, as well as restating some of our key definitions for frames.

Definition 4.5

Given a Σ-frame $\mathcal{F}$ and a formula $\varphi \in \mathcal{L}_{\Box\bigcirc}(\Sigma)$, we say $\mathcal{F}$ **validates** φ and write $\mathcal{F} \models \varphi$ just in case

$$(\mathcal{F}, V) \models \varphi \text{ for all } V : \Phi \to \mathcal{P}(|\mathcal{F}|).$$

Definition 4.6

A set $\Delta \subseteq \mathcal{L}_{\Box\bigcirc}(\Sigma)$ is said to **define** a class $\mathcal{K}$ of Σ-frames if

$$\mathcal{F} \in \mathcal{K} \quad \text{iff} \quad \mathcal{F} \models \Delta$$

So this is a similar idea, but affords us much greater power to specify the desired structure. The reason this notion is better equipped for requiring certain structure is that we are able to "control the valuation" for the purposes of designing counterexamples: if $\mathcal{F} \models \varphi$, this says that *every* DTM we could define on $\mathcal{F}$ – not just one particular DTM – must validate φ. Therefore, if we have a frame which is *not* in the class we're trying to define, then we have an easier time refuting formulas from Δ: we can adversarially pick some V, and know that if $(\mathcal{F}, V) \not\models \varphi$ for just this single V and a single $\varphi \in \Delta$, then $\mathcal{F} \not\models \varphi$ as a whole. This higher bar means there's less chance of a "false positive", a frame which validates the formulas but is not in the class. This lends hope that we'll do better sniffing out counterfeits, and might be able to define the classes of structures we're interested in.

The classes of frames we're interested in, of course, are augmented frames.

Definition 4.7 (Frame-to-Frame Program Constructor)

Let $n \in \mathbb{N}$ be given. An (n-ary) **program constructor** C consists of an n-ary function symbol c, a topological space (Γ, τ_Γ) and a rule which assigns to each Π-frame $\mathcal{F} = (X, \tau_X, \{f_\pi\}_{\pi\in\Pi})$ a Π^c-frame $\mathcal{F}^C$ such that

- $\left|\mathcal{F}^C\right| = X \times \Gamma$;
- the topology of $\mathcal{F}^C$ is the product topology of τ_X and τ_Γ;
- for all $\pi \in \Pi$, all $x \in X$, and all $\gamma \in \Gamma$,

$$\|\pi\|_{\mathcal{F}^C}(x, \gamma) = (f_\pi(x), \gamma)$$

This definition, of course, is just Defn. 2.1 with any reference to valuations omitted (since frames do not have an associated valuation). Similarly, we can do the same with our definition of bisimulations by omitting the (Base) condition.

Definition 4.8

A relation $s \subseteq X \times Y$ is said to be a **(frame) Σ-bisimulation** between Σ-frames $\mathcal{F} = (X, \tau_X, \{f_\sigma\})$ and $\mathcal{G} = (Y, \tau_Y, \{g_\sigma\})$ if the following conditions are met.

- (Forth) If $x \in U \in \tau_X$ and xsy, then there is a $U' \in \tau_Y$ such that $y \in U' \subseteq s(U)$
- (Back) If $y \in U' \in \tau_Y$ and xsy, then there is a $U \in \tau_X$ such that $x \in U \subseteq s^{-1}(U')$
- (Respects Σ) For each $\sigma \in \Sigma$,
  - (σ-Forth) If xsy and $f_\sigma(x)$ is defined, then $g_\sigma(y)$ is defined and $f_\sigma(x)$ is related to $g_\sigma(y)$ by s.

  - (σ-Back) If xsy and $g_\sigma(y)$ is defined, then $f_\sigma(x)$ is defined and $f_\sigma(x)$ is related to $g_\sigma(y)$ by s.

- (Total) s is total: the domain of s is X.

If the image of s is all of Y, we call s **surjective**. We adopt all our same notations from Defn. 4.1 and Defn. 4.2, e.g. writing $s : (\mathcal{F}, x) \simeq_\Sigma (\mathcal{G}, y)$ to indicate that s is a surjective Σ-bisimulation between $\mathcal{F}$ and $\mathcal{G}$ and xsy.

Note 4.2
Given Σ-DTMs $\mathfrak{M} = (\mathcal{F}, V)$ and $\mathfrak{N} = (\mathcal{G}, V')$ and a Σ-bisimulation $s : \mathfrak{M} \twoheadrightarrow_\Sigma \mathfrak{N}$, then s is also a frame Σ-bisimulation between the frames $\mathcal{F}$ and $\mathcal{G}$.

So, in particular, the bisimulation $T^C_{\mathfrak{M}} : \mathfrak{M} \simeq_\Pi \mathfrak{M}^C$ can be regarded as a bisimulation of the underlying frames $T^C_{\mathcal{F}} : \mathcal{F} \simeq_\Pi \mathcal{F}^C$. It turns out that we do not have the same invariance result between bisimilar frames that we did for models: it's possible to have $\mathcal{F} \simeq_\Sigma \mathcal{G}$ but for $\mathcal{F}$ and $\mathcal{G}$ to have different $\mathcal{L}_{\Box\bigcirc}(\Sigma)$-theories. In the next section,[13] part of what we will obtain is a notion of a frame's "theory" which *is* invariant across bisimulations. But that won't matter for the point we make below.

So, continuing to trace the steps we made in the model case, we define the class of "C-augmented frames, up to bisimulation".

Definition 4.9
Given a program constructor C, the class Frame^C-BISIM consists of exactly those Π^c-frames $\mathcal{G}$ such that $\mathcal{G} \simeq_{\Pi^c} \mathcal{F}^C$ for some Π-frame $\mathcal{F}$

And we arrive again at the question: *can we define this class?* If we could, then we would have an excellent object-language characterization of how our program constructor works: if Δ defined Frame^C-BISIM, then *every* DTM atop a C-augmented frame (including all C-augmented DTMs) would validate Δ, and moreover a frame would validate Δ only if it was (up to bisimulation) of the form $\mathcal{F}^C$ for some $\mathcal{F}$. Frame definition, since it quantifies over all valuations, provide a much deeper structural description of the frame (and is impervious to the contingencies of a particular valuation).

But this also seems to be impossible for our basic example of $C = \mathsf{OR}$.

Conjecture 1
The class $\mathsf{Frame}^{\mathsf{OR}}$-$\mathsf{BISIM}$ is not definable.

The above is stated as a conjecture because we do not endeavour to prove it here: I believe it can be proved using a more elaborate version of the proof of Theorem 4.3, but the details which must be checked are too extensive (and the topic deserves separate treatment). The main issue here, it turns out, is that frame-level satisfaction is *too* restrictive: requiring $\mathcal{F}^{\mathsf{OR}} \models \varphi$ for all $\mathcal{F}$ is a high bar, so the only formulas validated by *all* OR-augmented frames are ones which are generic enough for other Π^{or}-frames to validate. Note that this is the opposite problem from the model case: there, it was too *easy* to validate formulas. But the result is the same: no single set Δ suffices. So we must look elsewhere.

[13] Or actually more so in Sect. E

5 Refined Frame Theory

5.1 Idea

Our woes in the previous section both arise from not exercising the correct level of control over the valuations. In the model case, we were able to prove $\mathsf{DTM}^{\mathsf{OR}}$-$\mathsf{BISIM}$ undefinable by carefully designing a particular DTM (with a very specific valuation) to have a $\mathcal{L}^{\mathsf{or}}_{\Box\bigcirc}$ theory *like* a member of $\mathsf{DTM}^{\mathsf{OR}}$-$\mathsf{BISIM}$ even though it wasn't a member of that class. Such annoying contingencies are just a feature of trying to define classes of models. When working at the *frame* level, we had the opposite problem: by quantifying over *all* valuations on some $\mathcal{F}^C$, we refuted too many formulas – again leading to undefinability.

But if we reflect more carefully, it makes more sense why the latter failure happens. Notice that the $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ theory of a Σ-frame is defined to be those formulas which that frame validates for *every* valuation. So the $\mathcal{L}^{\mathsf{or}}_{\Box\bigcirc}$-theory of some frame $\mathcal{F}^{\mathsf{OR}}$ is given to be the set of those formulas φ which $(\mathcal{F}^{\mathsf{OR}}, V)$ validates for *every* $V : \Phi \to \mathcal{P}(\left|\mathcal{F}^{\mathsf{OR}}\right|)$. But this is casting too broad a net: there are many valuations on $\mathcal{F}^{\mathsf{OR}}$ which don't make sense to study if we're trying to understand how the OR program constructor works. In particular, this notion of the frame's "theory" takes into account valuations which assign $(x, 0)$ and $(x, 1)$ different truth values for some primitive propositions. This violates how the program constructor is supposed to work: $(x, 0)$ and $(x, 1)$ are supposed to be indistinguishable "copies" of the same world x, but augmented with data to nondeterministically resolve a single binary choice – this doesn't work if they *are* distinguishable, by primitive propositions no less. Moreover, a valuation which makes such an assignment would never arise in the process of applying program constructors at the model level: for any $x \in |\mathfrak{M}|$, $(x, 0)$ and $(x, 1)$ have the exact same valuations in $\mathfrak{M}^{\mathsf{OR}}$ by definition (not by definition of OR specifically, but rather more generally as specified *for all program constructors* in Defn. 2.1).So what we pursue in this section is a notion of the frame's "theory" which excludes these kinds of valuations.

But how do we express this kind of "restriction" on what valuations are allowed to be considered in the frame's theory? In particular, what we were trying to do with the definition results earlier was carve the class of OR-augmented models (or frames) out of an ambient class (the classes of all Π^{or}-DTMs and Π^{or}-frames, respectively) – what ambient class are we trying to define this class of "frames but only with certain valuations" structures out of? What does it mean for such a structure to be bisimilar to one produced by the program constructor? These are all questions which we must address in developing this notion of theory. We'll find that they all admit satisfying answers, and moreover equip us with conceptual tools to better apprehend dynamic topological structures generally, and OR-augmented, $\mathsf{U}\omega$-augmented, and $\mathsf{U}\infty$-augmented frames specifically.

5.2 Formal Details

Let us make explicit the kind of restriction on valuations we will impose. What we will do is "package" another piece of data along with a Σ-frame which limits what valuations may be attached to this frame. More specifically, we will supplement our frame $\mathcal{G}$ with an equivalence relation $\mathcal{R}$ on the state space of $\mathcal{G}$ (which satisfies certain properties). This will encode a restriction on valuations by stipulating which worlds must get the same valuation: worlds related by $\mathcal{R}$ must agree on all primitive propositions. Before we get there, let us begin by defining what kind of equivalence relation is suitable for this purpose.

Definition 5.1

Let $\mathcal{G} = (Y, \tau_Y, \{g_\sigma\})$ be a Σ-frame for some $\Sigma \supseteq \Pi$.[14] A relation $\mathcal{R} \subseteq Y \times Y$ is said to constitute a **refinement relation** on $\mathcal{G}$ if the following conditions are met.

(1) $\mathcal{R}$ is an equivalence relation.

(2) For all $U \in \tau_Y$, the set $\mathcal{R}(U)$[15] is open: $\mathcal{R}(U) \in \tau_Y$.

(3) For all $\pi \in \Pi$, if $y, y' \in Y$ are such that $y\mathcal{R}y'$ and $g_\pi(x)$ is defined, then $g_\pi(y')$ is defined and $g_\pi(y)\mathcal{R}g_\pi(y')$.

The idea of a refinement relation is to relate together worlds which are equivalent as far as Π-programs go, but may be different when it comes to programs in $\Sigma \setminus \Pi$. Of course, our ultimate interest is in $\Sigma = \Pi^c$, in which case a refinement relation encodes worlds that behave identically as far as primitive programs are concerned but differ on constructed programs – like $(x, 0)$ and $(x, 1)$. The purpose of (1) is just to ensure that $\mathcal{R}$ encodes a notion of equivalence (from the standpoint of set theory). We shall see the relevance of (2) later, but it will guarantee that the $\mathcal{R}$ relation interfaces with the topology properly. (3) is the interesting one: it states that the interpretation of Π-programs coheres with the notion of equivalence $\mathcal{R}$ stipulates: if two worlds are equivalent per $\mathcal{R}$, then the interpretation of π is equivalent in the two worlds (either $\|\pi\|$ is undefined at both, or defined at both and the resulting states are $\mathcal{R}$-equivalent too). Note, however, that this definition makes no requirement about the interpretation of programs $\sigma \in \Sigma \setminus \Pi$: the interpretation g_σ of such a σ may interface with $\mathcal{R}$ in any manner whatsoever, and usually will not satisfy the requirement that (3) makes for $\pi \in \Pi$.

Of course, program construction furnishes our leading example.

Example 5.1

Let $\mathcal{F}$ be a Π-frame and C a program constructor. We define the relation $\mathcal{R}^C_{\mathcal{F}} \subseteq \left|\mathcal{F}^C\right| \times \left|\mathcal{F}^C\right|$ by the following equation: for all $x \in |\mathcal{F}|$,

$$\left[(x, \gamma)\right]_{\mathcal{R}^C_{\mathcal{F}}} = \{x\} \times \Gamma \subseteq \left|\mathcal{F}^C\right|.$$

In words: each state (x, γ) of $\mathcal{F}^C$ is related to all worlds (x, γ') which have the same first coordinate but whatever second coordinate.

This is a refinement relation on $\mathcal{F}^C$. (1) It's straightforward to check that this is an equivalence relation. (2) is slightly more involved,[16] and (3) follows quickly from the definition of $\|\pi\|_{\mathcal{F}^C}$ for $\pi \in \Pi$.

What the relation $\mathcal{R}^C_{\mathcal{F}}$ does is indicate which worlds of $\mathcal{F}^C$ are "copies" of the same world of $\mathcal{F}$. Recall that the state space of $\mathcal{F}^C$ is $|\mathcal{F}| \times \Gamma$, i.e. it consists of Γ-many copies of each state of $\mathcal{F}$. $\mathcal{R}^C_{\mathcal{F}}$ relates exactly those worlds are the same x paired with (perhaps) different γs. Remember: the point of this relation is to tell us which worlds should always get the same valuation when we're trying to figure out the theory of $\mathcal{F}^C$ for class definition purposes. We said earlier that we want

[14] In the present work, we're interested in $\Sigma = \Pi^c$ for various c.

[15] Recall $\mathcal{R}(U)$ is the set of all those $y' \in Y$ such that $y\mathcal{R}y'$ for some $y \in U$

[16] Picking U open in $\mathcal{F}^C$, we can see that $\mathcal{R}^C_{\mathcal{F}}(U)$ is open through the following reasoning: it can be proved in general that for any topological spaces (X, τ_X) and (Γ, τ_Γ), the first projection function $\mathsf{pr}_1 : X \times \Gamma \to X$ is *open* (for any open $U \subseteq X \times \Gamma$, its image $\mathsf{pr}_1(U) \subseteq X$ is in τ_X). So $\mathsf{pr}_1(U) \in \tau_X$. So then observe that

$$\mathcal{R}^C_{\mathcal{F}}(U) = \mathsf{pr}_1(U) \times \Gamma$$

which must be open by definition of the product topology, since $\mathsf{pr}_1(U)$ is open in τ_X (by the above) and Γ is open in τ_Γ (by the axioms of topology).

to only include those valuations which arise from model-level program construction. As we'll see shortly, $\mathcal{R}^C_{\mathcal{F}}$ achieves this perfectly.

Next, let us define *refined frames*.

Definition 5.2

A **refined (Σ-)frame** is a pair $(\mathcal{F}, \mathcal{R})$, where $\mathcal{F}$ is a Σ-frame (for $\Sigma \supseteq \Pi$) and $\mathcal{R}$ is a refinement relation on $\mathcal{F}$.

This is the kind of object that a C-augmented frame is: not only does it consist of the frame $\mathcal{F}^C$, but it comes included with some information (the relation $\mathcal{R}^C_{\mathcal{F}}$) indicating which worlds ought to get the same valuation (thereby restricting the kinds of valuations which can be placed on this frame). Indeed, we will think of frame-level program construction as producing a refined frame.

Note 5.1

Henceforth, we will think of a program constructor C, when applied to a Π-frame $\mathcal{F}$, as producing not merely a Π^c-frame $\mathcal{F}^C$, but a *refined* Π^C*-frame*, $(\mathcal{F}^C, \mathcal{R}^C_{\mathcal{F}})$.

By making the definition much more abstract than just program construction (a refined frame need not arise from program construction), we have set up an ambient class of structures which the class of C-augmented structures may be *defined out of*. Defining a class of models by $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ formulas designates a subclass of the general class of Σ-DTMs, defining a class of frames carves out a subclass of the class of all Σ-frames, and we may indeed give a notion of "definition" for refined frames. But in order to do so, we must have a notion of "theory" for refined frames. Introducing this notion will allow us to make good on our repeated assertion that the refinement relation $\mathcal{R}$ of a refined frame "restricts the allowed valuations" and "declares which worlds should get the same valuation".

Definition 5.3

Given a refined Σ-frame $(\mathcal{G}, \mathcal{R})$ we say that a valuation $V : \Phi \to \mathcal{P}(|\mathcal{G}|)$ **respects** $\mathcal{R}$ if, for all $(x, x') \in \mathcal{R}$ and all $p \in \Phi$,

$$x \in V(p) \quad \iff \quad x' \in V(p)$$

i.e. $\mathcal{R}$-related worlds get the same valuation.

Given a formula $\varphi \in \mathcal{L}_{\Box\bigcirc}(\Sigma)$, we say that $(\mathcal{G}, \mathcal{R})$ **validates** φ and write $(\mathcal{G}, \mathcal{R}) \models \varphi$ just in case $(\mathcal{G}, V) \models \varphi$ for all V which respect $\mathcal{R}$.

We write $\mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathcal{G}, \mathcal{R})$ to denote the set of all φ validated by $(\mathcal{G}, \mathcal{R})$. We adopt the same notational conventions for $\mathrm{Th}_{\Box\bigcirc}$ as before.[17]

First, to see the relevance of this notion of a valuation "respecting" a refinement relation, note that

$$\left\{(\mathcal{F}^{\mathsf{OR}}, V) \ : \ V \text{ respects } \mathcal{R}^{\mathsf{OR}}_{\mathcal{F}}\right\} = \left\{\mathfrak{M}^{\mathsf{OR}} \ : \ \mathfrak{M} = (\mathcal{F}, v) \text{ for some } v : \Phi \to \mathcal{P}(|\mathcal{F}|)\right\}$$

i.e. $\mathcal{R}^{\mathsf{OR}}_{\mathcal{F}}$-respecting valuations are exactly those valuations which arise from applying the OR program constructor to Π-DTMs. So the valuations which $\mathcal{R}^{\mathsf{OR}}_{\mathcal{F}}$ permits to be included in the "theory" of $(\mathcal{F}^C, \mathcal{R}^{\mathsf{OR}}_{\mathcal{F}})$ are those which could've arisen by applying the program constructor at the model level, establishing the theory of $(\mathcal{F}^C, \mathcal{R}^{\mathsf{OR}}_{\mathcal{F}})$ as the notion of "theory" which best reflects how program constructors work.

In general, this is how we wish to think of refined frames: they're frames which have a refinement relation which specifies restrictions on valuations. This "restriction" is implemented in the definition

[17] E.g. writing $\mathrm{Th}^{\mathsf{or}}_{PDL}(\mathcal{F}^{\mathsf{OR}}, \mathcal{R}^{\mathsf{OR}}_{\mathcal{F}})$ to mean the set of all those $\varphi \in \mathcal{L}^{\mathsf{or}}_{PDL}$ such that $(\mathcal{F}^{\mathsf{OR}}, V) \models \varphi$ for every valuation V on $\mathcal{F}^{\mathsf{OR}}$ which respects $\mathcal{R}^{\mathsf{OR}}_{\mathcal{F}}$.

of theory: $(\mathcal{G}, \mathcal{R}) \models \varphi$ means that any model built atop $\mathcal{G}$ *whose valuation respects* $\mathcal{R}$ will validate φ. This is a *weaker* notion of satisfaction than the standard notion ($\mathcal{G} \models \varphi$) we covered before, because there are fewer valuations allowed, hence less potential for countermodels. Indeed, the standard frame theory corresponds to the weakest refinement relation on a frame.

Note 5.2

If $\mathcal{R}$ is the identity relation $\{(x, x) : x \in |\mathcal{F}|\}$ on $\mathcal{F}$ (which, note, is a refinement relation). Then every valuation respects $\mathcal{R}$, so the associated theory $\mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathcal{F}, \mathcal{R})$ is just the standard frame theory, $\mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathcal{F})$.

For another example of refinement relations, where to get them, and how they behave, see Sect. E (this material is not strictly necessary for anything that follows).

Let us take a pause to summarize. We have introduced the notion of a refinement relation, which is a kind of relation that can be placed on a frame which coheres with the structure of the frame appropriately. A frame equipped with such a relation is a refined frame. Refined frames have their own notion of theory, which only takes into account valuations which respect the refinement relation. Alongside this general development, we developed the example of refined frames being produced by program construction: a program constructor C produces a refined Π^c-frame $(\mathcal{F}^C, \mathcal{R}^C_{\mathcal{F}})$.

What we have developed thus far is quite a bit too general. There's not too much we can say about refined frames or about program constructors at this level of generality, as this level of generality admits too broad a class of examples – including many which are poorly-behaved or pathological in various ways. So let us develop some special properties refined frames (and program constructors) can have, and see what further results they allow us to obtain. In particular, we will be interested in articulating in abstract terms those properties common to the examples listed in the previous chapter (which may not hold of all program constructors), and the special properties of the refined frames they produce. We do so in the concept of *outcome-dense refinement classes*.

Understanding the precise meaning of this condition is not essential, and can be overlooked if desired. What is important is to understand is that it is a condition which may or may not hold of a given refined frame, and that it does hold of all the main examples of program constructors we're working with, by virtue of the structural properties *of the program constructor*.

Definition 5.4

A refined Σ-frame $(\mathcal{G}, \mathcal{R})$ is said to have **outcome-dense refinement classes** if the following is true for every $\sigma \in \Sigma$ and every $w \in |\mathcal{G}|$:[18]

- **If $\|\sigma\|_{\mathcal{G}}(w)$ is defined:** For every open set U such that $[w]_{\mathcal{R}} \cap U \neq \emptyset$, there exists some $w' \in [w]_{\mathcal{R}} \cap U$ such that
$$(\|\sigma\|_{\mathcal{G}}(w), \|\sigma\|_{\mathcal{G}}(w')) \in \mathcal{R}$$
- **If $\|\sigma\|_{\mathcal{G}}(w)$ is not defined:** For every open set U such that $[w]_{\mathcal{R}} \cap U \neq \emptyset$, there exists some $w' \in [w]_{\mathcal{R}} \cap U$ such that $\|\sigma\|_{\mathcal{G}}(w')$ is undefined.

One of the most significant things common to all the examples of program constructors we gave is that the topology τ_Γ on the "constructor state space" Γ is indiscrete. Epistemologically, this corresponded to the constructor state being completely opaque to the agent. Mathematically,

[18] Some intuition on this definition: the goal of this definition is to facilitate Theorem 5.1, which guarantees that $\mathcal{R}$-related worlds have the same PDL theories. So we need to guarantee that if $\Diamond \bigcirc_\sigma \psi$ holds in one world of the equivalence class, then it holds in all worlds of the equivalence class. This can be defeated if there's some open set U nontrivially intersecting some $[w]_{\mathcal{R}}$ where none of the worlds of U validate $\bigcirc_\sigma \psi$, but where some world $w' \in [w]_{\mathcal{R}} \setminus U$ *does* satisfy $\bigcirc_\sigma \psi$. Then, $\Diamond \bigcirc_\sigma \psi$ holds at w', but won't hold in $[w]_{\mathcal{R}} \cap U$, because U witnesses $\Box\neg \bigcirc_\sigma \psi$. A refined frame with outcome-dense refinement classes is one where exactly this kind of thing cannot occur.

it means (among other things) that they all produce refined frames which have outcome-dense refinement classes.

Example 5.2

Let C be an "*indiscrete* program constructor": one whose corresponding topological space (Γ, τ_Γ) is indiscrete ($\tau_\Gamma = \{\emptyset, \Gamma\}$). It turns out that any refined Π^c-frame $(\mathcal{F}^C, \mathcal{R}^C_{\mathcal{F}})$ produced with such a program constructor has outcome-dense refinement classes.[19][20]

Since all the examples we're interested in have this property, results relying on that property are of interest to us. Assuming outcome-dense refinement classes indeed allows us to prove quite a nice result, one which makes precise the claim that worlds x, x' which are $\mathcal{R}$-related in a refined Σ-frame $(\mathcal{G}, \mathcal{R})$ are "Π-equivalent, but may differ on $\Sigma \setminus \Pi$". It also shows that the PDL theories (even for the full program set Σ) are invariant across $\mathcal{R}$-equivalence classes in such a refined frame – setting up the point (to be made in a second) that they actually constitute relational models of PDL at the same time. The case where $\Sigma = \Pi^c$ is summarized in Figure 5.1.

Theorem 5.1 (Refinement-PDL)

Suppose $(\mathcal{G}, \mathcal{R})$ is a refined Σ-frame which has outcome-dense refinement classes and let V be any valuation on $\mathcal{G}$ which respects $\mathcal{R}$. Then, for any $x, x' \in |\mathcal{G}|$ such that $x\mathcal{R}x'$, the following equalities hold.

$$\mathrm{Th}_{\Box\bigcirc}(\Pi; (\mathcal{G}, V), x) = \mathrm{Th}_{\Box\bigcirc}(\Pi; (\mathcal{G}, V), x')$$

$$\mathrm{Th}_{PDL}(\Sigma; (\mathcal{G}, V), x) = \mathrm{Th}_{PDL}(\Sigma; (\mathcal{G}, V), x')$$

In words: $\mathcal{R}$-related worlds agree on all agent DTL formulas about *primitive* programs (i.e. $\varphi \in \mathcal{L}_{\Box\bigcirc}(\Pi)$), but when it comes to the expanded set of programs Σ, we can only guarantee that $\mathcal{R}$-related worlds agree on formulas in the more restricted PDL language, $\mathcal{L}_{PDL}(\Sigma)$. Viewed slightly differently, we might say that each $\mathcal{R}$-equivalence class seems to have a single "opinion" about which $\mathcal{L}_{\Box\bigcirc}(\Pi)$ formulas are true and which $\mathcal{L}_{PDL}(\Sigma)$ formulas are true (and, where these overlap, its "opinion" is consistent). So even though $\mathcal{R}$-related worlds may have different $\mathcal{L}_{\Box\bigcirc}(\Sigma)$ theories (indeed, this is how we achieve nondeterminism), they manage to all agree on these two important fragments.

This reflects a deeper fact about these structures: a refined Σ-frame is essentially a Π-frame with a relational semantics for $\Sigma \setminus \Pi$ "on top". To see the first part of this, suppose $(\mathcal{F}, \mathcal{R})$ is a

[19] This is because – if C is indiscrete – then for any given $x \in |\mathcal{F}|$, there cannot possibly be an open set $U \subseteq \left|\mathcal{F}^C\right| = |\mathcal{F}| \times \Gamma$ containing (x, γ) but not (x, γ'): the set $\{x\} \times \Gamma \subseteq \left|\mathcal{F}^C\right|$ is either contained in, or disjoint from U, or else U is not open (this can easily be proven from the definition of the product topology and the definition of "indiscrete"). Call this fact (*).

We can then prove the appropriate condition from Defn. 5.4 as follows: suppose we have a $\sigma \in \Sigma$, a $w = (x, \gamma) \in \left|\mathcal{F}^C\right|$ and an open U such that

$$\left[(x, \gamma)\right]_{\mathcal{R}T^C_{\mathcal{F}}} \cap U \neq \emptyset. \tag{**}$$

Recalling the definition of $\mathcal{R}^C_{\mathcal{F}}$ from Example 5.1 and Example E.2, we have that

$$\left[(x, \gamma)\right]_{\mathcal{R}^C_{\mathcal{F}}} = \{x\} \times \Gamma.$$

Therefore, combining (*) with (**), we see that $\{x\} \times \Gamma \subseteq U$. Therefore (x, γ) itself is in U, so the requirements of Defn. 5.4 can be achieved trivially by putting $w' = w$.

[20] Note that we have specifically proved here that $w \in U$ under the assumption that (Γ, τ_Γ) is indiscrete. Absent this assumption, the open sets U which intersect nontrivially with $[w]_{\mathcal{R}}$ may not contain w itself, and so, if we're claiming that $(\mathcal{F}, \mathcal{R})$ has outcome-dense refinement classes, we must demonstrate that $[w]_{\mathcal{R}} \cap U$ contains some world w' in which σ has the same outcome (up to $\mathcal{R}$-equivalence) as it does in w (or no defined outcome, if that's what happens in w).

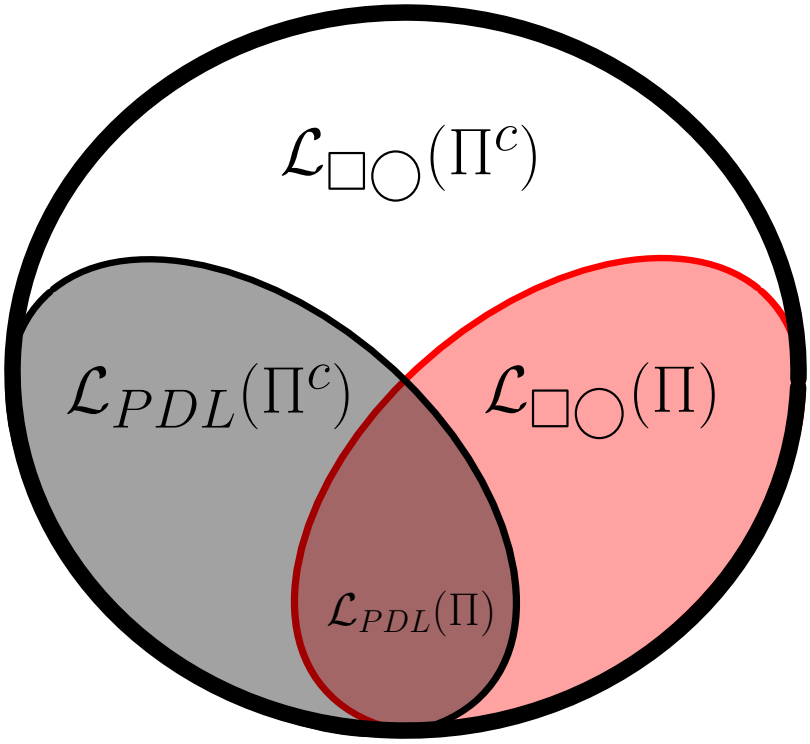


Figure 5.1: Languages interpreted by a refined Π^c-frame
The red-shaded region ($\mathcal{L}_{\Box\bigcirc}(\Pi)$) is invariant across the $\mathcal{R}$-equivalence classes of a refined Π^c-frame, and so too is $\mathcal{L}_{PDL}(\Pi^c)$, provided those refinement classes are outcome-dense. In general, however, $\mathcal{R}$-related worlds are still allowed to have different $\mathcal{L}_{\Box\bigcirc}(\Pi^c)$ theories.

refined Σ-frame, and $\mathfrak{M} = (\mathcal{F}, V)$ is a Σ-DTM whose valuation V respects the refinement relation $\mathcal{R}$. Since $\mathfrak{M}$ is a Σ-DTM, $\mathfrak{M}$ is also a Π-DTM, by virtue of the fact that $\Sigma \supseteq \Pi$. But when viewed as a Π-DTM, $\mathfrak{M}$ may contain a lot of "redundancy", i.e. worlds which play the exact same role in the Π-structure, and which could be "glued together" without changing the $\mathcal{L}_{\Box\bigcirc}(\Pi)$-theories of any of the surrounding worlds. *Which* worlds we could identify is encoded by the refinement relation $\mathcal{R}$: the point of $\mathcal{R}$, recall, is to say which worlds, from the perspective of the Π-structure, are the same.

Indeed, we could explicitly collapse an entire $\mathcal{R}$-equivalence class into a single world, and we would obtain a structurally equivalent (read: Π-bisimilar) model to what we started with. This process is made explicit with the definition of *quotients*.

Definition 5.5
Let $(\mathcal{G}, \mathcal{R})$ be a refined Σ-frame, with $\mathcal{G} = (Y, \tau_Y, \{g_\sigma\})$. The **quotient** of $\mathcal{G}$ by $\mathcal{R}$ (denoted $\mathcal{G}/\mathcal{R}$) is a Π-frame with

- State space $Y/\mathcal{R}$, the set of $\mathcal{R}$-equivalence classes
- The quotient topology of τ_Y by $\mathcal{R}$

- Interpretations of $\pi \in \Pi$ given by:

$$\|\pi\|_{\mathcal{G}/\mathcal{R}}([y]) = \begin{cases} [g_\pi(y)] & \text{if } g_\pi(y) \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Note that the last definition is well-defined, by the 3rd defining condition of refinement relations.

If $\mathfrak{N} = (\mathcal{G}, V)$ is a Σ-DTM whose valuation V respects $\mathcal{R}$, then we may also define $\mathfrak{N}/\mathcal{R}$ to be the frame $\mathcal{G}/\mathcal{R}$ with the corresponding valuation $V_{\mathcal{R}}$:

$$[y]_{\mathcal{R}} \in V_{\mathcal{R}}(p) \quad \text{iff} \quad y \in V(p)$$

which is well-defined by the assumption that V respects $\mathcal{R}$.

Note 5.3

Given any refined Σ-frame $(\mathcal{G}, \mathcal{R})$, the quotient function

$$Q_{\mathcal{R}} : y \mapsto [y]_{\mathcal{R}}$$

is a surjective Π-bisimulation[21] $\mathcal{G} \simeq_\Pi \mathcal{G}/\mathcal{R}$.

So what quotienting a frame $\mathcal{G}$ by a refinement relation $\mathcal{R}$ does is identify the worlds which $\mathcal{R}$ considers equivalent. The conditions of a refinement relation make it so the topology and the interpretations of Π-programs survive this operation. The condition of V *respecting* $\mathcal{R}$ guarantees that V can be quotiented in a well-defined manner too. However, quotients need not preserve the structure of programs in $\Sigma \setminus \Pi$: we do not make a requirement like (3) from Defn. 5.1 for thse programs, so $\|\sigma\|_{\mathcal{G}/\mathcal{R}}$ need not be well-defined (hence why $\mathcal{G}/\mathcal{R}$ is simply a Π-frame).

We said above that we can view refined Σ frames as "Π-frames, with a relational interpretation of $\Sigma \setminus \Pi$ atop". The "Π-frame" aspect refers to the fact that refined frames may be quotiented – a process which "boils $\mathcal{G}$ down"[22] to just its Π-structure, removing any redundancy of worlds which $\mathcal{R}$ has marked as unnecessary for interpreting Π. The "relational interpretation of $\Sigma \setminus \Pi$ atop" refers to this point: if we try to carry the interpretation of $\sigma \in \Sigma \setminus \Pi$ through the operation of quotienting, the resulting interpretation $\|\sigma\|_{\mathcal{G}/\mathcal{R}}$ may not be a partial function. In particular, if there were worlds y, y' of $\mathcal{G}$ such that $y\mathcal{R}y'$ but where $g_\sigma(y)$ was not related to $g_\sigma(y')$ by $\mathcal{R}$ (this setup is only precluded by Defn. 5.1 for Π-programs), this would create an ambiguity if we tried to define $\|\sigma\|_{\mathcal{G}/\mathcal{R}}$ as in Defn. 5.5:

$$\|\sigma\|_{\mathcal{G}/\mathcal{R}}([y]) := [g_\sigma(y)] \neq [g_\sigma(y')] =: \|\sigma\|_{\mathcal{G}/\mathcal{R}}([y']).$$

Instead, using this definition for elements of $\Sigma \setminus \Pi$, in general, produces binary relations rather than partial functions. This is why we describe this as a "relational interpretation of $\Sigma \setminus \Pi$": though it's implemented using dynamic topological logic ($\mathcal{G}$ itself is still very much a Σ-frame, and interprets all of Σ as partial functions), we can view it as implementing a relational-PDL-style logic over and above the interpretation of Π-programs. Let's make this more explicit.

[21] Note that this function is automatically total and surjective (by basic properties of equivalence relations) and moreover is continuous (which, recall, is equivalent to (Back) from Defn. 4.8) by definition: the quotient topology is defined as the finest topology making this function continuous. The second condition of Defn. 5.1 guarantees that this function satisfies (Forth). As mentioned, the third condition of Defn. 5.1 guarantees that the given definition of $\|\sigma\|_{\mathcal{G}/\mathcal{R}}$ is well-defined, and this definition immediately guarantees that $Q_{\mathcal{R}}$ respects Π (in the sense required by Defn. 4.8).

[22] This was the original motivation for the word "refinement".

Definition 5.6

Given a refined Σ-frame $(\mathcal{G}, \mathcal{R})$ and a DTM $\mathfrak{M} = (\mathcal{G}, V)$ where V respects $\mathcal{R}$, define the **Σ-collapse** of $\mathfrak{M}$ by $\mathcal{R}$, denoted $\mathfrak{M} /\!\!/ \mathcal{R}$, to be the $(\Sigma \setminus \Pi)$-relational model whose

- state space $X = \{[y] \ : \ y \in |\mathcal{G}|\}$ is the set $|\mathcal{G}| / \mathcal{R}$ of $\mathcal{R}$-equivalence classes,
- relations R_σ for $\sigma \in \Sigma \setminus \Pi$ are defined by

$$[x] R_\sigma [z] \quad \text{iff} \quad \|\sigma\|_{\mathcal{G}} (x') = z' \text{ for some } x' \in [x], z' \in [z],$$

- and whose valuation $v : \Phi \to \mathcal{P}(X)$ is given by[23]

$$[x] \in v(p) \quad \text{iff} \quad x \in V(p)$$

We'll also write $\mathcal{G} /\!\!/ \mathcal{R}$ to denote the *relational frame* $(X, \{R_\sigma\})$, where X and R_σ are as above.

So the idea here is that $\mathcal{G}/\mathcal{R}$ and $\mathcal{G} /\!\!/ \mathcal{R}$ tell us everything we need to know about $\mathcal{G}$: the quotient $\mathcal{G}/\mathcal{R}$ conveys the Π-logic of $\mathcal{G}$, and the collapse $\mathcal{G} /\!\!/ \mathcal{R}$ tells us how the rest of the programs are interpreted (it gives us the relational structure which of non-Π programs that $(\mathcal{G}, \mathcal{R})$ is "implementing" in the dynamic-topological structure). As we'll see in the next section, the collapse will prove quite helpful understanding the structure of augmented DTMs, especially because we're intending them to be dynamic-topological analogues of relational PDL. We can calculate the collapse of frames augmented with our program constructors, and get back the familiar relational structure we were aiming for. Let's turn our attention to doing this.

6 Characterization

6.1 Refined Frame Theory and Program Construction

Before returning to our main impetus (characterization), let's begin by using the tools of the previous section to study C-augmented frames. First, we can instantiate Theorem 5.1 to be about augmented frames, and obtain this result.

Corollary 6.0.1

If C is an indiscrete program constructor (in the sense of Example 5.2), then for all Π-DTMs $\mathfrak{M} = (\mathcal{F}, V)$, and all worlds w, w' of $\mathfrak{M}^C$ such that $w \mathcal{R}^C_{\mathcal{F}} w'$,

$$\mathrm{Th}_{PDL}(\Pi^c; \mathfrak{M}^C, w) = \mathrm{Th}_{PDL}(\Pi^c; \mathfrak{M}^C, w').$$

This says that the different "copies" of the same $\mathfrak{M}$-world in a C-augmented DTM $\mathfrak{M}^C$ (with the assumption that C is indiscrete) have the same PDL theories, even about constructed programs. So the whole $\mathcal{R}^C_{\mathcal{F}}$ equivalence class has a single opinion of the truth value of every $\mathcal{L}^c_{PDL}$ formula, kind of like a world in a relational PDL model (the point of collapses is to make this connection a bit more explicit). Moving along, we can use quotients to express an important property of augmented DTMs.

Proposition 6.1

For any program constructor C and any Π-frame $\mathcal{F}$,

$$\mathcal{F} \simeq_\Pi \mathcal{F}^C / \mathcal{R}^C_{\mathcal{F}}.$$

Indeed,

$$\mathcal{F} \cong_\Pi \mathcal{F}^C / \mathcal{R}^C_{\mathcal{F}}.$$

[23] This is well-defined by the assumption that V respects $\mathcal{R}$.

This just verifies what we've been saying all along: that an augmented frame $\mathcal{F}^C$ interprets primitive programs in the exact same way as $\mathcal{F}$. The program constructor makes Γ-many copies of each state of its input frame $\mathcal{F}$, but all these copies are Π-equivalent (as encoded by the refinement relation $\mathcal{R}^C_{\mathcal{F}}$), and indeed we can identify all the copies to obtain our original frame back.

As mentioned, a refined Σ-frame is a DTL implementation of some relational logic on top of a Π-frame. Our interest was to capture the relational logic of PDL program constructors in the dynamic topological setting. Collapses give us the tools we need to argue we have faithfully "captured" this logic.

Example 6.1

Let $\mathcal{F}$ be any Π-frame, and consider the $(\Pi^{\cup} \setminus \Pi)$-relational frame

$$\mathcal{F}^{\mathsf{U}\omega} /\!\!/ \mathcal{R}^{\mathsf{U}\omega}_{\mathcal{F}}.$$

The relations $R_{\sigma_0 \cup \sigma_1}$ of this relational frame tell us what $\mathsf{U}\omega$ is "doing". What $\mathsf{U}\omega$ is doing is implementing the PDL $\cup$ constructor in DTL: for all $\sigma_0, \sigma_1 \in \Pi^{\cup}$,

$$R_{\sigma_0 \cup \sigma_1} = R_{\sigma_0} \cup R_{\sigma_1}.$$

Likewise for $\mathsf{U}\infty$.

What does this mean? Well, notice again what condition (3) of Defn. 5.1 says: primitive programs, when executed from $\mathcal{R}$-related worlds, must have $\mathcal{R}$-equivalent behavior. So, for primitive programs π, the entire $\mathcal{R}$-equivalence class is unified: either $\|\pi\|$ is undefined at every point of the class, or it's defined across the $\mathcal{R}$-class and ends up in a single $\mathcal{R}$-class (you cannot have $\mathcal{R}$-equivalent worlds which have non-$\mathcal{R}$-equivalent results). So a primitive program π "connects" each $\mathcal{R}$-equivalence class to *at most one* other $\mathcal{R}$-class. But not so for constructed programs: it's possible to have $\mathcal{R}$-related worlds w, w' where $\|\pi_0 \cup \pi_1\|(w)$ and $\|\pi_0 \cup \pi_1\|(w')$ are *not* $\mathcal{R}$-related. So non-primitive programs connect an $\mathcal{R}$-equivalence class to possibly many other $\mathcal{R}$-equivalence classes. What this result shows is that, if we use $\mathsf{U}\omega$ to interpret $\cup$, then an $\mathcal{R}^{\mathsf{U}\omega}_{\mathcal{F}}$-equivalence class E is "connected" (in this sense) by $\sigma_0 \cup \sigma_1$ to any equivalence class E is connected to by σ_0 or by σ_1, i.e. it is implementing *union*.

Ultimately, this is what was going on "behind the scenes" of the results of Subsect. 3.3: $\mathsf{U}\omega$ and $\mathsf{U}\infty$ were achieving the same thing in DTL that the $\cup$ program constructor does in relational PDL. Hence why the PDL theories corresponded closely. A similar analysis of some of the other constructors we defined (e.g. $\mathsf{STAR}\infty$) will show that they too represent classic relational constructors in the dynamic-topological setting.

Finally, back to the initial purpose of refined frame theory: defining program constructors in the object language. We have established a general class of objects (the class of all refined Π^c-frames) and located our augmented objects (refined frames of the form $(\mathcal{F}^C, \mathcal{R}^C_{\mathcal{F}})$) within it. Let's articulate and define this subclass.

As before, our classes will be closed under a notion of bisimilarity. A bisimulation of refined frames is simply a bisimulation between the frames which respects the refinement relations on its domain and codomain in the proper way. Our ultimate goal is to provide a set Δ of $\mathcal{L}^c_{\Box\bigcirc}$-formulas such that, if a refined Π^c-frame $(\mathcal{G}, \mathcal{R})$ validates all of Δ, then $(\mathcal{G}, \mathcal{R})$ is "essentially" of the form $(\mathcal{F}^C, \mathcal{R}^C_{\mathcal{F}})$ for some $\mathcal{F}$. That means we want $\mathcal{R}$ to play the same role in shaping the structure and logic of $\mathcal{G}$ as $\mathcal{R}^C_{\mathcal{F}}$ does for $\mathcal{F}^C$. Therefore, the notion of "bisimilarity of refined frames" we'll use includes a tight correspondence between the refinement relations.

Definition 6.1

Let $(\mathcal{F}, \mathcal{R})$ and $(\mathcal{F}', \mathcal{R}')$ be refined Σ-frames. We say that s is a **Σ-bisimulation of refined frames** between $(\mathcal{F}, \mathcal{R})$ and $(\mathcal{F}', \mathcal{R}')$ – denoted $s : (\mathcal{F}, \mathcal{R}) \twoheadrightarrow_{\Sigma} (\mathcal{F}', \mathcal{R}')$ – if the following conditions hold.

- s is a Σ-bisimulation from $\mathcal{F}$ to $\mathcal{F}'$
- For all worlds x, y of $\mathcal{F}$ and all worlds x', y' of $\mathcal{F}'$ such that xsx' and ysy'

$$x\mathcal{R}y \quad \Longleftrightarrow \quad x'\mathcal{R}'y'$$

As usual, we furthermore write $s : (\mathcal{F}, \mathcal{R}) \simeq_\Sigma (\mathcal{F}', \mathcal{R}')$ to indicate that s is surjective.

So this is the notion of two refined frames being "essentially the same". So, as mentioned, we want an object-language condition which is strong enough to guarantee that every refined frame which validates it is "essentially" an augmented frame. At long last, this is the notion of characterization.

Definition 6.2

Let C be a program constructor. A set $\Delta \subseteq \mathcal{L}^c_{\Box\bigcirc}$ is said to **characterize** C if the following two conditions hold.

- For every Π-frame $\mathcal{F}$, $(\mathcal{F}^C, \mathcal{R}^C_\mathcal{F}) \models \Delta$
- For every refined Π^c-frame $(\mathcal{G}, \mathcal{R})$, if $(\mathcal{G}, \mathcal{R}) \models \Delta$, then there exists a Π-frame $\mathcal{F}$ such that

$$(\mathcal{G}, \mathcal{R}) \simeq_{\Pi^c} (\mathcal{F}^C, \mathcal{R}^C_\mathcal{F})$$

In short: Δ characterizes C just in case Δ picks out exactly those refined frames which (up to bisimulation) arise from application of the program constructor. We'll spend the rest of this section making a few comments and claims about this definition, and then the subsequent two sections will give a full characterization of OR, and a discussion of how to characterize (and some of the further difficulties which arise when we try to characterize) $\mathsf{U}\omega$ and $\mathsf{U}\infty$.

We conclude this section by giving a more convenient form for the first condition.

Proposition 6.2

For any C, the following are equivalent.

- For every Π-frame $\mathcal{F}$, $(\mathcal{F}^C, \mathcal{R}^C_\mathcal{F}) \models \Delta$
- For every Π-DTM $\mathfrak{M}$, $\mathfrak{M}^C \models \Delta$

The goal of this claim is simply to make the notion of "characterization" more simple and concrete by eliminating the reference to refinement relations. This statement holds in light of the observation above that $\mathcal{R}^C_\mathcal{F}$-respecting valuations are exactly those valuations that C-augmented *DTMs* are equipped with. So, when we go to prove a characterization result, this is often how we'll go about it. The main thrust of the definition of characterization is the second requirement: to stipulate a kind of converse of this statement, at the proper level of universality.

Accordingly, the second condition of Defn. 6.2 will prove significantly more cumbersome, for the same reasons that completeness results are generally more challenging to prove than soundness results. For the characterization proofs we'll be doing, the Π-frame $\mathcal{F}$ which we'll use to witness this second condition is $\mathcal{G}/\mathcal{R}$. We can think of this as "taking away the interpretations of constructed programs" (taking $\mathcal{G}/\mathcal{R}$ to obtain a mere Π-frame) and then "adding it back" (by augmenting). If the $\mathcal{G}$ we started with was of the right form to begin with (which is what we're trying to show), then this should produce a bisimilar result. This is what the condition requires.

With that, let us now proceed to carry out a full example, which will hopefully make more clear how all these parts fit together.

7 Or

The point of this section is to provide a concrete example of how to conduct a characterization proof. OR is a simple enough program constructor, but, as we've seen, managed to evade definition using more traditional means (thus proving the efficacy of this notion of characterization). We make some definitions for convenience of notation; the role of Maybe_0, Maybe_1, etc. will become clear as the proof proceeds.

Definition 7.1

For any $\pi_0, \pi_1 \in \Pi$ and $p \in \Phi$, define

$$\mathsf{Maybe}_0(p, \pi_0, \pi_1) \equiv \bigcirc_{\pi_0} p \leftrightarrow \bigcirc_{\pi_0 \text{ or } \pi_1} p$$
$$\mathsf{Maybe}_1(p, \pi_0, \pi_1) \equiv \bigcirc_{\pi_1} p \leftrightarrow \bigcirc_{\pi_0 \text{ or } \pi_1} p$$

and

$$\mathsf{Only}_0(p, \pi_0, \pi_1) \equiv (\bigcirc_{\pi_0} p \wedge \neg \bigcirc_{\pi_1} p \wedge \bigcirc_{\pi_0 \text{ or } \pi_1} p) \vee (\neg \bigcirc_{\pi_0} p \wedge \bigcirc_{\pi_1} p \wedge \neg \bigcirc_{\pi_0 \text{ or } \pi_1} p)$$
$$\mathsf{Only}_1(p, \pi_0, \pi_1) \equiv (\neg \bigcirc_{\pi_0} p \wedge \bigcirc_{\pi_1} p \wedge \bigcirc_{\pi_0 \text{ or } \pi_1} p) \vee (\bigcirc_{\pi_0} p \wedge \neg \bigcirc_{\pi_1} p \wedge \neg \bigcirc_{\pi_0 \text{ or } \pi_1} p)$$

Definition 7.2

Let χ_{OR} be the set of $\mathcal{L}^{\mathsf{or}}_{\Box\bigcirc}$ formulas consisting of all instances of the following, where

- $\pi_0, \pi_1, \ldots$ varies over Π
- p varies over Φ
- σ varies over Π^{or}.

(OR-Typ)	$\mathsf{Maybe}_0(p, \pi_0, \pi_1) \vee \mathsf{Maybe}_1(p, \pi_0, \pi_1)$	Typicality
(OR-Reg0)	$\mathsf{Only}_0(p, \pi_0, \pi_1) \rightarrow \mathsf{Maybe}_0(q, \pi_2, \pi_3)$	Regularity 0
(OR-Reg1)	$\mathsf{Only}_1(p, \pi_0, \pi_1) \rightarrow \mathsf{Maybe}_1(q, \pi_2, \pi_3)$	Regularity 1
(OR-Real0)	$p \wedge \mathsf{Only}_1(q, \pi_0, \pi_1) \rightarrow \Diamond(p \wedge \mathsf{Only}_0(q, \pi_0, \pi_1))$	Realization 0
(OR-Real1)	$p \wedge \mathsf{Only}_0(q, \pi_0, \pi_1) \rightarrow \Diamond(p \wedge \mathsf{Only}_1(q, \pi_0, \pi_1))$	Realization 1
(OR-Persist0)	$\mathsf{Only}_0(p, \pi_0, \pi_1) \rightarrow \bigcirc_\sigma \top \rightarrow \bigcirc_\sigma \mathsf{Maybe}_0(q, \pi_2, \pi_3)$	Persistence 0
(OR-Persist1)	$\mathsf{Only}_1(p, \pi_0, \pi_1) \rightarrow \bigcirc_\sigma \top \rightarrow \bigcirc_\sigma \mathsf{Maybe}_1(q, \pi_2, \pi_3)$	Persistence 1

Theorem 7.1

χ_{OR} characterizes the OR program constructor.

Proof. —

We state one part as a lemma.

Lemma 7.2

For every Π-frame $\mathcal{F}$,

$$(\mathcal{F}^{\mathsf{OR}}, \mathcal{R}^{\mathsf{OR}}_{\mathcal{F}}) \models \chi_{\mathsf{OR}}$$

which is proved by just checking each of the axioms.

For the other direction, let $\mathcal{G} = (X, \tau, \{g_\sigma\}_{\sigma \in \Pi^{\mathsf{or}}})$ be a Π^{or}-frame and $\mathcal{R}$ a refinement relation on $\mathcal{G}$, and suppose $(\mathcal{G}, \mathcal{R}) \models \chi_{\mathsf{OR}}$. To begin, we introduce some convenient terminology.

Definition 7.3

Given an $w \in |\mathcal{G}|$ and σ, σ', we write

$$\|\sigma\|_{\mathcal{G}}(w) \approx_{\mathcal{R}} \|\sigma'\|_{\mathcal{G}}(w')$$

if either (a) both sides are undefined, or (b) both sides are defined and are $\mathcal{R}$-related.

Definition 7.4

Now, given a world x of $\mathcal{G}$ and an ordered pair (π_0, π_1) of primitive programs,

- x is a *0-world for* (π_0, π_1)

$$\|\pi_0\|(x) \approx_{\mathcal{R}} \|\pi_0 \text{ or } \pi_1\|(x)$$

- x is a *1-world for* (π_0, π_1)

$$\|\pi_1\|(x) \approx_{\mathcal{R}} \|\pi_0 \text{ or } \pi_1\|(x)$$

We'll see the significance of these definitions as we proceed with the proof. Note that these are not mutually exclusive: x can only be both a 0-world and a 1-world for (π_0, π_1) if $\|\pi_0\|(x) \approx_{\mathcal{R}} \|\pi_1\|(x)$.

We establish the desired result by proving three successive claims:

1. For every x and every (π_0, π_1), x is either a 0-world or a 1-world (or both) for (π_0, π_1);
2. Every x is either a 0-world for *every* pair of programs (in which case we call x simply "a 0-world") or a 1-world for every pair of programs (a "1-world"), or both;
3. The relation relating each 0-world x of $\mathcal{G}$ to $([x], 0)$ of $(\mathcal{G}/\mathcal{R})^{\mathsf{OR}}$ and each 1-world y to $([y], 1)$ (where $[x]$ is the $\mathcal{R}$-equivalence class of x) constitutes a Π^{or}-bisimulation between $(\mathcal{G}, \mathcal{R})$ and $((\mathcal{G}/\mathcal{R})^{\mathsf{OR}}, \mathcal{R}^{\mathsf{OR}}_{\mathcal{G}/\mathcal{R}})$.

Thus $\mathcal{G}/\mathcal{R}$ will suffice as a witness for the second requirement of characterization.

7.1 Claim 1

We state the first claim as a lemma:

Lemma 7.3

For any world x and any primitive programs π_0, π_1, if

$$((\mathcal{G}, v), x) \models (\bigcirc_{\pi_0} p \leftrightarrow \bigcirc_{\pi_0 \mathsf{or} \pi_1} p) \vee (\bigcirc_{\pi_1} p \leftrightarrow \bigcirc_{\pi_0 \mathsf{or} \pi_1} p)$$

for all v which respect $\mathcal{R}$, then x is either a 0-world or a 1-world (or both) for (π_0, π_1)

Since the given formula is (OR-Typ), part of χ_{OR} (and thus $(\mathcal{G}, v)$ is assumed to validate it at every world for every valuation v which respects $\mathcal{R}$), we conclude that every world is either a 0-world, a 1-world, or both, for each pair of programs (π_0, π_1). For notation, we define a function

$$\mathsf{PreType} : X \times \Pi \times \Pi \to \{\{0\}, \{1\}, \{0, 1\}\}$$

by putting $0 \in \mathsf{PreType}(x, \pi_0, \pi_1)$ if x is a 0-world for (π_0, π_1), and likewise for 1. The name comes from the fact that we are assigning each world a "type" (either 0, 1, or both) according to the behavior of or there. But this is the "pre-type", since it relies on the choice of programs π_0, π_1 – we now show that each world has a well-defined "type" over all pairs of programs.

7.2 Claim 2

Begin by noticing that, for each fixed world x, exactly one of the following must be true

- For every pair of programs (π, π'), $\mathsf{PreType}(x, \pi, \pi') = \{0, 1\}$
- There is some pair of programs (π, π') where $\mathsf{PreType}(x, \pi, \pi') \neq \{0, 1\}$

In the first case, claim 2 is already done: x is a 0-world for every pair of programs and is a 1-world for every pair of programs. So assume the second case: without loss of generality, assume x is a 0-world for (π_0, π_1), but *is not* a 1-world for (π_0, π_1). In order to establish claim 2, we must now prove that x is a 0-world for *every* pair of programs. In symbols:

$$(\exists \pi_0, \pi_1)(\mathsf{PreType}(x, \pi_0, \pi_1) = \{0\}) \quad \Longrightarrow \quad (\forall \pi_2, \pi_3)(0 \in \mathsf{PreType}(x, \pi_2, \pi_3))$$

So we have that $\|\pi_0\|(x) \approx_{\mathcal{R}} \|\pi_0 \text{ or } \pi_1\|(x)$. So let's case on the two possibilities from Defn. 7.3.

First handle the case where $\|\pi_0\|(x), \|\pi_0 \text{ or } \pi_1\|(x)$ are defined and $\mathcal{R}$-related. Assume for contradiction that (π_2, π_3) is such that x is not a 0-world for (π_2, π_3). So either (a) only one of $\|\pi_2\|$, $\|\pi_2 \text{ or } \pi_3\|$ is defined at x, or (b) both are defined but $\|\pi_2\|(x)$ is not $\mathcal{R}$-related to $\|\pi_2 \text{ or } \pi_3\|(x)$. Now, consider the following formula, which is straightforward logical consequence of (OR-Reg0) (and, as the reader can check, is validated anywhere (OR-Reg0) is validated):

$$(\bigcirc_{\pi_0} p \wedge \neg \bigcirc_{\pi_1} p \wedge \bigcirc_{\pi_0 \text{or} \pi_1} p) \rightarrow (\bigcirc_{\pi_2} q \leftrightarrow \bigcirc_{\pi_2 \text{or} \pi_3} q). \qquad \text{(OR-Reg0+)}$$

We will obtain a contradiction by defining an $\mathcal{R}$-respecting valuation which refutes this formula at x. Put $v(q)$ to be the $\mathcal{R}$-equivalence class of whichever of $\|\pi_2\|(x)$ and $\|\pi_2 \text{ or } \pi_3\|(x)$ is defined (for case (a)) or to the equivalence class of $\|\pi_2\|(x)$ (if in case (b)), and put $v(p)$ to the $\mathcal{R}$-equivalence class of $\|\pi_0\|(x)$ (which, recall, is also the $\mathcal{R}$-equivalence class of $\|\pi_0 \text{ or } \pi_1\|(x)$). Note that this valuation respects $\mathcal{R}$. Then, x validates the antecedent of (OR-Reg0+), because $\|\pi_0\|(x)$ and $\|\pi_0 \text{ or } \pi_1\|(x)$ are in $v(p)$ but $\|\pi_1\|(x)$, if defined, is not (x is not a 1-world). But x does not validate the consequent of (OR-Reg0+): we defined $v(q)$ so that only one of $\|\pi_2\|(x)$, $\|\pi_2 \text{ or } \pi_3\|(x)$ is in it. Thus we obtain the desired contradiction.

The proof where $\|\pi_0\|(x)$ and $\|\pi_0 \text{ or } \pi_1\|(x)$ is undefined is basically the same: put $v(p)$ to the $\mathcal{R}$-equivalence class of $\|\pi_1\|(x)$ – which must be defined since x is not a 1-world – and define $v(q)$ in the same fashion as above. This will defeat the "negative" version of (OR-Reg0+) (which also follows from (OR-Reg0)):

$$(\neg \bigcirc_{\pi_0} p \wedge \bigcirc_{\pi_1} p \wedge \neg \bigcirc_{\pi_0 \text{or} \pi_1} p) \rightarrow (\bigcirc_{\pi_2} q \leftrightarrow \bigcirc_{\pi_2 \text{or} \pi_3} q). \qquad \text{(OR-Reg0-)}$$

We can conduct the analogous proof when x is instead a 1-world but not a 0-world for (π_0, π_1) and get that x must be a 1-world for every pair of programs. So claim 2 is finished.

So we have seen that either x is either a 0-world for all pairs of programs, a 1-world for all pairs of programs, or perhaps both. For the sake of notation, we'll write the function

$$\mathsf{Type} : X \rightarrow \{\{0\}, \{1\}, \{0, 1\}\}$$

to mark each world with which case it falls into: $0 \in \mathsf{Type}(x)$ iff x is a 0-world for all (π, π'), and likewise for 1. Everything we've proven so far was to demonstrate that Type is well-defined and total.

7.3 Claim 3

Now for the final step: defining a Π-frame $\mathcal{F}$ such that

$$(\mathcal{G}, \mathcal{R}) \simeq_{\Pi^c} (\mathcal{F}^{\mathsf{OR}}, \mathcal{R}_{\mathcal{F}}^{\mathsf{OR}}).$$

To do this, we'll take $\mathcal{F} = \mathcal{G}/\mathcal{R}$, the quotient of $\mathcal{G}$ by $\mathcal{R}$. So the worlds of $(\mathcal{G}/\mathcal{R})^{\mathsf{OR}}$ are pairs $([x], \gamma)$ where $[x]$ is an $\mathcal{R}$-equivalence class of $\mathcal{G}$-worlds, and γ is 0 or 1. For the bisimulation, define s by

$$(x, (A, \gamma)) \in s \quad \iff \quad A = [x] \text{ and } \gamma \in \mathsf{Type}(x)$$

So, if $\mathsf{Type}(x) = \{0\}$, then x is related to $([x], 0)$, if $\mathsf{Type}(x) = \{1\}$, then x is related to $([x], 1)$, and if $\mathsf{Type}(x) = \{0, 1\}$, then x is related to $([x], 0)$ and $([x], 1)$. To complete our proof, we need to check that s satisfies the (σ-Back) and (σ-Forth) conditions for all $\sigma \in \Pi^{\mathsf{or}}$, that it's open and continuous, that it's surjective and total, and that it respects the refinement relations properly. The totality of s (i.e. for all $x \in X$ there's some world w of $(\mathcal{G}/\mathcal{R})^{\mathsf{OR}}$ such that xsw) is a straightforward consequence of the totality of Type. Continuity is also straightforward: an open set in the frame $(\mathcal{G}/\mathcal{R})^{\mathsf{OR}}$ is of the form $U \times \{0, 1\}$ for some U open in $\mathcal{G}/\mathcal{R}$. But notice that

$$\{x \ : \ xsw \text{ for some } w \in U \times \{0, 1\}\} = \{x \ : \ [x] \in U\}$$

But this is just the preimage of U under the quotient map $\mathcal{G} \to \mathcal{G}/\mathcal{R}$, which is necessarily continuous, so this set is open.

Surjectivity, openness, and respect for the Π^{or}-dynamics are more complex, so we include them as separate lemmas. For the sake of convenient notation, let $\mathcal{J} = (\mathcal{G}/\mathcal{R})^{\mathsf{OR}}$.

Lemma 7.4

s is surjective and open

Lemma 7.5

For every $\sigma \in \Pi^{\mathsf{or}}$, every world x of $\mathcal{G}$ and every world w of $\mathcal{J}$,

- If xsw and $\|\sigma\|_{\mathcal{G}}(x)$ is defined, then $\|\sigma\|_{\mathcal{J}}(w)$ is defined and $\|\sigma\|_{\mathcal{G}}(x)$ is related to $\|\sigma\|_{\mathcal{J}}(w)$ by s
- If xsw and $\|\sigma\|_{\mathcal{J}}(w)$ is defined, then $\|\sigma\|_{\mathcal{G}}(x)$ is defined and $\|\sigma\|_{\mathcal{G}}(x)$ is related to $\|\sigma\|_{\mathcal{J}}(w)$ by s

Note 7.1

Proving Lemma 7.4 is why we need to include (OR-Real0) and (OR-Real1). Similarly, Lemma 7.5 makes (OR-Persist0) and (OR-Persist1) necessary.

Thus, we have:

$$s : \mathcal{G} \simeq_{\Pi^{\mathsf{or}}} \mathcal{J}$$

Finally, we note that the bisimulation s interfaces with the refinement relations $\mathcal{R}$ and $\mathcal{R}_{\mathcal{G}/\mathcal{R}}^{\mathsf{OR}}$ on $\mathcal{G}$ and $\mathcal{J}$, respectively (the second condition of bisimulation of refined frames): for any $x, x' \in X$ and any $\gamma \in \mathsf{Type}(x)$, $\gamma' \in \mathsf{Type}(x')$,

$$x \mathcal{R} x' \quad \iff \quad [x] = [x'] \quad \iff \quad ([x], \gamma) \mathcal{R}_{\mathcal{G}/\mathcal{R}}^{\mathsf{OR}} ([x'], \gamma').$$

With that, we conclude:

$$s : (\mathcal{G}, \mathcal{R}) \simeq_{\Pi^{\mathsf{or}}} ((\mathcal{G}/\mathcal{R})^{\mathsf{OR}}, \mathcal{R}_{\mathcal{G}/\mathcal{R}}^{\mathsf{OR}}).$$

□(Theorem 7.1)

And with that we're done! In the conclusion of the chapter, we discuss some similarities with our attempts to characterize $\mathsf{U}\omega$ and $\mathsf{U}\infty$, and the epistemic meaning of the axioms.

8 Union

Finally, we indicate how a characterization of $\mathsf{U}\omega{+}\mathsf{SKIP}$ (recall Example 2.1) and $\mathsf{U}\infty{+}\mathsf{SKIP}$ might proceed. This is not a finished characterization – this task encounters further issues. It's not clear that it's possible to characterize $\mathsf{U}\omega$ and $\mathsf{U}\infty$, with or without SKIP; a weakening of the definition may be required. Nonetheless, all the below formulas are indeed validated by the augmented refined frames for these constructors and begin to detail the fine structure. Throughout, $\pi, \pi', \pi_0, \pi_1, \pi_2$, etc. range over Π; $\sigma, \sigma', \sigma_0, \sigma_1$, etc. range over $\Pi^{\cup}$; and p ranges over Φ.

We use the following abbreviations.

$$\begin{aligned}
\mathsf{Null} &\equiv \neg \bigcirc_{\mathsf{skip}\cup\mathsf{skip}} \top \\
\mathsf{Maybe}_0(p, \pi_0, \pi_1) &\equiv \bigcirc_{\pi_0} p \leftrightarrow \bigcirc_{\pi_0\cup\pi_1} p \\
\mathsf{Maybe}_1(p, \pi_0, \pi_1) &\equiv \bigcirc_{\pi_1} p \leftrightarrow \bigcirc_{\pi_0\cup\pi_1} p \\
\mathsf{Only}_0(p, \pi_0, \pi_1) &\equiv (\bigcirc_{\pi_0} p \wedge \neg \bigcirc_{\pi_1} p \wedge \bigcirc_{\pi_0\cup\pi_1} p) \vee (\neg \bigcirc_{\pi_0} p \wedge \bigcirc_{\pi_1} p \wedge \neg \bigcirc_{\pi_0\cup\pi_1} p) \\
\mathsf{Only}_1(p, \pi_0, \pi_1) &\equiv (\neg \bigcirc_{\pi_0} p \wedge \bigcirc_{\pi_1} p \wedge \bigcirc_{\pi_0\cup\pi_1} p) \vee (\bigcirc_{\pi_0} p \wedge \neg \bigcirc_{\pi_1} p \wedge \neg \bigcirc_{\pi_0\cup\pi_1} p)
\end{aligned}$$

Towards a characterization of $\mathsf{U}\omega + \mathsf{SKIP}$, we have the following axioms. The 'Effect' refers to the corresponding model condition which arises from stipulating the axiom scheme at the level of refined frames.

Axiom Schemes	**Effect**
$\varphi \leftrightarrow \bigcirc_{\mathsf{skip}}\varphi$	$\|\mathsf{skip}\|$ is (essentially) the identity function
$p \wedge \neg\mathsf{Null} \rightarrow \bigcirc_{\mathsf{skip}\cup\mathsf{skip}}p$	Executing $\mathsf{skip} \cup \mathsf{skip}$ keeps you in the same $\mathcal{R}$-equivalence class
$\mathsf{Null}(\pi_0,\pi_1) \vee \mathsf{Maybe}_0(p,\pi_0,\pi_1) \vee \mathsf{Maybe}_1(p,\pi_0,\pi_1)$	For every world x and for every pair of **primitive** programs (π_0,π_1), x is either a "null world", a "0-world", a "1-world", or a "01-world" for (π_0,π_1)
$\mathsf{Only}_0(p,\pi_0,\pi_1) \rightarrow \mathsf{Maybe}_0(p,\pi_2,\pi_3)$ $\mathsf{Only}_1(p,\pi_0,\pi_1) \rightarrow \mathsf{Maybe}_1(p,\pi_2,\pi_3)$	Every world x is either a null-world for every pair of primitive programs, a 0-world for every pair of primitive programs, a 1-world for every pair of primitive programs, or a 01-world for every pair of primitive programs
$p \rightarrow \Diamond(p \wedge \mathsf{Maybe}_0(q,\pi_0,\pi_1))$ $p \rightarrow \Diamond(p \wedge \mathsf{Maybe}_1(q,\pi_0,\pi_1))$ $p \rightarrow \Diamond(p \wedge \mathsf{Null})$	Every $\mathcal{R}$-equivalence class contains a null-world, a 0-world, and a 1-world
$\mathsf{Only}_0(p,\pi_0,\pi_1) \wedge \bigcirc_{\mathsf{skip}\cup\mathsf{skip}} \bigcirc_\sigma q \rightarrow \bigcirc_{\sigma\cup\sigma'}q$ $\mathsf{Only}_1(p,\pi_0,\pi_1) \wedge \bigcirc_{\mathsf{skip}\cup\mathsf{skip}} \bigcirc_{\sigma'} q \rightarrow \bigcirc_{\sigma\cup\sigma'}q$	The "nondeterministic choice" of which program to do for (possibly non-primitive) programs σ,σ' is consistent with the choice for primitive programs

Similarly, towards a characterization of $\mathsf{U}\infty + \mathsf{SKIP}$, we have the following axioms.

Axiom Schemes	**Effect**
$\varphi \leftrightarrow \bigcirc_{\mathsf{skip}}\varphi$	$\|\mathsf{skip}\|$ is (essentially) the identity function
$p \rightarrow \bigcirc_{\mathsf{skip}\cup\mathsf{skip}}p$	Executing $\mathsf{skip} \cup \mathsf{skip}$ keeps you in the same $\mathcal{R}$-equivalence class
$\mathsf{Maybe}_0(p,\pi_0,\pi_1) \vee \mathsf{Maybe}_1(p,\pi_0,\pi_1)$	For every world x and for every pair of **primitive** programs (π_0,π_1), x is either a "0-world", a "1-world", or a "01-world" for (π_0,π_1)
$\mathsf{Only}_0(p,\pi_0,\pi_1) \rightarrow \mathsf{Maybe}_0(p,\pi_2,\pi_3)$ $\mathsf{Only}_1(p,\pi_0,\pi_1) \rightarrow \mathsf{Maybe}_1(p,\pi_2,\pi_3)$	Every world x is either a 0-world for every pair of primitive programs, a 1-world for every pair of primitive programs, or a 01-world for every pair of primitive programs
$p \rightarrow \Diamond(p \wedge \mathsf{Maybe}_0(q,\pi_0,\pi_1))$ $p \rightarrow \Diamond(p \wedge \mathsf{Maybe}_1(q,\pi_0,\pi_1))$	Every $\mathcal{R}$-equivalence class contains a null-world, a 0-world, and a 1-world
$\mathsf{Only}_0(p,\pi_0,\pi_1) \wedge \bigcirc_{\mathsf{skip}\cup\mathsf{skip}} \bigcirc_{\sigma} q \rightarrow \bigcirc_{\sigma\cup\sigma'}q$ $\mathsf{Only}_1(p,\pi_0,\pi_1) \wedge \bigcirc_{\mathsf{skip}\cup\mathsf{skip}} \bigcirc_{\sigma'} q \rightarrow \bigcirc_{\sigma\cup\sigma'}q$	The "nondeterministic choice" of which program to do for (possibly non-primitive) programs σ,σ' is consistent with the choice for primitive programs

The issue that arises for $\mathsf{U}\omega$ is the issue of "nonstandard program constructor states": we can prove that for any $(\mathcal{G},\mathcal{R})$ validating the schemes in the $\mathsf{U}\omega + \mathsf{SKIP}$ table above, the $\mathcal{R}$-classes contain worlds which interpret constructed programs like each of the worlds of $\{0,1\}^*$ (the hidden state space Γ of $\mathsf{U}\omega$). However, it's not clear if there's a way to stipulate in the object language that these are *all* the worlds of the $\mathcal{R}$-class. Thus we have no way of guaranteeing that the relation we construct between $(\mathcal{G},\mathcal{R})$ and $((\mathcal{G}/\mathcal{R})^{\mathsf{U}\omega+\mathsf{SKIP}}, \mathcal{R}_{\mathcal{G}/\mathcal{R}}^{\mathsf{U}\omega+\mathsf{SKIP}})$ – which we're trying to argue is a bisimulation – is continuous or total: the $\mathcal{R}$ class could contain additional worlds not corresponding to any program constructor state of a "genuine" $\mathsf{U}\omega$-augmented frame. So either we need to modify our characterization, or give a weaker notion of 'characterization' which has similar properties but isn't so demanding.

$\mathsf{U}\infty + \mathsf{SKIP}$ suffers from the opposite problem: we don't appear to have a way to require in the object language that $(\mathcal{G},\mathcal{R})$ has all the structure of a $\mathsf{U}\infty$-augmented refined frame. In particular, it doesn't seem that there's any way to require all uncountably-many elements of $\{0,1\}^{\mathbb{N}}$ to have analogues in each $\mathcal{R}$-class, when we only have finite-length formulas to stipulate the existence of such worlds. So $(\mathcal{G},\mathcal{R})$ may validate any Δ we propose, but still not have as much structure as a $\mathsf{U}\infty$-augmented frame. Thus we have no way to guarantee that the relation we construct between $(\mathcal{G},\mathcal{R})$ and $((\mathcal{G}/\mathcal{R})^{\mathsf{U}\omega+\mathsf{SKIP}}, \mathcal{R}_{\mathcal{G}/\mathcal{R}}^{\mathsf{U}\omega+\mathsf{SKIP}})$ is open or surjective.

Definitively resolving these questions will require more work than we're able to do here. But

hopefully doing so stimulates interesting further research, and deepens our knowledge of these structures, their logic, and the phenomena they seek to capture.

Conclusion of Chapter 2

What we have seen in this chapter is a display of the difficulty of articulating the logic of program constructors, and an indication of a possible way to attack this complex problem. Though it is certainly not clear that all interesting program constructors admit characterization (or even if our choice examples of $\mathsf{U}\omega$ and $\mathsf{U}\infty$ do), this notion of characterization enjoyed greater success at assigning meaningful object-language descriptions of program constructor behavior. In particular, it succeeded where model- and frame-definition explicitly failed: the OR constructor. As we'll discuss a bit more in Sect. 9, it is hopefully the case that the definition of characterization given here, even if it does not prove adequate for more elaborate program constructors, represents genuine progress towards understanding program construction and the phenomena it seeks to model.

Let us close with a few observations. A theme that briefly emerged earlier in the chapter was the opposite failures of model- and frame-definition to carve out the class of OR-augmented structures. Model definition of $\mathsf{DTM}^{\mathsf{OR}}$-$\mathsf{BISIM}$ failed principally because it was too *easy* to validate formulas at the level of dynamic topological *models*: by cleverly designing a Π^{or}-DTM (using carefully-chosen, very specific valuations), we could make it "look like" a OR-augmented DTM (as far as $\mathcal{L}^{\mathsf{or}}_{\Box\bigcirc}$ can express) without actually being (bisimilar to) a OR-augmented DTM at all. By doing this, we were able to show that any definable class of DTMs containing all OR-augmented ones must also include such "counterfeits", showing that model definition was not going to be a fruitful approach. This was opposite to the primary obstacle in the frame case: there it was too *hard* to satisfy formulas at the frame level. Therefore, the only formulas φ which were indeed satisfied by all OR-augmented DTMs said little of interest about OR-augmentation. Indeed, such sets of formulas defined classes of frames which were much broader than anything we would want to consider "the class of OR-augmented frames". Opposite problem, same result. To address these issues simultaneously, we needed an intermediate notion, one which provided some degree of quantification over valuations – but not too much. Refined frame theory sits comfortably in that intermediate position (and coheres nicely with the existing theory of bisimulations and program construction – hence our interest in it), but it is by no means the only such notion.

Let us also notice the commonalities between the characterization of OR and the attempted characterizations of $\mathsf{U}\omega$ and $\mathsf{U}\infty$. At a high level, all characterizations must take the form of "positing" structure using the object language: we want to pick Δ such that any $(\mathcal{G}, \mathcal{R})$ validating Δ must possess certain structure. In particular, we needed to guarantee that each of the worlds of $\mathcal{G}$ has constructed behavior matching that of some C-augmented DTM (for whichever C we're concerned with). This was made explicit by the process of assigning "types" in Sect. 7, and presumably a characterization of $\mathsf{U}\omega$ or $\mathsf{U}\infty$ would proceed in the same way.

Going about this, we found the following structural commonalities between OR, $\mathsf{U}\omega$, and $\mathsf{U}\infty$, which – returning to our epistemic interpretation – we can read as basic structural axioms of how coin-flipping situations operate.

- To get started, each made use of a "typicality" axiom scheme which stated that each world was either a "0-world" or a "1-world" (or a "null-world", for $\mathsf{U}\omega$). In a coin-flipping situation, there is some fact of the matter about how the next flip will come up: it will either be heads or tails (or may fail in some way, if we're allowing that as a possibility).

- Each had a "regularity" axiom, which expressed the indifference of the coin to the question of *which* actions are being decided between. If our agent flips a coin while in a certain world

x (where, say, the coin happens to come up heads), then it does not affect the coin flip at all if she's using it to decide between π_0 and π_1, or between π_2 and π_3. Though this axiom is certainly obvious when we think about it, we should consider this a virtue: our mathematical analysis has brought this key fact to light (one which certainly seems true of how we think of coin flips), when it's not at all clear that a purely philosophical reflection on coin-flipping would have identified this as a relevant feature.

- Each has a "realization" axiom, which states that both heads and tails are live possibilities (as far as the agent can possibly know before the flip). Our pretheoretic reflections identified this as an important aspect of coin flipping (we said that a flip without this condition must obviously be rigged), so it is reassuring to see it reflected in the formal results.

Beyond these, the remaining axiom schemes of these characterizations appear to express "implementation details" peculiar to the given constructor. For instance, the (OR-Persist0) and (OR-Persist1) axioms were made to account for the fact that OR only permits one flip, so there needs to be some reflection in the theory of this. But it doesn't seem to say anything essential about coin-flipping. Similarly, some of the axioms we used to try to make sense of $\mathsf{U}\omega$ and $\mathsf{U}\infty$, as well as some of the issues preventing a full characterization, seem to be rooted more in the details of how we represented them mathematically than anything worthwhile philosophically.

So what these three commonalities represent are some features inherent to properly-conducted coin flips, which a knowledgeable agent (usually implicitly and unconsciously) expects to be the case when navigating such a situation. Though modest, we have managed to isolate, analyze, and formalize a piece of common reason. Such is our business.

9 Conclusion and Future Work

In addition to thoroughly exploring the topic of epistemic agents flipping coins, our interest was also to significantly develop dynamic topological logic (specifically this "agent DTL" variety), and hopefully make way for even deeper and more striking results. Let us close by identifying just some of the fascinating questions which might take the present work as their point of departure.

Though we were able to obtain some satisfying results, the project initiated here is by no means finished. The characterizations of $\mathsf{U}\omega$ and $\mathsf{U}\infty$ remain at large. As indicated, it's unclear whether the current notion of characterization will suffice, or whether further modification is needed. Moreover, we did not attempt any systematic development of STAR program constructors (beyond giving a definition as example). Certainly, such program constructors will have their unique difficulties and considerations. I suspect they will admit a useful epistemic interpretation as well, which also deserves development.

We briefly indicated with the idea of a $C+\mathsf{SKIP}$ program constructor that these constructions might be *combined*. If we wish to fully implement PDL program construction in the dynamic-topological setting, this requires the addition of multiple program constructors. It remains to be seen exactly how the "nesting" of program constructors works: what's the difference between $(\mathfrak{M}^{C_1})^{C_2}$ and $(\mathfrak{M}^{C_2})^{C_1}$, and how do they interact? A full account of PDL constructors also will require more development of sequencing and test programs; for instance, it's not clear how to define test programs at the frame level (at least their PDL semantics make essential reference to the valuation), so it's unclear whether "characterization" even makes sense – perhaps we need a different notion for such constructions.

There seems to be opportunities for developing this body of work from the standpoint of the epistemic interpretation: if dynamic topological structures function as models of an agent reasoning in a situation (at least what they're able to do and know), it is perhaps worth investigating what

other features can be encoded. We indicated possibilities for modelling not just the agent's *capacity* for knowledge but rather their *actual* knowledge (using subset space semantics), but this is just the beginning. Can we develop a notion of 'belief' in this setting? Can we expand this to encode multiple agents and possibilities for common knowledge? Can we make our agents more realistic (e.g. not logically omniscient) or make other features more plausible (e.g. account for the possibility of unreliable or contradictory evidence)? These are all questions worth exploring more. Moreover, there is much potential for modelling complex phenomena using program constructors: we said that the execution of $\sigma_0 \cup \sigma_1$ was the agent "*deciding*" whether to do σ_0 or σ_1 – can we give a richer and more systematic account of how program constructors provide a way for the agent to *consider* several actions and *decide* to undertake one of them (or some combination of them)?[24] One might also expand on the somewhat vague intuition that program constructors generally represent "devices" of some kind, and use this framework to develop more of a theory of the capacity of agents to reason about the status of their "tools". This perhaps promises an interesting theory of an agent's epistemic standpoint in relation to the technologies they employ to achieve things in the world (of which coin-flipping is only a very basic example).

We could also expand more on the mathematical aspects of this work. We mentioned that the definition of "program constructor" is quite broad – are there meaningful requirements we can place on program constructors,[25] and can we use these to prove interesting properties about their behavior? We've also seen numerous operations that can be performed on DTMs – quotienting, collapsing, "forgetting" about structure, applying program constructors – much work remains to be done to develop this theory. In particular, we have only just begun to give a mathematical theory of bisimulations and refinement relations – undoubtedly there is much which could be generalized and expanded upon.

Finally, we have mostly neglected the connections to computer science, which formed the original motivation behind PDL. Our use of the term "program constructor" suggests that this presents a theory for how to combine programs into more complex programs. Common constructions in the semantics of programming languages (such as loops and conditionals) seem to be definable as program constructors; does it do us any benefit to understand "programming in dynamic topological models"? Explicitly modelling an agent's capacity for knowing pieces of information as they go about executing programs certainly seems relevant to considerations in cryptography, programming language theory, etc., and I suspect that there might be significant interest in this development.

These are just some possibilities. A central highlight of this work is its dense connections to other disciplines, and there are undoubtedly other connections not listed here. But hopefully this provides a jumping-off point to a wide array of fascinating and invigorating topics.

[24] Do some program constructors make choices which are more "*morally praiseworthy*" or "*right*" than others? Can we make sense of such a thing in this framework?

[25] More sophisticated, perhaps, than the only such property we dealt with: *indiscreteness*.

Bibliography

[1] *1903-The First Flight*. National Park Service, Apr 2015.

[2] Marco Aiello and Johan Benthem. Logical patterns in space. models, simulations, and games. 10 2000.

[3] Marco Aiello, Johan Van Benthem, and Guram Bezhanishvili. Reasoning about space: the modal way. *Journal of Logic and Computation*, 13(6):889–920, 2003.

[4] SN Artemov, JM Davoren, and A Nerode. Modal logics and topological semantics for hybrid systems. 1997.

[5] Adam Bjorndahl. The epistemology of nondeterminism. *Lecture Notes in Computer Science*, pages 145–162, 2018.

[6] Adam Bjorndahl. *Topological Subset Space Models for Public Announcements*, pages 165–186. Springer International Publishing, Cham, 2018.

[7] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[8] Maarten De Rijke and Holger Sturm. Global definability in basic modal logic. *Essays on non-classical logic*, 1:111, 2001.

[9] Michael J Fischer and Richard E Ladner. Propositional dynamic logic of regular programs. *Journal of computer and system sciences*, 18(2):194–211, 1979.

[10] Philip Kremer and Grigori Mints. Dynamic topological logic. *Annals of Pure and Applied Logic*, 131(1-3):133–158, 2005.

[11] J. C. C. McKinsey and Alfred Tarski. The algebra of topology. *Annals of Mathematics*, 45(1):141–191, 1944.

[12] Robin Milner. An algebraic definition of simulation between programs. In *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI'71, pages 481–489, San Francisco, CA, USA, 1971. Morgan Kaufmann Publishers Inc.

[13] J.R. Munkres. *Topology*. Featured Titles for Topology. Prentice Hall, Incorporated, 2000.

[14] David Park. Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI-Conference on Theoretical Computer Science*, pages 167–183, London, UK, UK, 1981. Springer-Verlag.

[15] Maarten De Rijke. Modal model theory. *Annals of Pure and Applied Logic*, 1995.

[16] Nicolas Troquard and Philippe Balbiani. Propositional dynamic logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2019 edition, 2019.

[17] Johan van Benthem and Guram Bezhanishvili. Modal logics of space. In *Handbook of spatial logics*. Springer, 2007.

A Sect. 1 Proofs

(Prop. 1.4)

The axiom scheme (U) defines the class of "union-frames": given a pair $(X, \{R_\sigma\}_{\sigma\in\Pi^\cup})$ where X is a set and $R_\sigma \subseteq X \times X$ for all $\sigma \in \Pi^\cup$,[26] the following are equivalent:

1. $R_{\sigma_0\cup\sigma_1} = R_{\sigma_0} \cup R_{\sigma_1}$ for all $\sigma_0, \sigma_1 \in \Pi^\cup$
2. For all valuation functions $v : \Phi \to \mathcal{P}(X)$,

$$(X, \{R_\sigma\}, v) \models [\sigma_0 \cup \sigma_1]\varphi \leftrightarrow [\sigma_0]\varphi \wedge [\sigma_1]\varphi$$

for all $\sigma_0, \sigma_1 \in \Pi^\cup$ and all $\varphi \in \mathcal{L}^\cup_{PDL}(\Pi)$.

Proof. —

To prove the first statement implies the second: letting v be arbitrary and putting $M = (X, \{R_\sigma\}, v)$, note that

$$(M, x) \models [\sigma]\varphi \quad \text{iff} \quad R_\sigma(x) \subseteq [\![\varphi]\!]_M$$

where $[\![\varphi]\!]_M$, as usual, denotes the set of worlds $x' \in X$ which validate φ. So then

$$\begin{aligned}
(M, x) \models [\sigma_0 \cup \sigma_1]\varphi &\iff R_{\sigma_0\cup\sigma_1}(x) \subseteq [\![\varphi]\!]_M \\
&\iff R_{\sigma_0}(x) \cup R_{\sigma_1}(x) \subseteq [\![\varphi]\!]_M && \text{(assumption)} \\
&\iff R_{\sigma_0}(x) \subseteq [\![\varphi]\!]_M \text{ and } R_{\sigma_1} \subseteq [\![\varphi]\!]_M && \text{(set theory)} \\
&\iff (M, x) \models [\sigma_0]\varphi \text{ and } (M, x) \models [\sigma_1]\varphi \\
&\iff (M, x) \models [\sigma_0]\varphi \wedge [\sigma_1]\varphi && \text{(relational semantics for } \wedge\text{)}
\end{aligned}$$

and our result follows.

For the other direction, we prove the contrapositive: that if $R_{\sigma_0\cup\sigma_1} \neq R_{\sigma_0} \cup R_{\sigma_1}$, then there's some v such that $(X, \{R_\sigma\}, v)$ refutes (U). If σ_0, σ_1 are such that $R_{\sigma_0\cup\sigma_1} \neq R_{\sigma_0} \cup R_{\sigma_1}$, then there are two possibilities: either there is some (x, x') in $R_{\sigma_0\cup\sigma_1}$ but not in $R_{\sigma_0} \cup R_{\sigma_1}$, or vice versa.

If x and x' are related by $R_{\sigma_0\cup\sigma_1}$ but not $R_{\sigma_0} \cup R_{\sigma_1}$, then, for some $p \in \Phi$, pick v such that $v(p) = R_{\sigma_0}(x) \cup R_{\sigma_1}(x)$. Then, again letting $M = (X, \{R_\sigma\}, v)$, we can check that $(M, x) \models [\sigma_0]p \wedge [\sigma_1]p$ immediately. However, $(M, x) \not\models [\sigma_0 \cup \sigma_1]p$ since p is not true at x' and $xR_{\sigma_0\cup\sigma_1}x'$. So the biimplication of (U) fails. The other case is analogous.

□

(Prop. 1.5)

For any set Π, PDL_0 is a sound and complete axiomatization of $\mathcal{L}_{PDL}(\Pi)$ with respect to the class of all Π-DTMs: for all $\varphi \in \mathcal{L}_{PDL}(\Pi)$,

$$\vdash_{\mathsf{PDL}_0} \varphi \quad \iff \quad \mathfrak{M} \models \varphi \text{ for all } \Pi\text{-DTMs } \mathfrak{M}$$

[26] We'll call such a pair $(X, \{R_\sigma\})$ a "$\Pi^\cup$ relational frame".

Proof. —

The soundness result is proved as Theorem 2 of [5]. That proof also proves the completeness of the class of Π-DTMs with respect to a similar axiom system, by specifying a model transformation which takes a Π-relational PDL model and produces a Π-DTM refuting the same formulas. However, this model transformation assumes that the input model is *serial* ($R_\pi(x)$ is nonempty for all π, x), and produces a serial Π-DTM as output ($\|\pi\|$ is a *total* function for all π). Because of this, the resulting Π-DTMs cannot refute the "seriality axiom scheme" $[\pi]\varphi \to \langle\pi\rangle\varphi$, and therefore this axiom scheme must be included the axiom system.

We drop the assumption of seriality entirely, and give an alternative model transformation which does not assume its input Π-relational model is serial, and, in general, does not produce serial Π-DTMs. Given an arbitrary Π-relational model $M = (X, \{R_\pi\}, v$, define a Π-DTM $\mathfrak{M}$ with

- State space $X \times X^{\mathbb{N}}$: the worlds of $\mathfrak{M}$ are pairs (x, S) where x is a world of M and S is an infinite stream of M-worlds (we use all the same notations as before).
- The topology on $\mathfrak{M}$ is the product topology of the *discrete* topology on X ($\tau_X = \mathcal{P}(X)$) with the *indiscrete* topology on $X^{\mathbb{N}}$ ($\tau_{X^{\mathbb{N}}} = \left\{\emptyset, X^{\mathbb{N}}\right\}$). So the open sets are exactly those of the form $A \times X^{\mathbb{N}}$ for any $A \subseteq X$ (including $A = \emptyset$).
- For $\pi \in \Pi$, the partial function $\|\pi\| : X \times X^{\mathbb{N}} \rightharpoonup X \times X^{\mathbb{N}}$ is given by

$$\|\pi\|(x, x'^\frown xs) = \begin{cases} (x', xs) & \text{if } x' \in R_\pi(x) \\ \text{undefined} & \text{otherwise} \end{cases}$$

  In words: to execute π from (x, S), take the first element x' from the stream S. If it happens to be the case that $xR_\pi x'$, then executing π from (x, S) leads to the $\mathfrak{M}$-world $\|\pi\|(x, S)$, which has M-state x' and has the tail xs of S (S with the x' at the front removed) as its infinite stream. If the first element of S is not in $R_\pi(x)$, then π is undefined at (x, S).
- Valuation $V : \Phi \to \mathcal{P}(X \times X^{\mathbb{N}})$ given by

$$V(p) = v(p) \times X^{\mathbb{N}}$$

So now we claim:[27]

Claim 2

If $\mathfrak{M}$ is the Π-DTM produced from M by the above transformation,

$$\mathrm{Th}_{PDL}(\Pi; M) = \mathrm{Th}_{PDL}(\Pi; \mathfrak{M}).$$

Proof. —

We show that

$$(M, x) \models \varphi \quad \Longleftrightarrow \quad (\mathfrak{M}, (x, S)) \models \varphi \text{ for all } S \in X^{\mathbb{N}}$$

for all x, φ, and the claim follows. Proceed by structural induction on φ.

The case where φ is a primitive proposition is directly guaranteed by the definition of V, and the inductive steps for $\wedge$ and $\neg$ are trivial. Now assume inductively that

$$(M, z) \models \psi \quad \Longleftrightarrow \quad (\mathfrak{M}, (z, t)) \models \psi \text{ for all } S \in X^{\mathbb{N}}$$

[27] This is numbered as "Claim 2" for consistency with the proof of Theorem 3.7 below.

for arbitrary $z \in X$ and some $\psi \in \mathcal{L}_{PDL}(\Pi)$. Let π be arbitrary.

If $(M, x) \models \langle\pi\rangle\psi$, then there must be some $x' \in R_\pi(x)$ such that $(M, x') \models \psi$. By the inductive hypothesis, we have

$$(\mathfrak{M}, (x', S')) \models \psi \text{ for all } S' \in X^{\mathbb{N}}. \tag{*}$$

So now pick arbitrary $xs \in X^{\mathbb{N}}$ and consider the world

$$(x, x'^\frown xs) \in X \times X^{\mathbb{N}}.$$

By definition of $\|\pi\|$ and the fact that $x' \in R_\pi(x)$, we have

$$\|\pi\|\,(x, x'^\frown xs) = (x', xs)$$

But by (*) we get that $(\mathfrak{M}, (x', xs)) \models \psi$ and therefore

$$(\mathfrak{M}, (x, x'^\frown xs)) \models \bigcirc_\pi \psi$$

Now, if $S \in X^{\mathbb{N}}$ is arbitrary, observe that any open set $U \subseteq X \times X^{\mathbb{N}}$,

$$(x, x'^\frown xs) \in U \quad \text{iff} \quad (x, S) \in U$$

by the fact that $\tau_{X^{\mathbb{N}}}$ is indiscrete. Since no open set separates $(x, x'^\frown xs)$ from (x, S) and $(\mathfrak{M}, (x, x'^\frown xs)) \models \bigcirc_\pi \psi$, it follows that

$$(\mathfrak{M}, (x, S)) \models \Diamond \bigcirc_\pi \psi$$

as desired.

If $(M, x) \not\models \langle\pi\rangle\psi$, then all elements of $R_\pi(x)$ (if any) refute ψ. Therefore, for any $S \in X^{\mathbb{N}}$, $\|\pi\|\,(x, S)$ is either undefined or is a $\neg\psi$-world by the inductive hypothesis. Thus $(\mathfrak{M}, (x, S)) \models \neg \bigcirc_\pi \psi$. So, since $\{x\} \times X^{\mathbb{N}}$ is an open set,

$$(\mathfrak{M}, (x, S)) \models \Box\neg \bigcirc_\pi \psi,$$

which is equivalent to

$$(\mathfrak{M}, (x, S) \not\models \Diamond \bigcirc_\pi \psi.$$

Since S is arbitrary, the claim is proved.

□(Claim)

So now we have all we need to prove completeness. If φ is a nontheorem of PDL_0, then, by Prop. 1.3, we can obtain a Π-relational model M with a world x such that

$$(M, x) \not\models \varphi.$$

Apply the above model transformation, and obtain $\mathfrak{M}$. Observe by Claim 1 that

$$(\mathfrak{M}, (x, S)) \not\models \varphi$$

for arbitrary $S \in X^{\mathbb{N}}$. We have refuted an arbitrary nontheorem of PDL_0 on a Π-DTM, so completeness is proved.

□(Prop)

B Sect. 3 Proofs

(Prop. 3.1)
For every Π-DTM $\mathfrak{M}$, every world $x \in |\mathfrak{M}|$, every program constructor C, and every program constructor state $\gamma \in \Gamma$,

$$\mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{M}, x) = \mathrm{Th}_{\Box\bigcirc}\left(\Pi; \mathfrak{M}^C, (x, \gamma)\right)$$

Proof. —
Recall the Π-bisimulation (introduced in Example 4.1) $T^C_{\mathfrak{M}} : \mathfrak{M} \simeq_\Pi \mathfrak{M}^C$ which relates x to each (x, γ). By Theorem 4.2, we obtain the desired result.

□

(Corollary 3.1.1)
For every Π-DTM $\mathfrak{M}$ and every program constructor C,

$$\mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{M}) = \mathrm{Th}_{\Box\bigcirc}\left(\Pi; \mathfrak{M}^C\right).$$

Proof. —
Recall the Π-bisimulation (introduced in Example 4.1) $T^C_{\mathfrak{M}} : \mathfrak{M} \simeq_\Pi \mathfrak{M}^C$. By Corollary 4.2.1, we obtain the desired result.

□

(Corollary 3.1.2)
For all $\mathfrak{M}, C, x, \gamma, \gamma'$,

$$\mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{M}^C, (x, \gamma)) = \mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{M}^C, (x, \gamma'))$$

Proof. —
By two applications of Prop. 3.1,

$$\mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{M}^C, (x, \gamma)) = \mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{M}, x) = \mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{M}^C, (x, \gamma')).$$

□

(Prop. 3.2)
Let $\mathfrak{M}$ be a Π-DTM, C a program constructor, and $\pi \in \Pi$.

$$\|\pi\|_{\mathfrak{M}} \text{ is continuous (open)} \quad \text{iff} \quad \|\pi\|_{\mathfrak{M}^C} \text{ is continuous (open)}$$

Proof. —
As usual, let (X, τ_X) and (Γ, τ_Γ) be the topological spaces underlying $\mathfrak{M}$ and C, respectively.

Assume f_π continuous with respect to τ_X. Pick an arbitrary open set $O \subseteq \left|\mathfrak{M}^C\right|$. By the definition of the product topology, we can write O as

$$O = \bigcup_{i \in I} U_i \times V_i$$

for some sets $U_i \subseteq X$ and $V_i \subseteq \Gamma$ which are open with respect to their respective topologies. But recall how $\|\pi\|_{\mathfrak{M}^C}$ was defined in Defn. 2.1, and observe:

$$\|\pi\|^{-1}_{\mathfrak{M}^C}(U_i \times V_i) = (\|\pi\|^{-1}_{\mathfrak{M}}(U_i)) \times V_i = (f_\pi^{-1}(U_i)) \times V_i.$$

By the continuity of f_π, we get that $f_\pi^{-1}(U_i)$ is open in τ_X, and thus that $f_\pi^{-1}(U_i) \times V_i$ is open with respect to the product topology, for all $i \in I$. Since we've managed to write $\|\pi\|^{-1}_{\mathfrak{M}^C}(O)$ as the union of open sets, it itself is open. Since O was arbitrary, $\|\pi\|_{\mathfrak{M}^C}$ is continuous.

Conversely, supposing that $\|\pi\|_{\mathfrak{M}^C}$ is continuous and picking open $U \subseteq X$, observe that

$$f_\pi^{-1}(U) = \mathsf{pr}_1(\|\pi\|^{-1}_{\mathfrak{M}^C}(U \times \Gamma))$$

where $\mathsf{pr}_1 : X \times \Gamma \to X$ is the first projection function. By the continuity of $\|\pi\|_{\mathfrak{M}^C}$, we get that $\|\pi\|^{-1}_{\mathfrak{M}^C}(U \times \Gamma)$ is open. One can readily check (using similar reasoning about product topologies as above) that pr_1 is open, in the sense that $\mathsf{pr}_1(A)$ is open whenever A is open. Thus we conclude $f_\pi^{-1}(U)$ must be open. Since U was arbitrary, f_π is continuous.

Proving the result about openness instead of continuity is almost the same throughout: just replace all instances of f_π^{-1} with f_π, $\|\pi\|^{-1}_{\mathfrak{M}^C}$ with $\|\pi\|_{\mathfrak{M}^C}$, and appeal instead to the continuity of pr_1, instead of its openness.

□

(Prop. 3.3)
Let C be either $\mathsf{U}\omega$ or $\mathsf{U}\infty$ and suppose $\mathfrak{M}$ is some Π-DTM, w a world of $\mathfrak{M}^C$, φ an $\mathcal{L}^{\cup}_{\Box\bigcirc}$ formula, and $\sigma_0, \sigma_1 \in \Pi^{\cup}$ such that $(\mathfrak{M}^C, w) \models \bigcirc_{\sigma_0}\varphi$ but $(\mathfrak{M}^C, w) \models \bigcirc_{\sigma_1}\neg\varphi$. Then,

$$(\mathfrak{M}^C, w) \models \Diamond \bigcirc_{\sigma_0 \cup \sigma_1} \varphi \wedge \Diamond \bigcirc_{\sigma_0 \cup \sigma_1} \neg\varphi.$$

Proof. —
For concreteness, we prove the result for $\mathsf{U}\infty$; the exact same proof may be used for $\mathsf{U}\omega$.

Recall that $\mathsf{U}\infty$ is based on the indiscrete topology $\tau_\Gamma = \{\emptyset, \Gamma\}$ (where, recall, Γ is $\{0,1\}^{\mathbb{N}}$, so every open set of $\mathfrak{M}^{\mathsf{U}\infty}$ (for any $\mathfrak{M}$) is of the form $U \times \Gamma$ for some open $U \subseteq |\mathfrak{M}|$. Therefore, in order to show that $(\mathfrak{M}^C, (x,\gamma)) \models \Diamond\psi$ for some ψ, it suffices to find some $\gamma' \in \Gamma$ such that

$$(\mathfrak{M}^{\mathsf{U}\infty}, (x, \gamma')) \models \psi.$$

This is because, as we established, there cannot be an open set containing (x,γ) but not (x,γ'), so, since the latter is a ψ-world, every open neighborhood of (x,γ) contains a ψ-world.

In this case, the ψ's we're interested in are $\bigcirc_{\sigma_0\cup\sigma_1}\varphi$ and $\bigcirc_{\sigma_0\cup\sigma_1}\neg\varphi$. Supposing that we have a $w = (x, S)$ as given in the statement of the proposition, observe:

$$\|\sigma_0 \cup \sigma_1\|_{\mathfrak{M}^{\mathsf{U}\infty}}(x, 0^\frown S) = \|\sigma_0\|_{\mathfrak{M}^{\mathsf{U}\infty}}(x, S).$$

But we know that $\|\sigma_0\|_{\mathfrak{M}^{\mathsf{U}\infty}}$ is a φ-world by the assumption that $(\mathfrak{M}^{\mathsf{U}\infty}, (x,S)) \models \bigcirc_{\sigma_0}\varphi$. Therefore, we get

$$(\mathfrak{M}^{\mathsf{U}\infty}, (x, 0^\frown S)) \models \bigcirc_{\sigma_0\cup\sigma_1}\varphi$$

and thus, by the above,

$$(\mathfrak{M}^{\mathsf{U}\infty}, (x,S)) \models \Diamond \bigcirc_{\sigma_0\cup\sigma_1} \varphi.$$

Using the exact same logic for σ_1 and $1^\frown S$, we obtain

$$(\mathfrak{M}^{\mathsf{U}\infty}, (x,S)) \models \Diamond \bigcirc_{\sigma_0\cup\sigma_1} \neg\varphi$$

and we're done.

□

(Lemma 3.4)
Let C be either $\mathsf{U}\omega$ or $\mathsf{U}\infty$. Then, for any Π-DTM $\mathfrak{M}$, any $\sigma_0, \sigma_1 \in \Pi^{\cup}$ and any $\varphi \in \mathcal{L}^{\cup}_{PDL}$,

$$\mathfrak{M}^C \models \Diamond\bigcirc_{\sigma_0}\varphi \vee \Diamond\bigcirc_{\sigma_1}\varphi \rightarrow \Diamond\bigcirc_{\sigma_0\cup\sigma_1}\varphi$$

Proof. —
Pick an arbitrary $w = (x, S)$ such that $(\mathfrak{M}^C, w) \models \Diamond \bigcirc_{\sigma_0} \varphi \vee \Diamond \bigcirc_{\sigma_1} \varphi$. Let's assume without loss of generality that

$$(\mathfrak{M}^C, w) \models \Diamond \bigcirc_{\sigma_0} \varphi,$$

and then the other case is similar. Given what we know about the topology of $\mathfrak{M}^C$, we know that there must be some $S' \in \Gamma$ such that

$$(\mathfrak{M}^C, (x, S')) \models \bigcirc_{\sigma_0}\varphi.$$

Which implies $\|\sigma_0\|_{\mathfrak{M}^C}(x, S')$ is defined and is a φ-world. Observe then that

$$\|\sigma_0 \cup \sigma_1\|_{\mathfrak{M}^C}(x, 0^\frown S') = \|\sigma_0\|_{\mathfrak{M}^C}(x, S')$$

so

$$(\mathfrak{M}^C, (x, 0^\frown S')) \models \bigcirc_{\sigma_0\cup\sigma_1}\varphi.$$

Since every open set containing $(x, 0^\frown S')$ also contains (x, S), we finally get

$$(\mathfrak{M}^C, (x, S)) \models \Diamond \bigcirc_{\sigma_0\cup\sigma_1} \varphi.$$

Since w was arbitrary, the result follows.

□

C Subsect. 3.3 Proofs

(Lemma 3.5)
Let C be either $\mathsf{U}\omega$ or $\mathsf{U}\infty$. Then, for any Π-DTM $\mathfrak{M}$, any $\sigma_0, \sigma_1 \in \Pi^{\cup}$ and any $\varphi \in \mathcal{L}^{\cup}_{PDL}$,

$$\mathfrak{M}^C \models \Diamond\bigcirc_{\sigma_0\cup\sigma_1}\varphi \rightarrow \Diamond\bigcirc_{\sigma_0}\varphi \vee \Diamond\bigcirc_{\sigma_1}\varphi$$

Proof. —
Let $w = (x, S)$ be any world of $\mathfrak{M}^C$ which validates $\Diamond\bigcirc_{\sigma_0\cup\sigma_1}\varphi$. So, by the topology on $\mathfrak{M}^C$ (and particularly the fact that τ_Γ is indiscrete), there must be some $S' \in \Gamma$ such that

$$(\mathfrak{M}^C, (x, S')) \models \bigcirc_{\sigma_0\cup\sigma_1}\varphi.$$

Notice that, if $C = \mathsf{U}\omega$, this precludes the possibility of $S' = \epsilon$. So S' is nonempty[28] – assume without loss of generality that its first element is 0. Consequently,

$$\|\sigma_0 \cup \sigma_1\|_{\mathfrak{M}^C}(x, S') = \|\sigma_0\|_{\mathfrak{M}^C}(x, S'')$$

where S'' is the rest of S'. We therefore get that

$$(\mathfrak{M}^C, (x, S'')) \models \bigcirc_{\sigma_0}\varphi.$$

Since (x, S'') is in every open set containing (x, S), we get

$$(\mathfrak{M}^C, (x, S)) \models \Diamond \bigcirc_{\sigma_0} \varphi$$

and the result follows. The case where the first element of S' is 1 is handled similarly. We established that $S' \neq \epsilon$, so it must have a first element, which must be either 0 or 1. So we've established the desired disjunction.

[28] This is, of course, always true if $C = \mathsf{U}\infty$.

□

(Theorem 3.6)
$\mathsf{PDL}_0 + (\mathsf{U})$ is a sound and complete axiomatization of $\mathcal{L}^{\cup}_{PDL}$ with respect to $\mathsf{DTM}^{\mathsf{U}\omega}$

(Theorem 3.7)
$\mathsf{PDL}_0 + (\mathsf{U})$ is a sound and complete axiomatization of $\mathcal{L}^{\cup}_{PDL}$ with respect to $\mathsf{DTM}^{\mathsf{U}\infty}$

Proof. —

Throughout, C is either $\mathsf{U}\omega$ or $\mathsf{U}\infty$, and Γ is either $\{0,1\}^*$ or $\{0,1\}^{\mathbb{N}}$, as appropriate. For either, $\tau_\Gamma = \{\emptyset, \Gamma\}$. Besides the one point where we mention $\mathsf{U}\omega$ specifically, it won't matter which one C is.

The soundness result is established in Prop. 1.5, Lemma 3.4, and Lemma 3.5.

For completeness, we will coopt the model transformation from before (originally introduced in the proof of Prop. 1.5) to turn a completeness result for relational models (which we already have) into the desired result for augmented DTMs. We'll make use of 'Claim 2' from that proof as well. Our proof breaks into three steps.

1. Take an arbitrary nontheorem φ of $\mathsf{PDL}_0 + (\mathsf{U})$ and refute it on some union-model[29] M
2. Consider M as merely a Π-relational model (i.e. forget about the fact that it interprets constructed programs $\sigma_0 \cup \sigma_1$), then transform it to a Π-DTM $\mathfrak{M}$ in a $\mathcal{L}_{PDL}(\Pi)$-theory-preserving way
3. Argue that M and $\mathfrak{M}^C$ have the same $\mathcal{L}^{\cup}_{PDL}$ theories, and thereby conclude.

Each of these three steps has a corresponding claim, which together are enough to prove the theorem (we prove the claims at the end of the proof). Let's begin. Start with a formula $\varphi \in \mathcal{L}^{\cup}_{PDL}$ which is not a theorem of $\mathsf{PDL}_0 + (\mathsf{U})$. We may then apply the following result.

Claim 1
For every nontheorem φ of $\mathsf{PDL}_0 + (\mathsf{U})$, there exists a union-model M such that

$$M \not\models \varphi.$$

Our ultimate goal is to produce a $\Pi^{\cup}$-DTM $\mathfrak{N}$ (particularly one produced by the program constructor C) which refutes a given nontheorem φ. What we have at this point is a $\Pi^{\cup}$-*relational model* M which refutes φ. So we need to turn M into $\mathfrak{N}$ in a $\mathcal{L}^{\cup}_{PDL}$-theory-preserving way, so $\mathfrak{N}$ refutes φ too. How we'll go about this is first *forgetting* about the interpretation of constructed programs $\sigma_0 \cup \sigma_1$ in M, and just consider it as a Π-relational model. We're not really forgetting, of course: the fact that M is a union-model uniquely determines its interpretation of constructed programs from its interpretation of primitive ones. We'll use this fact later.

The reason we want to think of M as a Π-relational model instead of a $\Pi^{\cup}$-relational model is that we have a theory-preserving way to turn Π-relational models into Π-DTMs: the above-mentioned model transformation. We don't want to "copy over" the full $\Pi^{\cup}$ structure of M via this process and produce a $\Pi^{\cup}$-DTM right away,[30] because we specifically want the interpretation of $\cup$ in the $\Pi^{\cup}$-DTM we produce to come from the program constructor, not M. So, for this step, we'll just produce a Π-DTM to feed into C later. Here's Claim 2 again, though note we've formulated it slightly more strongly (this is what's proved).

[29] See Defn. 1.7.

[30] Though the model transformation is in fact general enough that this would be well-defined. It just wouldn't produce a DTM in DTM^C, which is what we want.

Claim 2

If $\mathfrak{M}$ is the Π-DTM produced from $M = (X, \{R_\pi\}, v)$ via the transformation specified in the proof of Prop. 1.5, then for all $x \in X$ and all $S \in X^{\mathbb{N}}$,

$$\mathrm{Th}_{PDL}(\Pi; M, x) = \mathrm{Th}_{PDL}(\Pi; \mathfrak{M}, (x, S)).$$

Recall, for later reference, that if $M = (X, \{R_\pi\}, v)$, then $|\mathfrak{M}| = X \times X^{\mathbb{N}}$, $V(p) = v(p) \times X^{\mathbb{N}}$ and $\|\pi\|_{\mathfrak{M}}(x, x'^\frown xs) = (x', xs)$ if $x \in R_\pi(x)$ and is undefined otherwise.

So then, as mentioned, we use the program constructor C to interpret all of $\Pi^\cup$ again: if the previous step produced the Π-DTM $\mathfrak{M}$, then $\mathfrak{M}^C$ is a $\Pi^\cup$-DTM, and specifically is a member of DTM^C, the class from which we're trying to find a DTM refuting φ. So if we can just prove that $\mathfrak{M}^C$ does indeed refute φ, then we'll be done.

We'll prove a bit more than that, in fact. Notice that the state space of $\mathfrak{M}^C$ is $X \times X^{\mathbb{N}} \times \Gamma$[31], i.e. it consists of triples (x, S, γ) where $x \in X$ (the state space of M), $S \in X^{\mathbb{N}}$ (used to encode the possible nondeterminism of the Π-programs), and $\gamma \in \Gamma$ (used to resolve $\cup$s). It turns out that the $\mathcal{L}^\cup_{PDL}$ theory only depends on the first coordinate.

Claim 3

For every point x of X, every $S \in X^{\mathbb{N}}$, and every $\gamma \in \Gamma$,

$$\mathrm{Th}_{PDL}(\Pi^\cup; M, x) = \mathrm{Th}_{PDL}(\Pi^\cup; \mathfrak{M}^C, (x, S, \gamma))$$

From there, we're basically done: $M \not\models \varphi$ by hypothesis, so there must be some $x \in X$ such that $(M, x) \models \neg\varphi$. By Claim 3 (and the fact that $\varphi \in \mathcal{L}^\cup_{PDL}$), we have that for arbitrary $S \in X^{\mathbb{N}}$ and $\gamma \in \Gamma$, $(\mathfrak{M}^C, (x, S, \gamma) \models \neg\varphi$. It follows that

$$\mathfrak{M}^C \not\models \varphi$$

and we have completeness. Now to prove the claims.

Proof (Claim 1). —

Section 4 of [7] provides a "strong completeness" result for the $\mathsf{(K)}$ axiom system with respect to the class of all single-modality relational frames, and sketches how to generalize this proof to modal logics of arbitrary "similarity type". Since PDL_0 is the least normal modal logic over this suite of modalities, the strong completeness result applies. Here is that result, instantiated for our circumstances (in our notation):

> For all sets $\Delta \subseteq \mathcal{L}^\cup_{PDL}$ and all formulas $\varphi \in \mathcal{L}_{PDL}$, if $\Delta \models \varphi$,[32] then $\Delta \vdash_{\mathsf{PDL}_0} \varphi$.[33]

This generalizes Prop. 1.3, which is simply the $\Delta = \emptyset$ case. We'll be interested in the case where Δ is the set of all instances of $\mathsf{(U)}$. If we instantiate Δ to be that, then the contrapositive of the above becomes:

> If φ is a nontheorem of $\mathsf{PDL}_0 + \mathsf{(U)}$, then $\Delta \not\models \varphi$.

[31] Technically, $(X \times X^{\mathbb{N}}) \times \Gamma$. We suppress this distinction.

[32] I.e. for all $\Pi^\cup$-relational frames F, all valuations v, and all points x of F, if $((F, v), x) \models \Delta$, then this implies $((F, v), x) \models \varphi$.

[33] I.e. we can deduce φ in the proof system PDL_0, if we're additionally allowed to take the formulas of Δ as axioms.

The antecedent could alternatively be expressed $\Delta \nvdash_{\mathsf{PDL}_0} \varphi$ or $\nvdash_{\mathsf{PDL}_0+\Delta} \varphi$

But what does $\Delta \not\models \varphi$ mean? It means that there are $\Pi^{\cup}$-relational frames F such that $(F, v) \models \Delta$ for every $v : \Phi \to \mathcal{P}(X)$, but where $(F, v) \not\models \varphi$ for some v. But recall Prop. 1.4 (and recall that Δ is the set of all instances of (U)), which gave us that

$$(F, v) \models \Delta \quad \iff \quad F \text{ is a union-frame}$$

Where a "union-frame" was a pair $(X, \{R_\sigma\}_{\sigma \in \Pi^{\cup}})$ where $R_{\sigma_0 \cup \sigma_1} = R_{\sigma_0} \cup R_{\sigma_1}$ for all σ_0, σ_1. Notice, then, that such a "union-frame" equipped with *any* valuation constitutes a union-model (the definition of a union-model – Defn. 1.7 – does not restrict the valuation at all). So, putting this all together (and assuming we have our nontheorem φ on hand, as before), we have:

> There is a union-frame F where $(F, v) \not\models \varphi$.

I.e. there's a union-model refuting φ.

□ (Claim 1)

Proof (Claim 3). —

So begin by observing – in light of Claim 2 and Prop. 3.1, respectively – that the model transformation $M \mapsto \mathfrak{M}$ and the application of the program constructor $\mathfrak{M} \mapsto \mathfrak{M}^C$ both preserve pointwise $\mathcal{L}_{PDL}(\Pi)$ theories:

> For all $x \in X$, all $S \in X^{\mathbb{N}}$, and all $\gamma \in \Gamma$,
>
> $$\mathrm{Th}_{PDL}(\Pi; M, x) = \mathrm{Th}_{PDL}(\Pi; \mathfrak{M}, (x, S)) = \mathrm{Th}_{PDL}(\Pi; \mathfrak{M}^C, (x, S, \gamma)).$$

So this is almost all of Claim 3. It only remains to inductively extend this conclusion to $\langle \sigma_0 \cup \sigma_1 \rangle$. More formally, we assume for some $\psi \in \mathcal{L}^{\cup}_{PDL}$ and some $\sigma_0, \sigma_1 \in \Pi^{\cup}$ that for all z,

- $(M, z) \models \langle \sigma_0 \rangle \psi$ iff $(\mathfrak{M}^C, (z, S, \gamma)) \models \langle \sigma_0 \rangle \psi$ for all $S \in X^{\mathbb{N}}$ and all $\gamma \in \Gamma$ (IH1), and
- $(M, z) \models \langle \sigma_1 \rangle \psi$ iff $(\mathfrak{M}^C, (z, S, \gamma)) \models \langle \sigma_1 \rangle \psi$ for all $S \in X^{\mathbb{N}}$ and all $\gamma \in \Gamma$ (IH2),

and then prove

$$(M, x) \models \langle \sigma_0 \cup \sigma_1 \rangle \psi \quad \text{iff} \quad (\mathfrak{M}^C, (x, S, \gamma)) \models \langle \sigma_0 \cup \sigma_1 \rangle \psi \text{ for all } S, \gamma.$$

To prove the forward direction, assume $(M, x) \models \langle \sigma_0 \cup \sigma_1 \rangle \psi$. So then, since M is a union-model, either $x R_{\sigma_0} x'$ or $x R_{\sigma_1} x'$ for some x' which validates ψ. Without loss of generality, assume $x R_{\sigma_0} x'$. So then we have

$$(M, x) \models \langle \sigma_0 \rangle \psi$$

By (IH1), we get that

$$(\mathfrak{M}^C, (x, S, \gamma)) \models \Diamond \bigcirc_{\sigma_0} \psi \text{ for all } S, \gamma.$$

Now, the open subsets of $X \times X^{\mathbb{N}} \times \Gamma$ are exactly those of the form $U \times X^{\mathbb{N}} \times \Gamma$ for any arbitrary subset $U \subseteq X$ (because X has the discrete topology, whereas the latter two have the indiscrete topology). Subsequently, $\{x\} \times X^{\mathbb{N}} \times \Gamma$ is an open set. Consequently, what

it means for (x, S, γ) to validate $\Diamond \bigcirc_{\sigma_0} \psi$ is that there's some world of the form (x, S', γ') – same first coordinate, possibly different second and third – such that

$$(\mathfrak{M}^C, (x, S', \gamma')) \models \bigcirc_{\sigma_0} \psi.$$

Therefore, we have that $\|\sigma_0\|_{\mathfrak{M}^C}(x, S', \gamma')$ is defined and is a ψ-world. Consider now, if we executed $\sigma_0 \cup \sigma_1$ from $(x, S', 0^\frown\gamma')$. Then recall

$$\|\sigma_0 \cup \sigma_1\|_{\mathfrak{M}^C}(x, S', 0^\frown\gamma') = \|\sigma_0\|_{\mathfrak{M}^C}(x, S', \gamma').$$

So we then obtain that

$$(\mathfrak{M}^C, (x, S', 0^\frown\gamma')) \models \bigcirc_{\sigma_0 \cup \sigma_1} \psi$$

which, since no open set can separate $(x, S', 0^\frown\gamma')$ from (x, S, γ), gives us

$$(\mathfrak{M}^C, (x, S, \gamma)) \models \Diamond \bigcirc_{\sigma_0 \cup \sigma_1} \psi$$

as desired.

Finally, for the other direction, suppose $(\mathfrak{M}^C, (x, S, \gamma)) \models \Diamond \bigcirc_{\sigma_0 \cup \sigma_1} \psi$ for all S, γ. So then, again by what we know about the topology, there must be some S' and some γ' such that

$$(\mathfrak{M}^C, (x, S', \gamma')) \models \bigcirc_{\sigma_0 \cup \sigma_1} \psi.$$

Note that, if C is $\mathsf{U}\omega$, γ cannot be ϵ. Now, there must be a first element of the string/stream γ'. Assume without loss of generality that it's 0. So

$$\|\sigma_0 \cup \sigma_1\|_{\mathfrak{M}^C}(x, S', \gamma') = \|\sigma_0\|_{\mathfrak{M}^C}(x, S', \gamma'')$$

where γ'' is the rest of γ'. So since the left-hand side is a ψ-world by assumption, so too much be the right-hand side. In other words,

$$(\mathfrak{M}^C, (x, S', \gamma'')) \models \bigcirc_{\sigma_0} \psi.$$

Since (x, S', γ'') cannot be separated from any other element of $\{x\} \times X^{\mathbb{N}} \times \Gamma$ by any open set, we have that

$$(\mathfrak{M}^C, (x, S, \gamma)) \models \Diamond \bigcirc_{\sigma_0} \psi \text{ for all } S, \gamma.$$

Apply (IH1) to get

$$(M, x) \models \langle \sigma_0 \rangle \psi$$

so there must be some ψ-world $x' \in X$ such that $x R_{\sigma_0} x'$. Since M is a union-model, this immediately implies $x R_{\sigma_0 \cup \sigma_1} x'$, and so

$$(M, x) \models \langle \sigma_0 \cup \sigma_1 \rangle \psi.$$

The case where the first element of γ' is 1 similarly proves $(M, x) \models \langle \sigma_0 \cup \sigma_1 \rangle \psi$ as well. With that, our induction carries through.

□(Claim 3)

□

D Sect. 4 Proofs

(Prop. 4.1)
Suppose $s \subseteq X \times Y$ is a binary relation between two topological spaces (X, τ_X) and (Y, τ_Y). (1) and (2) are equivalent to each other, and (3) and (4) are equivalent to each other.

(1) s is *continuous*: for every $U' \in \tau_Y$, its preimage $s^{-1}(U') = \{x \in X \ : \ xsy \text{ for some } y \in U'\}$ is in τ_X

(2) s satisfies (Back): If $y \in U' \in \tau_Y$ and xsy, then there is a $U \in \tau_X$ such that $x \in U \subseteq s^{-1}(U')$.

(3) s is *open*: for every $U \in \tau_X$, its image $s(U) = \{y \in Y \ : \ xsy \text{ for some } x \in U\}$ is in τ_Y

(4) s satisfies (Forth): If $x \in U \in \tau_X$ and xsy, then there is a $U' \in \tau_Y$ such that $y \in U' \subseteq s(U)$.

Proof. —
For (1) $\Longrightarrow$ (2): If s is continuous and $y \in U' \in \tau_Y$ and xsy, then put $U = s^{-1}(U')$. Observe that $x \in U$ and $U \in \tau_X$ and, of course, $U \subseteq s^{-1}(U')$.

For (2) $\Longrightarrow$ (1): Suppose s satisfies (Back), then pick $U' \in \tau_Y$. To show $s^{-1}(U')$ is open, pick arbitrary $x \in s^{-1}(U')$, and let y be one element of U' such that xsy (there must be at least one such element, by definition of $s^{-1}(U')$). By (Back), obtain a $U \in \tau_X$ such that $x \in U \subseteq s^{-1}(U')$. So x is in the interior of $s^{-1}(U')$. Since x was arbitrary, every element of $s^{-1}(U')$ must be in the interior, i.e. $s^{-1}(U')$ is open.

For (3) $\Longrightarrow$ (4): If s is open and $x \in U \in \tau_X$ and xsy, then putting $U' = s(U)$ works.

For (4) $\Longrightarrow$ (3): Suppose s satisfies (Forth) and $U \in \tau_X$. Then pick arbitrary $y \in s(U)$, and let x be such that xsy. From (Forth) obtain a $U' \in \tau_Y$ such that $y \in U' \subseteq S(U)$. So y is in the interior of U', and so U' is open.

□

(Theorem 4.2)
If $(\mathfrak{M}, x) \twoheadrightarrow_\Sigma (\mathfrak{N}, y)$, the following equality holds:

$$\mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{M}, x) = \mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{N}, y).$$

Proof. —
The proof is by a straightforward induction on $\varphi \in \mathcal{L}_{\Box\bigcirc}(\Sigma)$. (Base) guarantees that $(\mathfrak{M}, x)$ and $(\mathfrak{N}, y)$ agree on primitive propositions. The $\wedge$ and $\neg$ inductive steps carry through straightforwardly.

For $\Box$, inductively assume that $(\mathfrak{M}, x') \models \psi$ iff $(\mathfrak{N}, y') \models \psi$ for all $(x', y') \in s$. Suppose $(\mathfrak{M}, x) \models \Box\psi$. Then there is an open $U \subseteq [\![\psi]\!]_{\mathfrak{M}}$ containing x. By (Forth), obtain a U' with $y \in U' \subseteq s(U)$. Then observe by the inductive hypothesis that $s(U) \subseteq [\![\psi]\!]_{\mathfrak{N}}$, so $(\mathfrak{N}, y) \models \Box\psi$. The other direction is similar, using (Back) instead of (Forth).

For $\bigcirc_\sigma$, inductively assume that $(\mathfrak{M}, x') \models \psi$ iff $(\mathfrak{N}, y') \models \psi$ for all $(x', y') \in s$. Suppose $(\mathfrak{M}, x) \models \bigcirc_\sigma\psi$. Then $\|\sigma\|_{\mathfrak{M}}(x)$ is defined and is a ψ-world. Use (σ-Forth) and the inductive hypothesis to get that $\|\sigma\|_{\mathfrak{N}}(y)$ is defined and is a ψ-world. Hence $(\mathfrak{N}, y) \models \bigcirc_\sigma\psi$. Again, the other direction is similarly achieved using (σ-Back).

□

(Corollary 4.2.1)
If $\mathfrak{M} \simeq_\Sigma \mathfrak{N}$,

$$\mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{M}) = \mathrm{Th}_{\Box\bigcirc}(\Sigma; \mathfrak{N}).$$

Proof. —
Call the surjective bisimulation witnessing this s. If $\mathfrak{M} \models \varphi$, then $(\mathfrak{M}, x) \models \varphi$ for each world x of $\mathfrak{M}$. For any $y \in s(x)$, we have by Theorem 4.2 that $(\mathfrak{N}, y) \models \varphi$. Since s is surjective, every $y \in |\mathfrak{N}|$ is in $s(x)$ for some x, and we thereby conclude that $\mathfrak{N} \models \varphi$. The opposite direction proceeds similarly, using the totality of s and Theorem 4.2.

□

(Theorem 4.3)
The class $\mathsf{DTM}^{\mathsf{OR}}$-$\mathsf{BISIM}$ is not definable.

Proof. —
As described above, we give some Π^{or}-DTM $\mathfrak{N}_{\mathsf{bad}}$ and some Π-DTM $\mathfrak{P}$ such that

(a) $\mathfrak{P}^{\mathsf{OR}} \in \mathsf{DTM}^{\mathsf{OR}}$-$\mathsf{BISIM}$

(b) $\mathfrak{N}_{\mathsf{bad}} \notin \mathsf{DTM}^{\mathsf{OR}}$-$\mathsf{BISIM}$

(c) $\mathrm{Th}^{\mathsf{or}}_{\Box\bigcirc}(\mathfrak{P}^{\mathsf{OR}}) \subseteq \mathrm{Th}^{\mathsf{or}}_{\Box\bigcirc}(\mathfrak{N}_{\mathsf{bad}})$

And then we've seen how these premises combined contradict the definability of $\mathsf{DTM}^{\mathsf{OR}}$-$\mathsf{BISIM}$. Recall (or notice) that (a) is immediately true by definition of $\mathsf{DTM}^{\mathsf{OR}}$-$\mathsf{BISIM}$.

First, let's define $\mathfrak{N}_{\mathsf{bad}}$. For the purposes of this proof, we'll assume that $\Pi = \{\pi_0, \pi_1\}$, because that's all we need. For Π with more than 2 elements, the remaining programs can be interpreted in whatever manner (it won't matter to our conclusion). We'll also assume that the set Φ of atomic propositions is enumerated $\{p_i\}_{i \in \mathbb{N}}$.

Definition D.1
$\mathfrak{N}_{\mathsf{bad}}$ is the Π^{or}-DTM defined as follows.

- The underlying set N_{bad} of $\mathfrak{N}_{\mathsf{bad}}$ is $\mathcal{P}_{\mathsf{fin}}(\mathbb{N}) \cup \{t_0, t_1\}$, where $\mathcal{P}_{\mathsf{fin}}(\mathbb{N})$ is the set of all finite sets of natural numbers.[34]
- The topology of $\mathfrak{N}_{\mathsf{bad}}$ is just the indiscrete topology on N_{bad}: $\tau_{\mathsf{bad}} = \{\emptyset, N_{\mathsf{bad}}\}$.
- All programs of Π^{or} are undefined at t_0 and t_1. For $S \in \mathcal{P}_{\mathsf{fin}}(\mathbb{N})$,

$$\begin{aligned} \|\pi_0\|(S) &= t_0 \\ \|\pi_1\|(S) &= t_1 \\ \|\pi_i \,\mathsf{or}\, \pi_j\|(S) &= \begin{cases} \|\pi_i\|(S) & \text{if } |S| \text{ is even} \\ \|\pi_j\|(S) & \text{if } |S| \text{ is odd} \end{cases} \end{aligned}$$

I.e. π_0 always goes to t_0, π_1 always goes to t_1, and or-programs are resolved according to whether S has an even or odd number of elements.[35]

[34] We assume $t_0 \neq t_1$ and $t_0, t_1 \notin \mathcal{P}_{\mathsf{fin}}(\mathbb{N})$.
[35] It is for the sake of this definition that we use $\mathcal{P}_{\mathsf{fin}}(\mathbb{N})$, not $\mathcal{P}(\mathbb{N})$.

- t_0 only validates p_0 (i.e. $t_0 \in V_{\mathsf{bad}}(p_0)$, but $t_0 \notin V_{\mathsf{bad}}(p_i)$ for $i > 0$). t_1 does not validate any atomic proposition. For $S \in \mathcal{P}_{\mathsf{fin}}(\mathbb{N})$,

$$S \in V_{\mathsf{bad}}(p_i) \quad \Longleftrightarrow \quad i \in S$$

  So, for instance, $\emptyset$ does not validate any atomic propositions, whereas $\{37, 128\}$ validates only p_{37} and p_{128}.

So we have defined $\mathfrak{N}_{\mathsf{bad}}$, and it is indeed an Π^{or}-DTM. Next, we need our Π-DTM $\mathfrak{P}$. To obtain $\mathfrak{P}$, note that since $\mathfrak{N}_{\mathsf{bad}}$ is a Π^{or}-DTM and $\Pi^{\mathsf{or}} \supseteq \Pi$, we can consider $\mathfrak{N}_{\mathsf{bad}}$ as a Π-DTM by simply "forgetting" that $\mathfrak{N}_{\mathsf{bad}}$ interprets or-programs. Let $U(\mathfrak{N}_{\mathsf{bad}})$ be this Π-DTM. More formally, $U(\mathfrak{N}_{\mathsf{bad}})$ is defined to be the Π-DTM with $\mathcal{P}_{\mathsf{fin}}(\mathbb{N}) \cup \{t_0, t_1\}$ as its underlying set, the indiscrete topology, the same valuation as $\mathfrak{N}_{\mathsf{bad}}$, and the same interpretations of Π-programs as $\mathfrak{N}_{\mathsf{bad}}$. The only difference is that $\mathfrak{N}_{\mathsf{bad}}$ has interpretations for or-programs, but $U(\mathfrak{N}_{\mathsf{bad}})$ does not. We claim that taking $\mathfrak{P}$ to be $U(\mathfrak{N}_{\mathsf{bad}})$ will do the job.

Start with the following claim.

Lemma D.1

For any $S \in \mathcal{P}_{\mathsf{fin}}(\mathbb{N})$,

$$\mathrm{Th}(\mathfrak{N}_{\mathsf{bad}}, S) = \mathrm{Th}(U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, (S, |S| \mod 2))$$

and, for $t = t_0$ or $t = t_1$, and $\gamma \in \{0, 1\}$,

$$\mathrm{Th}(\mathfrak{N}_{\mathsf{bad}}, t) = \mathrm{Th}(U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, (t, \gamma))$$

In more words: if $\varphi \in \mathcal{L}^{\mathsf{or}}_{\Box\bigcirc}$ is some formula, then $(\mathfrak{N}_{\mathsf{bad}}, S) \models \varphi$ iff $(U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, (S, |S| \mod 2)) \models \varphi$, where $|S| \mod 2$ is 0 if S has an even number of elements and 1 if S has an odd number of elements.

Proof. —

This is proved by induction on the structure of φ. The case where φ is atomic follows by definition of $U(\mathfrak{N}_{\mathsf{bad}})$ (in particular: $U(\mathfrak{N}_{\mathsf{bad}})$ and $\mathfrak{N}_{\mathsf{bad}}$ evaluate atomic propositions the same way) and definition of OR-augmentation.

The $\wedge$ and $\neg$ cases are trivial. Now consider $\Diamond$. Suppose $(\mathfrak{N}_{\mathsf{bad}}, y) \models \Diamond\psi$ (assume $y \in \mathcal{P}_{\mathsf{fin}}(\mathbb{N})$ for simplicity. The reasoning is basically the same if $y = t_0$ or t_1). Since there is only one open set, N_{bad}, this just means that there's some other world y' of N_{bad} such that $(\mathfrak{N}_{\mathsf{bad}}, y') \models \psi$. So then by inductive hypothesis we have that the corresponding world of $U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}$ (either $(S, |S| \mod 2)$ if $y' = S \in \mathcal{P}_{\mathsf{fin}}(\mathbb{N})$, or $(y', 0)$ or $(y', 1)$ if $y' = t_0$ or $y' = t_1$) validates ψ. So no open set can distinguish this world from the world corresponding to y, $(y, |y| \mod 2)$, so $(U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, (y, |y| \mod 2) \models \Diamond\psi$. We can use a similar argument to prove the converse, though slightly more work is required there: if $(S, |S| \mod 2)$ validates $\Diamond\psi$ as witnessed by $(S', 1 - (|S'| \mod 2))$, we cannot apply our IH and be done. Instead, we need to pick some natural number m such that $m \notin S'$ and such that p_m does not occur in ψ. From these assumptions we get

$$(U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, (S' \cup \{m\}, 1 - (|S'| \mod 2))) \models \psi$$

because this world agrees with $(S', 1 - (|S'| \mod 2))$ on all the primitive propositions occuring in ψ, and has the same primitive dynamics (the primitive dynamics are trivial) and same constructed dynamics as (both have the same constructor state). But now,

$$|S' \cup \{m\}| \mod 2 = (1 + |S'|) \mod 2 = 1 - (|S'| \mod 2)$$

so we can apply the IH to this world and get a world of $\mathfrak{N}_{\mathsf{bad}}$ validating ψ, hence our argument carries through.

The $\bigcirc_\pi$ case for $\pi \in \Pi$ is also pretty straightforward, since the primitive dynamics are the same in $\mathfrak{N}_{\mathsf{bad}}$, $U(\mathfrak{N}_{\mathsf{bad}})$, and $U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}$.

The interesting case is formulas of the form $\bigcirc_{\pi_i \mathsf{or} \pi_j}\psi$ and $S \in \mathcal{P}_{\mathsf{fin}}(\mathbb{N})$. Recall that, in $\mathfrak{N}_{\mathsf{bad}}$, $(\pi_i \mathsf{\ or\ } \pi_j)$ gets interpreted as π_i if S has an even number of elements and as π_j if an odd number. Assume first that $|S|$ is even.

$$\begin{aligned}
(\mathfrak{N}_{\mathsf{bad}}, S) \models \bigcirc_{\pi_i \mathsf{or} \pi_j}\psi &\iff (\mathfrak{N}_{\mathsf{bad}}, \|\pi_i \mathsf{\ or\ } \pi_j\|(S)) \models \psi \\
&\iff (\mathfrak{N}_{\mathsf{bad}}, \|\pi_i\|(S)) \models \psi \\
&\iff (\mathfrak{N}_{\mathsf{bad}}, t_i) \models \psi \\
&\iff (U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, (t_i, 0)) \models \psi \qquad \text{(Inductive Hypothesis)} \\
&\iff (U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, \|\pi_i\|(S, 0)) \models \psi \\
&\iff (U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, \|\pi_i \mathsf{\ or\ } \pi_j\|(S, 0)) \models \psi \qquad (*) \\
&\iff (U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, (S, 0)) \models \bigcirc_{\pi_i \mathsf{or} \pi_j}\psi
\end{aligned}$$

The step labeled (*) is key: $(\pi_i \mathsf{\ or\ } \pi_j)$ gets interpreted as π_i in $(S, 0)$ since the OR-state is 0. That is why the claim is about $(S, |S| \mod 2)$: so the resolution of or-programs (in $U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}$, according to the OR-augmentation semantics) matches the "parity checking" of $\mathfrak{N}_{\mathsf{bad}}$. We can make the analogous claim for $|S|$ odd: then $(\pi_i \mathsf{\ or\ } \pi_j)$ is interpreted as π_j. One thing to note: we've shown that certain worlds of $\mathfrak{N}_{\mathsf{bad}}$ and $U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}$ have the same theories. But this is not a bisimulation! Indeed, we show below that there cannot be such a bisimulation.

We simultaneously prove the portion of the claim dealing with t_0 and t_1. These two worlds (of $\mathfrak{N}_{\mathsf{bad}}$), satisfy exactly the same formulas as their respective counterparts in $U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}$: the only primitive proposition $(\mathfrak{N}_{\mathsf{bad}}, t_0)$ satisfies is p_0, and the same is true for $(U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, (t_0, 0))$, $(U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, (t_0, 1))$; $(\mathfrak{N}_{\mathsf{bad}}, t_1)$ does not validate any primitive propositions, and neither do $(U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, (t_1, 0))$ and $(U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, (t_1, 1))$. Again, the $\wedge$ and $\neg$ cases of the induction are routine. The $\Box$ induction step also follows by the fact that the topology is indiscrete.

□

So, from this, we get the following lemma.

Lemma D.2
If $U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}} \models \varphi$, then $\mathfrak{N}_{\mathsf{bad}} \models \varphi$.

This follows quite easily from the previous lemma: suppose $U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}} \models \varphi$. Then, for $S \in \mathcal{P}_{\mathsf{fin}}(\mathbb{N})$, $\mathrm{Th}(\mathfrak{N}_{\mathsf{bad}}, S) = \mathrm{Th}(U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}, (S, |S| \mod 2))$, so $(\mathfrak{N}_{\mathsf{bad}}, S) \models \varphi$. Likewise for t_0 and t_1. From this latter lemma, we obtain our claim (c) from the top of the proof: $\mathrm{Th}^{\mathsf{or}}_{\Box\bigcirc}(U(\mathfrak{N}_{\mathsf{bad}})^{\mathsf{OR}}) \subseteq \mathrm{Th}_{\Box\bigcirc}(\mathfrak{N}_{\mathsf{bad}})$.

The last thing to prove is that $\mathfrak{N}_{\mathsf{bad}}$ is not Π^{or}-bisimilar to any DTM of the form $\mathfrak{M}^{\mathsf{OR}}$. Assume for the sake of contradiction that $s : \mathfrak{N}_{\mathsf{bad}} \mathrel{+\!\!\!\rightarrow}_{\Pi^{\mathsf{or}}} \mathfrak{M}^{\mathsf{OR}}$ for some Π-DTM $\mathfrak{M}$.

Now, remember that N_{bad}, the underlying set of $\mathfrak{N}_{\mathsf{bad}}$, must be an open set in τ_{bad}. By the (Forth) condition of bisimulation, it must be the case that $s(N_{\mathsf{bad}})$ – the set of worlds related to any world of $\mathfrak{N}_{\mathsf{bad}}$ by s – is open in the topology on $\mathfrak{M}^{\mathsf{OR}}$. We claim that it is not.

Consider $\emptyset$ as a world of $\mathfrak{N}_{\mathsf{bad}}$. There must be some world (x, γ) of $\mathfrak{M}^{\mathsf{OR}}$ such that $\emptyset$ is related to (x, γ) by s, since s is total. Furthermore, we know that γ must be 0: since $\emptyset$ has

an even number of elements, $(\pi_i \mathsf{or} \pi_j)$ gets interpreted as π_i at $(\mathfrak{N}_{\mathsf{bad}}, \emptyset)$. So if γ were 1, then we'd have

$$(\mathfrak{M}^{\mathsf{OR}}, (x, \gamma)) \not\models \bigcirc_{\pi_0 \mathsf{or} \pi_1} p_0$$

because π_0 or π_1 would be interpreted as π_1, and π_1 leads to a world where p_0 is false. But we have

$$(\mathfrak{N}_{\mathsf{bad}}, \emptyset) \models \bigcirc_{\pi_0 \mathsf{or} \pi_1} p_0$$

so γ must be 0. However, $(x, 0)$ and $(x, 1)$ are topologically indistinguishable in $\mathfrak{M}^{\mathsf{OR}}$: any open set containing $(x, 0)$ must also contain $(x, 1)$. So we assumed that $s(N_{\mathsf{bad}})$ is open in order to have s constitute a bisimulation. Putting this together, we have:

$$(x, 1) \in s(N_{\mathsf{bad}})$$

i.e. there must be some world w of $\mathfrak{N}_{\mathsf{bad}}$ such that w is related to $(x, 1)$ by s. This, however, is impossible: if w is related to $(x, 1)$ by a Π^{or}-bisimulation, then w and $(x, 1)$ must have the same theory. But we know that $(x, 1)$ must make all atomic propositions false:

$$\begin{aligned}
(\mathfrak{M}^{\mathsf{OR}}, (x, 1)) \models p_i &\iff (\mathfrak{M}^{\mathsf{OR}}, (x, 0)) \models p_i && \text{(Corollary 3.1.2)} \\
&\iff (\mathfrak{N}_{\mathsf{bad}}, \emptyset) \models p_i && (s : (\mathfrak{N}_{\mathsf{bad}}, \emptyset) \twoheadrightarrow (\mathfrak{M}^{\mathsf{OR}}, (x, 0)), \text{Theorem 4.2}) \\
&\iff i \in \emptyset && \text{(Defn of } \mathfrak{N}_{\mathsf{bad}})
\end{aligned}$$

So w cannot be in $\mathcal{P}_{\mathsf{fin}}(\mathbb{N})$: the only element of $\mathcal{P}_{\mathsf{fin}}(\mathbb{N})$ which, as a world of $\mathfrak{N}_{\mathsf{bad}}$, makes every atomic proposition false is $\emptyset$. $\emptyset$ cannot be Π^{or}-bisimilar to $(x, 1)$ because they resolve or-programs differently: $\emptyset$ validates $\bigcirc_{\pi_0 \mathsf{or} \pi_1} p_0$ whereas $(x, 1)$ does not. w cannot be t_0 because t_0 validates p_0. Finally, w cannot be t_1 because $\|\pi_0\|$ must be defined at w ($(x, 1)$ validates $\bigcirc_{\pi_0} \top$, and therefore so too must w), but $\|\pi_1\|$ isn't defined at t_1.

Having exhausted the possibilities for what w could be, we conclude there is no such w. Therefore $(x, 1)$ is not in the image of s, thus s cannot satisfy (Forth). So no Π^{or}-bisimulation $s : \mathfrak{N}_{\mathsf{bad}} \twoheadrightarrow \mathfrak{M}^{\mathsf{OR}}$ can exist.

With that, we've established (a), (b), and (c).

□

E Sect. 5 Supplement: Induced Refinement Relations

Example E.1

Let $s : \mathcal{F} \simeq_\Sigma \mathcal{G}$ be a surjective Σ-bisimulation between $\mathcal{F} = (X, \tau_X, \{f_\sigma\})$ and $\mathcal{G} = (Y, \tau_Y, \{g_\sigma\})$. The following are refinement relations on $\mathcal{F}$ and $\mathcal{G}$, respectively.

- The "**before** s" relation, denoted $\mathcal{R}s$, is defined to be the least equivalence relation on X such that, for all $x, x' \in X$,

$$(\exists y \in Y)(xsy \text{ and } x'sy) \quad \Longrightarrow \quad x(\mathcal{R}s)x'$$

- The "**after** s" relation, denoted $s\mathcal{R}$, is defined to be the least equivalence relation on Y such that, for all $y, y' \in Y$,

$$(\exists x \in X)(xsy \text{ and } xsy') \quad \Longrightarrow \quad y(s\mathcal{R})y'$$

Example E.2

In Example 4.1 we defined the Π-bisimulation $T^C_{\mathfrak{M}} : \mathfrak{M} \twoheadrightarrow_\Pi \mathfrak{M}^C$. Considering it as a bisimulation of the underlying frames (à la Note 4.2), we obtain a frame bisimulation

$$T^C_{\mathcal{F}} : \mathcal{F} \twoheadrightarrow_\Pi \mathcal{F}^C$$

where each $x \in |\mathcal{F}|$ is related to all and only the worlds of $\left|\mathcal{F}^C\right| = |\mathcal{F}| \times \Gamma$ whose first coordinate *is* x.

Consider the before and after relations of $T^C_{\mathcal{F}}$. The before relation is just equality: the only scenario where $xT^C_{\mathcal{F}}w$ and $x'T^C_{\mathcal{F}}w$ for some $x, x' \in |\mathcal{F}|$ and $w \in \left|\mathcal{F}^C\right|$ is if $x = x'$. Thus,

$$\mathcal{R}T^C_{\mathcal{F}}(x) = \{x\}$$

The after relation will be more important for our considerations. Observe that (x, γ) and (x', γ') are $T^C_{\mathcal{F}}\mathcal{R}$-related just in case $x = x'$, i.e. they are two worlds with the same first coordinate but possibly different Γ-states. Thus, for any given world (x, γ) of $\mathcal{F}$, the set $[(x, \gamma)]_{T^C_{\mathcal{F}}\mathcal{R}}$ is given by

$$\left[(x, \gamma)\right]_{T^C_{\mathcal{F}}\mathcal{R}} = \{x\} \times \Gamma.$$

However, notice that this is exactly the relation $\mathcal{R}^C_{\mathcal{F}}$ introduced above – a fact which saves us from the slightly-more-clumsy notation "$T^C_{\mathcal{F}}\mathcal{R}$".

Theorem E.1 (Refined Primitive Invariance)

For all surjective Π-bisimulations $s : \mathcal{F} \twoheadrightarrow_\Pi \mathcal{G}$,

$$\mathrm{Th}_{\Box\bigcirc}(\Pi; \mathcal{F}, \mathcal{R}s) = \mathrm{Th}_{\Box\bigcirc}(\Pi; \mathcal{G}, s\mathcal{R})$$

Proof. —

Observe that we can "transfer" valuations back and forth across s: if V is a valuation on $\mathcal{F}$, we can define a valuation W on $\mathcal{G}$ by:

$$y \in W(p) \quad \Longrightarrow \quad s^{-1}(y) \subseteq V(p).$$

Likewise, for W a valuation on $\mathcal{G}$, we can define V by putting $x \in V(p)$ iff $s(x) \subseteq W(p)$. Notice that if we obtain W from V in this manner (or V from W), then s satisfies (Base), i.e. is a Π-bisimulation between the DTMs $(\mathcal{F}, V)$ and $(\mathcal{G}, W)$, which gives

$$\mathrm{Th}_{\Box\bigcirc}(\Pi; (\mathcal{F}, V)) = \mathrm{Th}_{\Box\bigcirc}(\Pi; (\mathfrak{G}, W)).$$

So, to prove the theorem, it suffices to show that all $\int R$-respecting valuations W on $\mathfrak{G}$ can be obtained in this manner from a $\mathcal{R}s$-respecting valuation on $\mathcal{F}$, and vice versa. In other words, we want to show that this "transferring" operation between $\mathcal{R}s$-respecting valuations and $s\mathcal{R}$-respecting valuations is bijective.

To see one direction, pick arbitrary $s\mathcal{R}$-respecting valuation W on $\mathcal{G}$. Then pull that back to a valuation V on $\mathcal{F}$ by $x \in V(p)$ iff $s(x) \subseteq W(p)$. It's easy to check that V respects $\mathcal{R}s$. But then, if we transfer V across s (call this W'), we get W back:

$$\begin{aligned} y \in W'(p) \quad &\Longleftrightarrow \quad s^{-1}(y) \subseteq V(p) \\ &\Longleftrightarrow \quad s^{-1}(y) \subseteq \{x \in |\mathcal{F}| \ : \ s(x) \subseteq W(p)\} \\ &\Longleftrightarrow \quad s(s^{-1}(y)) \subseteq W(p) \end{aligned}$$

But $s(s^{-1}(y))$ consists of those y' such that xsy' for some $x \in s^{-1}(y)$. But by the assumption that W respects $s\mathcal{R}$, y and y' must get the same valuation. So $y \in W'(p)$ iff $y \in W(p)$. A similar argument will show that any $\mathcal{R}s$-respecting valuation on $\mathcal{F}$ arises from a $s\mathcal{R}$-respecting valuation on $\mathcal{G}$, and we're done.

□

Corollary E.1.1
For all Π-frames $\mathcal{F}$ and all program constructors C,

$$\mathrm{Th}_{\Box\bigcirc}(\Pi; \mathcal{F}) = \mathrm{Th}_{\Box\bigcirc}(\Pi; \mathcal{F}^C, \mathcal{R}^C_{\mathcal{F}}).$$

Proof. —
This follows from Theorem E.1 and the observations from Example E.2.

□

F Sect. 5 Proofs

(Theorem 5.1)
Suppose $(\mathcal{G}, \mathcal{R})$ is a refined Σ-frame which has outcome-dense refinement classes and let V be any valuation on $\mathcal{G}$ which respects $\mathcal{R}$. Then, for any $x, x' \in |\mathcal{G}|$ such that $x\mathcal{R}x'$, the following equalities hold.

$$\mathrm{Th}_{\Box\bigcirc}(\Pi; (\mathcal{G}, V), x) = \mathrm{Th}_{\Box\bigcirc}(\Pi; (\mathcal{G}, V), x')$$
$$\mathrm{Th}_{PDL}(\Sigma; (\mathcal{G}, V), x) = \mathrm{Th}_{PDL}(\Sigma; (\mathcal{G}, V), x')$$

Proof. —
Let $\mathfrak{N} = (\mathcal{G}, V)$.

For the first statement (about the Π-theories), observe that the quotient function $Q_{\mathcal{R}} : \mathfrak{N} \to \mathfrak{N}/\mathcal{R}$ satisfies the (Base) condition by definition, and is hence a Π-bisimulation between these DTMs. Therefore,

$$\begin{aligned}
\mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{N}, x) &= \mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{N}/\mathcal{R}, [x]) && \text{(Theorem 4.2)}\\
&= \mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{N}/\mathcal{R}, [x']) && (x\mathcal{R}x')\\
&= \mathrm{Th}_{\Box\bigcirc}(\Pi; \mathfrak{N}, x'). && \text{(Theorem 4.2)}
\end{aligned}$$

For the second statement, induct on $\varphi \in \mathcal{L}_{PDL}(\Sigma)$. The base case (primitive propositions) follows from the assumption that V respects $\mathcal{R}$. The $\neg$ and $\wedge$ cases are immediate. To prove the $\langle\sigma\rangle$ case, inductively assume that all $\mathcal{R}$-related worlds agree on ψ. Now suppose

$$(\mathfrak{N}, x) \models \langle\sigma\rangle\psi,$$

i.e. $(\mathfrak{N}, x) \models \Diamond \bigcirc_\sigma \psi$. To show $(\mathfrak{N}, x') \models \Diamond \bigcirc_\sigma \psi$, we'll pick an arbitrary open set U containing x' and show that it contains a $\bigcirc_\sigma\psi$-world. Consider $\mathcal{R}(U)$, which must be open by definition of refinement relation. It must also contain x, since $x' \in U$ and $x\mathcal{R}x'$. Therefore, $\mathcal{R}(U)$ must contain a $\bigcirc_\sigma\psi$ world, by the assumption that $(\mathfrak{N}, x') \models \Diamond \bigcirc_\sigma \psi$. Call this world w. We know that $\|\sigma\|(w)$ is defined and is a ψ-world. Now, notice that U and $[w]_{\mathcal{R}}$ must intersect: otherwise, it wouldn't be the case that $w \in \mathcal{R}(U)$. By the assumption of outcome-dense refinement classes, there must be some $w' \in [w]_{\mathcal{R}} \cap U$ such that $\|\sigma\|(w')$ is defined and is $\mathcal{R}$-related to $\|\sigma\|(w)$. By the inductive hypothesis, this gives that $\|\sigma\|(w')$ is a ψ-world, hence w is a $\bigcirc_\sigma\psi$-world. We have located a $\bigcirc_\sigma\psi$-world in an arbitrary open set U containing x', and thus

$$(\mathfrak{N}, x') \models \langle\sigma\rangle\psi.$$

The opposite direction (that $(\mathfrak{N}, x') \models \langle\sigma\rangle\psi$ implies $(\mathfrak{N}, x') \models \langle\sigma\rangle\psi$) is identical.

□

G Sect. 6 Proofs

(Corollary 6.0.1)
If C is an indiscrete program constructor (in the sense of Example 5.2), then for all Π-DTMs $\mathfrak{M} = (\mathcal{F}, V)$, and all worlds w, w' of $\mathfrak{M}^C$ such that $w\mathcal{R}_{\mathcal{F}}^C w'$,

$$\mathrm{Th}_{PDL}(\Pi^c; \mathfrak{M}^C, w) = \mathrm{Th}_{PDL}(\Pi^c; \mathfrak{M}^C, w').$$

Proof. —
This simply instantiates Theorem 5.1, and makes the observation that the valuation on an augmented DTM $\mathfrak{M}^C$ respects $\mathcal{R}_{\mathcal{F}}^C$ (where $\mathcal{F}$ is the frame underlying $\mathfrak{M}$).

□

(Prop. 6.1)
For any program constructor C and any Π-frame $\mathcal{F}$,

$$\mathcal{F} \simeq_\Pi \mathcal{F}^C/\mathcal{R}_{\mathcal{F}}^C.$$

Indeed,

$$\mathcal{F} \cong_\Pi \mathcal{F}^C/\mathcal{R}_{\mathcal{F}}^C.$$

Proof. —
Observe that the $\mathcal{R}_{\mathcal{F}}^C$-equivalence classes are just the sets of all worlds with the same $\mathcal{F}$-world. So quotienting by this equivalence relation just gives $\mathcal{F}$ back.

□

(Prop. 6.2)
For any C, the following are equivalent.

- For every Π-frame $\mathcal{F}$, $(\mathcal{F}^C, \mathcal{R}_{\mathcal{F}}^C) \models \Delta$
- For every Π-DTM $\mathfrak{M}$, $\mathfrak{M}^C \models \Delta$

Proof. —
Observe for any Π-DTM $\mathfrak{M} = (\mathcal{F}, V)$ that if we denote write $\mathfrak{M}^C = (\mathcal{F}^C, V^C)$, then the valuation V^C respects $\mathcal{R}_{\mathcal{F}}^C$. Moreover, as noted at several points throught this thesis, all $\mathcal{R}_{\mathcal{F}}^C$-respecting valuations arise in this way. This gives us the desired equivalence: if all $\mathfrak{M}^C$ validate Δ, then any C-augmented frame $\mathcal{F}^C$ equipped with a $\mathcal{R}_{\mathcal{F}}^C$-respective valuation validates Δ, hence $(\mathcal{F}^C, \mathcal{R}_{\mathcal{F}}^C)$ validates Δ. Conversely, if $(\mathcal{F}^C, \mathcal{R}_{\mathcal{F}}^C)$ validates Δ, then any $\mathfrak{M}^C = (\mathcal{F}^C, V^C)$ validates Δ, because V^C respects $\mathcal{R}_{\mathcal{F}}^C$.

□

H Sect. 7 Proofs

(Lemma 7.2)
For every Π-frame $\mathcal{F}$,

$$(\mathcal{F}^{\mathsf{OR}}, \mathcal{R}_{\mathcal{F}}^{\mathsf{OR}}) \models \chi_{\mathsf{OR}}$$

Proof. —
Recall these definitions

Definition (Defn. 7.4)

Now, given a world x of $\mathcal{G}$ and an ordered pair (π_0, π_1) of primitive programs,

- x is a *0-world for* (π_0, π_1)

$$\|\pi_0\|(x) \approx_{\mathcal{R}} \|\pi_0 \text{ or } \pi_1\|(x)$$

- x is a *1-world for* (π_0, π_1)

$$\|\pi_1\|(x) \approx_{\mathcal{R}} \|\pi_0 \text{ or } \pi_1\|(x)$$

We'll use Prop. 6.2 and prove that every OR-augmented DTM validates χ_{OR}. Throughout, let $\mathfrak{M}$ be an arbitrary Π-DTM.

Let's begin by observing that for every $p \in \Phi$, every $\pi_0, \pi_1 \in \Pi$, and every $x \in |\mathfrak{M}|$,

$$(\mathfrak{M}^C, (x, 0)) \models \mathsf{Maybe}_0(p, \pi_0, \pi_1)$$
$$(\mathfrak{M}^C, (x, 1)) \models \mathsf{Maybe}_1(p, \pi_0, \pi_1)$$

Moreover, if $\|\pi_0\|_{\mathfrak{M}}(x)$ and $\|\pi_1\|_{\mathfrak{M}}(x)$ differ on p (e.g. one is defined and validates p whereas the other is undefined or refutes p) then

$$(\mathfrak{M}^C, (x, 0)) \models \mathsf{Only}_0(p, \pi_0, \pi_1)$$
$$(\mathfrak{M}^C, (x, 1)) \models \mathsf{Only}_1(p, \pi_0, \pi_1).$$

From these observations, the validity of (OR-Typ), (OR-Reg0), and (OR-Reg1) is immediate. The validity of (OR-Real0) and (OR-Real1) comes from the fact that every $\mathcal{R}^{\mathsf{OR}}_{\mathcal{F}}$-equivalence class (and any nonempty intersection of an open set with an $\mathcal{R}^{\mathsf{OR}}_{\mathcal{F}}$-equivalence class) contains both the 0-copy $(x, 0)$ and the 1-copy $(x, 1)$ of some world x of $\mathfrak{M}$. Either neither of these validate their respective Only formula (e.g. if $\|\pi_0\|(x)$ and $\|\pi_1\|(x)$ are $\mathcal{R}$-related) – in which case the antecedent of (OR-Real0) and (OR-Real1) is refuted, and Realization holds vacuously. If $(x, 0)$ validates $\mathsf{Only}_0(q, \pi_0, \pi_1)$, then the outcomes of π_0 and π_1 are dinstinguishable by q, and then $(x, 1)$ must validate $\mathsf{Only}_1(q, \pi_0, \pi_1)$, yielding (OR-Real1). Likewise for (OR-Real0). The Persistence axioms are validated because no Π^{or}-program σ actually changes the OR constructor state, i.e. execution of σ from $(x, 0)$ lands in some $(x', 0)$, and likewise for 1 – regardless of whether σ is primitive or constructed. Thus, if executing σ indeed "succeeds" from $(x, 0)$ (in the sense that $\bigcirc_\sigma \top$ holds),[36] then the world it results in, $(x', 0)$, also validates $\mathsf{Maybe}_0(q, \pi_2, \pi_3)$ for any q, π_2, π_3 because it is still a 0-copy of some world, x', of $\mathfrak{M}$ and therefore validates every Maybe_0 formula.

□

(Lemma 7.3)

For any world x and any primitive programs π_0, π_1, if

$$((\mathcal{G}, v), x) \models (\bigcirc_{\pi_0} p \leftrightarrow \bigcirc_{\pi_0 \mathsf{or} \pi_1} p) \vee (\bigcirc_{\pi_1} p \leftrightarrow \bigcirc_{\pi_0 \mathsf{or} \pi_1} p)$$

for all v which respect $\mathcal{R}$, then x is either a 0-world or a 1-world (or both) for (π_0, π_1)

[36] Under our earlier epistemic interpretation, this was the action denoted by σ being "possible" at the present point of the situation.

Proof. —

Pick some arbitrary world x of $\mathcal{G}$ and some arbitrary ordered pair (π_0, π_1) of programs in Π. So suppose (for contradiction) that x is neither a 0-world for (π_0, π_1) nor a 1-world. Then we're in one of three cases.

(1a) $\|\pi_0 \text{ or } \pi_1\|$ is defined at x, but $\|\pi_0\|$ and $\|\pi_1\|$ are not;

(1b) $\|\pi_0\|$ and $\|\pi_1\|$ are defined at x, but $\|\pi_0 \text{ or } \pi_1\|$ isn't;

(1c) $\|\pi_0 \text{ or } \pi_1\|$ and one or both of $\|\pi_0\|, \|\pi_1\|$ is defined at x, but their images are not $\mathcal{R}$-related:

$$(\|\pi_0 \text{ or } \pi_1\|(x), \|\pi_0\|(x)) \notin \mathcal{R} \qquad \text{and} \qquad (\|\pi_0 \text{ or } \pi_1\|(x), \|\pi_1\|(x)) \notin \mathcal{R}.$$

We'll deal with the subcase of (1c) where both $\|\pi_0\|$ and $\|\pi_1\|$ are defined, but the rest can be handled analogously.

Suppose $\|\pi_0\|$, $\|\pi_1\|$, and $\|\pi_0 \text{ or } \pi_1\|$ are defined at x but

$$(\|\pi_0 \text{ or } \pi_1\|(x), \|\pi_0\|(x)) \notin \mathcal{R} \qquad \text{and} \qquad (\|\pi_0 \text{ or } \pi_1\|(x), \|\pi_1\|(x)) \notin \mathcal{R}.$$

Then define a valuation v on $\mathcal{G}$ by putting

$$v(p) = \left[\|\pi_0\|(x)\right] \cup \left[\|\pi_1\|(x)\right] = \left\{x' \in X \ : \ x' \text{ is } \mathcal{R}\text{-related to } \|\pi_0\|(x) \text{ or } \|\pi_1\|(x)\right\},$$

Note this does *not* include $\|\pi_0 \text{ or } \pi_1\|(x)$. So x validates $\bigcirc_{\pi_0} p$ and $\bigcirc_{\pi_1} p$ but not $\bigcirc_{\pi_0 \text{or} \pi_1} p$, and we have

$$((\mathcal{G}, v), x) \not\models (\bigcirc_{\pi_0} p \leftrightarrow \bigcirc_{\pi_0 \text{or} \pi_1} p) \vee (\bigcirc_{\pi_1} p \leftrightarrow \bigcirc_{\pi_0 \text{or} \pi_1} p).$$

Since v respects $\mathcal{R}$, this proves

$$(\mathcal{G}, \mathcal{R}) \not\models (\bigcirc_{\pi_0 \text{or} \pi_1} p \leftrightarrow \bigcirc_{\pi_0} p) \vee (\bigcirc_{\pi_0 \text{or} \pi_1} p \leftrightarrow \bigcirc_{\pi_1} p)$$

contrary to our assumption that $(\mathcal{G}, \mathcal{R}) \models \chi_{\mathsf{OR}}$. So we have proved that x must be a 0-world or a 1-world for (π_0, π_1). As mentioned, similar constructions will work for the other cases falling under (1c), as well as (1a) and (1b)[37] .

□

(Lemma 7.4)
s is surjective and open

Proof. —

The main thrust of this proof is showing that in a refined Π^{or}-frame $(\mathcal{G}, \mathcal{R})$ validating χ_{OR}, every $\mathcal{R}$-equivalence class contains a 0-world and every $\mathcal{R}$-equivalence class contains a 1-world. We prove the 0 case; 1 is identical.

Suppose $[x]$ contained no 0-worlds. Then, in particular, x is not a 0-world and, by definition of 0-world, there must be some pair of primitive programs (π_0, π_1) for which x is a 1-world but *not* a 0-world (if there were no such pair, then x would be a 0-world). We'll construct an $\mathcal{R}$-respecting valuation V on $\mathcal{G}$ such that $(\mathcal{G}, V)$ refutes (OR-Real0).

Assume x fails to be a 0-world for (π_0, π_1) because $\|\pi_0\|(x)$ and $\|\pi_0 \text{ or } \pi_1\|(x)$ are both defined, but are not $\mathcal{R}$-related. Recall by **??** and Defn. 7.3 that there are several other cases

[37] (1b) uses the exact same definition of $v(p)$. For (1a) and the other (1c), put $v(p)$ to be the $\mathcal{R}$-equivalence class of $\|\pi_0 \text{ or } \pi_1\|(x)$.

– they can be handled analogously. x must then be a 1-world as arugued in the proof of Theorem 7.1. Put $V(p) = [x]$ and put $V(q)$ to be the $\mathcal{R}$-equivalence class of $\|\pi_0\|(x)$. Observe then that V respects $\mathcal{R}$, and that $((\mathcal{G}, V), x) \models p$ and

$$(\mathcal{G}, V), x) \models \bigcirc_{\pi_0} q \wedge \neg \bigcirc_{\pi_1} (x) \wedge \neg \bigcirc_{\pi_0 \mathsf{or} \pi_1} q$$

To see this, note that $\|\pi_0 \mathsf{\ or\ } \pi_1\|(x)$ cannot be in $V(q) = [\|\pi_0\|(x)]$, because x is not a 0-world for (π_0, π_1) and so $\|\pi_0\|(x)$ and $\|\pi_0 \mathsf{\ or\ } \pi_1\|(x)$ cannot be $\mathcal{R}$-related. We *are* assuming that x is a 1-world for (π_0, π_1), however, so $\|\pi_0 \mathsf{\ or\ } \pi_1\|(x)$ and $\|\pi_1\|(x)$ must be $\mathcal{R}$-related. So $\|\pi_1\|(x)$ can't be a q-world either, hence x validates $\neg \bigcirc_{\pi_1} q$. Putting this all together, we have

$$((\mathcal{G}, V), x) \models p \wedge \mathsf{Only}_1(q, \pi_0, \pi_1).$$

So, to refute (OR-Real0) at x, we need to refute $\Diamond(p \wedge \mathsf{Only}_0(q, \pi_0, \pi_1)$ at x. To do this, we just need an open set U witnessing that

$$((\mathcal{G}, V), x) \models \Box\neg(p \wedge \mathsf{Only}_0(q, \pi_0, \pi_1).$$

But we have something stronger: *no world of* $\mathcal{G}$ *validates* $p \wedge \mathsf{Only}_0(q, \pi_0, \pi_1)$ – and therefore $U = |\mathcal{G}|$ suffices. To see this, observe that p is only validated by the worlds of $[x]$. And no world of $[x]$ will validate $\mathsf{Only}_0(q, \pi_0, \pi_1)$. If some world x' of $[x]$ did validate $\mathsf{Only}_0(q, \pi_0, \pi_1)$, then x' could not possibly be a 1-world for (π_0, π_1). But, as proven above, x' must either be a 0-world for every pair of programs or a 1-world for every pair of programs. Since it *isn't* a 1-world for this pair of programs, it must be a 0-world for all pairs of programs, contrary to our assumption. We have reached a contradiction, so we conclude that $[x]$ must contain at least one 0-world. Again, by a similar proof, $[x]$ must contain a 1-world too.

But then surjectivity follows quickly: every world of $(\mathcal{G}/\mathcal{R})^{\mathsf{OR}}$ is of the form $([x], 0)$ or $([x], 1)$ for some x. To get a world x_0 which is related to $([x], 0)$ by s, just pick the 0-world in $x_0 \in [x]$ which must exist by the preceding argument. It's a 0-world in $[x]$, and hence must be related to $([x], 0)$ by s. Likewise for 1. Openness also follows quickly: if we take an open set U in $\mathcal{G}$, then the set of worlds related to it is just

$$\{([x], \gamma) \ : \ x \in U, \ \gamma \in \{0, 1\}\} = \{[x] \ : \ x \in U\} \times \{0, 1\}$$

where both $([x], 0)$ and $([x], 1)$ are included by the preceding argument. But notice that $\{[x] \ : \ x \in U\}$ is the image of U under the quotient map $\mathcal{G} \to \mathcal{G}/\mathcal{R}$, which is open by the fact that $\mathcal{R}$ is a refinement relation. This completes the proof of the lemma.

□

(Lemma 7.5)
For every $\sigma \in \Pi^{\mathsf{or}}$, every world x of $\mathcal{G}$ and every world w of $\mathcal{J}$,

- If xsw and $\|\sigma\|_{\mathcal{G}}(x)$ is defined, then $\|\sigma\|_{\mathcal{J}}(w)$ is defined and $\|\sigma\|_{\mathcal{G}}(x)$ is related to $\|\sigma\|_{\mathcal{J}}(w)$ by s
- If xsw and $\|\sigma\|_{\mathcal{J}}(w)$ is defined, then $\|\sigma\|_{\mathcal{G}}(x)$ is defined and $\|\sigma\|_{\mathcal{G}}(x)$ is related to $\|\sigma\|_{\mathcal{J}}(w)$ by s

Proof. —
First: if σ is a primitive program ($\sigma \in \Pi$), then this is follows from the definitions of quotients, OR-augmentation, and s: if xsw, then $w = ([x], \gamma)$ for $\gamma \in \mathsf{Type}(x)$. We've seen that the

behavior of a primitive program π at x in $\mathcal{G}$ is exactly the behavior of π at $[x]$ in $\mathcal{G}/\mathcal{R}$, which is exactly the behavior of π at $([x], \gamma)$ (for either $\gamma = 0$ or 1) in $\mathcal{J} = (\mathcal{G}/\mathcal{R})^{\mathsf{OR}}$.

For constructed programs: notice $w = ([x], \gamma)$ for $\gamma \in \mathsf{Type}(x)$. If $\gamma = 0$, then that means $0 \in \mathsf{Type}(x)$ and

$$\|\pi_0\|_{\mathcal{G}}(x) \approx_{\mathcal{R}} \|\pi_0 \text{ or } \pi_1\|_{\mathcal{G}}(x).$$

So if $xs([x], 0)$ and $\|\pi_0 \text{ or } \pi_1\|_{\mathcal{G}}(x)$ is defined, then $\|\pi_0\|_{\mathcal{G}}(x)$ is also defined and is $\mathcal{R}$-related to $\|\pi_0 \text{ or } \pi_1\|_{\mathcal{G}}(x)$. Thus,

$$\begin{aligned}
\|\pi_0 \text{ or } \pi_1\|_{\mathcal{J}}([x], 0) &= \|\pi_0\|_{\mathcal{J}}([x], 0) && \text{(Defn of OR)} \\
&= (\|\pi_0\|_{\mathcal{G}/\mathcal{R}}([x]), 0) && \text{(Defn. of OR-augment)} \\
&= ([\|\pi_0\|_{\mathcal{G}}(x)], 0) && \text{(Defn. of quotient)} \\
&= ([\|\pi_0 \text{ or } \pi_1\|_{\mathcal{G}}(x)], 0). && \text{(Above)}
\end{aligned}$$

Now, in any refined frame validating χ_{OR}, it must be the case that if $0 \in \mathsf{Type}(x)$, then $0 \in \mathsf{Type}(\|\sigma\|_{\mathcal{G}}(x))$. If not, we could be able to obtain a refutation of the Persistence axioms of χ_{OR} by carefully choosing an $\mathcal{R}$-respecting valuation of p and q. Therefore, $0 \in \mathsf{Type}(\|\pi_0 \text{ or } \pi_1\|_{mc\mathcal{G}}(x))$, and we have

$$(\|\pi_0 \text{ or } \pi_1\|_{\mathcal{G}}(x)) \; s \; (\|\pi_0 \text{ or } \pi_1\|_{\mathcal{J}}([x], 0))$$

as desired.

The other direction – noting that the behavior of programs in $\mathcal{J}$ is reflected in their behavior in $\mathcal{G}$ – is proved similarly.

□