10 Two-Dimensional Modal Logic

1. Introduction 2
2. Scoped-Actuality 2
3. System 2D(1) 2
4. Examples of Derivations in 2D(1) 3
5. The Problem of Necessitation 5
6. System 2D(2) 6
7. System 2D(3) 6
8. The Semantics of 2D(3) 8
9. How Explicit-Scope Actuality Escapes the Difficulties that Face Rigid Actuality 9
10. How Explicit-Scope Actuality Escapes the Difficulties that Face Implicit-Scope Actuality 10
11. Multiple Applications of ‘$’ and ‘○’ 11

1. Introduction

In the present chapter, we examine *two-dimensional* logic, according to which sentences are evaluated with respect to *pairs* of worlds, which contrasts with *one-dimensional* logic, according to which sentences are evaluated with respect to *single* worlds, both of which contrast with *zero-dimensional* logic, according to which sentences are evaluated *simpliciter*.

2. Scoped-Actuality

The chief motivation for 2D logic is the logic of actuality. Consider the following example, which involves multiple modalities.

Russell **believes** my yacht **could** be bigger than it (actually) is.

According to the rigid account of actuality, the term 'actually' *invariably* refers to the *original* world of utterance, no matter how removed from the original world one gets in the semantic evaluation. On the other hand, according to the flexible account of actuality proposed in the previous chapter, 'actually' refers to the world under consideration just prior to its governing modal operator, which in the above example is 'could'. According to a third, narrow-scope, reading, 'actually' is semantically redundant.

Now, if we take advantage of scoped-terms, we can clearly demarcate these three readings, as follows.

(r) $(s/x)\mathbb{B}\Diamond[s > x]$

(f) $\mathbb{B}(s/x)\Diamond[s > x]$

(n) $\mathbb{B}\Diamond(s/x)[s > x]$

Here, 's' means 'the size of my yacht', '>' means 'is greater than', and '$\mathbb{B}$' means 'Russell believes that'.

Notice that these formulas correspond respectively to a wide, a narrow, and an intermediate reading of 'actually', as reflected in the location of the scoped-term '(s/x)'. Notice also that these formulas are logically distinct.

If we wish to convert the above formulas into corresponding formulas involving 'the actual', it would be nice to have a syntactic device to mark its scope. For this purpose we introduce a scope-operator '$' – which we can employ to translate the above formulas respectively as follows.

(r) $\$\mathbb{B}\Diamond[s > \bigcirc s]$

(f) $\mathbb{B}\$\Diamond[s > \bigcirc s]$

(n) $\mathbb{B}\Diamond\$[s > \bigcirc s]$

Notice that '$' replaces '$(s/x)$', and '$\bigcirc s$' replaces '$x$'.

As suggestive as this notation is, it is not entirely helpful unless we provide a corresponding formal syntax and semantics for the new operator '$'. The syntax is no problem. We simply add to our list of logical symbols a new one – '$' – which is deemed to be a one-place sentential operator, as formally characterized in the following rule of formation.

if Φ is a formula, then so is $\$\Phi$.

3. System 2D(1)

The semantics is more involved. Indeed, we consider three different accounts, each one an improvement over the last. In each case, the semantics is **two-dimensional** in the sense that formulas are evaluated at **pairs** of worlds. Whereas the first component is the "active" world, which is used to evaluate ordinary formulas; the second component is the "actual" world, which is used to evaluate formulas involving '$\bigcirc$'. This is made clearer in the semantic definitions, which we present by way of rules of derivation.

The rules for □ and ◇ are nearly identical to the rules for ordinary modal logic. The difference is that a secondary index is appended to the lines.

□O □Φ /i /j ——— Φ /i+m (old) /j	□D ~~SHOW~~: □Φ /i /j ~~SHOW~~: Φ /i+n (new) /j │ │
◇O ◇Φ /i /j ——— Φ /i+n (new) /j	◇I Φ /i+m (old) /j ——— ◇Φ /i /j

Notice that the secondary index plays absolutely no role in evaluating □ and ◇ formulas.

Non-modal rules are the same as before, except that all lines are indexed by *pairs* of indices. As usual, we drop the indices when they are the same for input and output. We could adopt a similar notational tactic for the modal rules. In particular, we could adopt the convention that, if a rule does not explicitly mention the second index, then it is understood that all lines have the same secondary index. In that case, we don't have to rewrite the modal rules at all; they *look* exactly the same as for ordinary modal logic.

So far the secondary index plays no role whatsoever. It comes into play in connection with '$' and '○', as formulated in the following rules.

$O/I $Φ /$i$ /j ═══ Φ /i /i	○O/I ○Φ /i /j ═══ Φ /j /j

Notice that '$' and '○' are symmetrical to each other. Using the terminology of (machine-language) programming, whereas the $-operator pushes the active world into storage (the "stack"), the ○-operator pulls the stored world out of storage and makes it active.

4. Examples of Derivations in 2D(1)

First, we note that, in doing derivations in 2D logic, we index every line by a pair of indices. When we start a derivation, we index the premises and conclusion by the same index pair – 0/0. In other words, when we start a derivation, we take the original world of utterance/evaluation to be both the *active world* and the *actual world*.

This has some immediate consequences. The following examples suggest that both '$' and '○' are completely trivial.

Example 1

(1)	~~SHOW~~: $P ↔ P	/0	/0	↔D
(2)	| [→] $P	/0	/0	As
(3)	| ~~SHOW~~: P	/0	/0	2,$O
(4)	| [←]P	/0	/0	As
(5)	| ~~SHOW~~: $P	/0	/0	4,$I

Example 2

(1)	~~SHOW~~: ○P ↔ P	/0	/0	↔D
(2)	| [→] ○P	/0	/0	As
(3)	| ~~SHOW~~: P	/0	/0	2,○O
(4)	| [←] P	/0	/0	As
(5)	| □: ○P	/0	/0	4,○I

Example 3

(1)	~~SHOW~~: $P ↔ ○P	/0	/0	↔D
(2)	| $P	/0	/0	As
(3)	| ~~SHOW~~: ○P	/0	/0	4,○I
(4)	| | P	/0	/0	2,$O
(5)	| ○P	/0	/0	As
(6)	| ~~SHOW~~: $P	/0	/0	7,$I
(7)	| | P	/0	/0	5,○O

Example 4

(1)	~~SHOW~~: $○P ↔ P	/0	/0	↔D
(2)	| $○P	/0	/0	As
(3)	| ~~SHOW~~: P	/0	/0	4,○O
(4)	| | ○P	/0	/0	2,$O
(5)	| P	/0	/0	As
(6)	| ~~SHOW~~: $○P	/0	/0	7,$I
(7)	| | ○P	/0	/0	5,○I

Example 5

(1)	~~SHOW~~: ○$P ↔ P	/0	/0	↔D
(2)	| ○$P	/0	/0	As
(3)	| ~~SHOW~~: P	/0	/0	4,$O
(4)	| | $P	/0	/0	2,○O
(5)	| P	/0	/0	As
(6)	| ~~SHOW~~: ○$P	/0	/0	7,○I
(7)	| | $P	/0	/0	5,$I

The intended use of '$' and '○' is in combination with modal operators. Here we have our first anti-trivial result.

Example 6-k

(1)	SHOW: ○□$P → $□○P	/0	/0	CD
(2)	○□$P	/0	/0	As
(3)	SHOW: $□○P	/0	/0	4,$I
(4)	SHOW: □○P	/0	/0	ND
(5)	SHOW: ○P	/01	/0	5,○I
(6)	SHOW: P	/0	/0	???
(7)	□$P	/0	/0	2,○O
(8)	$P	/01	/0	5,□O
(9)	P	/01	/01	6,$O

Note, however, that if we enlarge our modal system to System T, we achieve triviality.

Example 6-t

(1)	~~SHOW~~: ○□$P → $□○P	/0	/0	CD
(2)	| ○□$P	/0	/0	As
(3)	| ~~SHOW~~: $□○P	/0	/0	4,$I
(4)	| ~~SHOW~~: □○P	/0	/0	ND
(5)	| ~~SHOW~~: ○P	/01	/0	5,○I
(6)	| ~~SHOW~~: P	/0	/0	DD
(7)	| | □$P	/0	/0	2,○O
(8)	| | $P	/0	/0	5,□O(t)
(9)	| | P	/0	/0	6,$O

So we still don't have very strong evidence that '○' and '$' do much semantically. Consider the following examples of attempted derivations.

Example 7

(1)	SHOW: ○□$P → □P	/0	/0	CD
(2)	○□$P	/0	/0	As
(3)	SHOW: □P	/0	/0	ND
(4)	SHOW: P	/01	/0	DD
(5)	□$P	/0	/0	2,○O
(6)	$P	/01	/0	5,□O
(7)	P	/01	/01	6,$O

Example 8

(1)	SHOW: $□○P → □P	/0	/0	CD
(2)	$□○P	/0	/0	As
(3)	SHOW: □P	/0	/0	ND
(4)	SHOW: P	/01	/0	DD
(5)	□○P	/0	/0	2,$O
(6)	○P	/01	/0	5,□O
(7)	P	/01	/01	6,○O

Notice that we are trying to show P/01/0, but what we achieve is P/01/01.[1]

5. The Problem of Necessitation

We next note that the following formula is valid

P ↔ ○P

as we have already seen, but its necessitation

□(P ↔ ○P)

is not valid. Consider the following attempted derivation.

(1)	SHOW: □(P ↔ ○P)	/0	/0	ND
(2)	SHOW: P ↔ ○P	/01	/0	↔D
(3)	P	/01	/0	As
(4)	SHOW: ○P	/01	/0	5,○I
(5)	SHOW: P	/0	/0	???

This derivation fails, because we propose to show P/0/0 but all we can show is P/01/0.

[1] These derivations can be completed in the logic proposed by Åqvist (+++reference+++), because he includes the following additional rule – from $\mathbb{A}/i/j$ one is entitled to infer $\mathbb{A}/i/k$, provided $\mathbb{A}$ is **atomic**.

6. System 2D(2)

In order to deal with the problem of necessitation, Humberstone and Davies[2] propose an alternative account of validity in 2D logics. Translating their account into our framework, we adjust the index rules so that, instead of originally taking the world of utterance as both the initial active world and the initial actual world, we take the world of utterance only as the initial active world, and we take an arbitrarily chosen second world as the initial actual world.

This means that many of the formulas that are provable in 2D(1) are not provable in the new system, 2D(2). Let us consider a few of them. Notice in these derivations that we start with distinct initial indices, 0 and 1.

Example 1

(1)	SHOW: $P ↔ P	/0	/1	↔D	
(2)	$P	/0	/1	As	
(3)	SHOW: P	/0	/1	2,$O	⋇⋇⋇
(4)	P	/0	/1	As	
(5)	SHOW: $P	/0	/1	4,$I	⋇⋇⋇

Example 2

(1)	SHOW: ○P ↔ P	/0	/1	↔D	
(2)	○P	/0	/1	As	
(3)	SHOW: P	/0	/1	2,○O	⋇⋇⋇
(4)	P	/0	/1	As	
(5)	SHOW: ○P	/0	/1	4,○I	⋇⋇⋇

Example 4

(1)	SHOW: $○P ↔ P	/0	/1	↔D	
(2)	$○P	/0	/1	As	
(3)	SHOW: P	/0	/1	4,○O	⋇⋇⋇
(4)	○P	/0	/1	2,$O	⋇⋇⋇
(5)	P	/0	/1	As	
(6)	SHOW: $○P	/0	/1	7,$I	⋇⋇⋇
(7)	○P	/0	/1	5,○I	⋇⋇⋇

7. System 2D(3)

We have considered three renditions of actuality – (1) simple (rigid) actuality, (2) implicit-scope (flexible) actuality, (3) simple two-dimensional actuality implemented in Systems 2D(1) and 2D(2).

As we have seen previously, rigid actuality is afflicted by two serious problems. On the one hand, there is a logical problem – the problem of necessitation. On the other hand, there is a semantic problem – the problem of embedded modalities.

Also, as we have seen previously, the implicit-scope (flexible) actuality operator avoids both problems. Unfortunately, we face serious difficulties when we attempt to add this operator to modal systems that extend System K. In order to avoid these difficulties, we have to make specific restrictions on the □O and ◇I rules for systems that extend K. These restrictions are motivated, but they undermine the elegance of the system.

As seen earlier, 2D(1) avoids the semantic problem, but not the logical problem. On the other hand, a slight modification of the definition of validity given by System 2D(2) – which requires every derivation to start at an arbitrary pair of worlds – avoids the logical problem as well.

Nevertheless, there is a residual oddity about 2D(1) and 2D(2) – namely, it treats '$' and '○' as formally completely symmetrical, even though their intended roles are quite asymmetrical.

[2] Davies, Martin and Humberstone, Lloyd. "Two notions of necessity", *Philosophical Studies*, 38 (1980) , 1-30.

Specifically, whereas '○' is a genuine phonetically-realized operator, '$' is a purely subsidiary operator that is not pronounced, but merely serves to mark the scope of its superior.

In the remainder of this chapter, we propose an account of actuality that explicitly formalizes the asymmetry of '$' and '○'.

As a *first approximation*, the syntax of 2D(3) is the same as 2D(1) and 2D(2), according to which '$' is a one-place sentential operator. This is not the whole story, however. In order to render '○' as an operator whose scope is marked by '$', we adjust the syntax of 2D(3) by adding the following further syntactic rule.

Further Condition for Well-Formedness

Let Φ be a well-formed formula. Suppose *o* is an occurrence of '○' in Φ. Then there is an occurrence ***s*** of '$' in Φ such that *o* lies inside the scope of ***s***.

In other words, we cannot have an actuality operator without an explicit scope operator covering it. This is analogous to the situation with our implicit-scope actuality operator – every occurrence of '○' must fall inside the scope of at least one modal operator (□ or ◇), with respect to which it has wide scope. The difference between our new scoped-actuality operator and the original operator is that the new operator requires that its scope is explicitly syntactically marked.

Notice that both languages agree in excluding the following sorts of formulas.[3]

○P
○P ↔ P
○(P ↔ P)

On the other hand, although the following all count as well-formed in the language of implicit-scope actuality,

□○P
◇□○○P
□∃*x*(○F*x* & G*x*)
□◇∀*x*(○F*x* → G*x*)
□◇∀*x*(○○F*x* → G*x*)

none of these count as well-formed in 2D(3).

In place of the above formulas, the new language substitutes the following formulas in which the implicit scopes are explicitly marked.

$□○P
$◇□○P
$□∃*x*(○F*x* & G*x*)
□$◇∀*x*(○F*x* → G*x*)
$□◇∀*x*(○F*x* → G*x*)

Furthermore, those formulas that neither language can abide may be converted into semantically acceptable formulas simply by covering all the "naked" actuality operators. The following are examples of conversions.

[3] Note carefully that these formulas are syntactically well-formed, but semantically ill-formed in QML+O, whereas they are syntactically ill-formed, and hence semantically ill-formed, in 2D(3)

$○P
$○P ↔ P
$(○P ↔ P)
$○(P ↔ P)

8. The Semantics of 2D(3)

The proposed semantics is quasi-two-dimensional; every formula requires an "active" world for its semantic evaluation; some formulas additionally require an "actual" world, including every formula whose main connective is '○'. Here is where the operator '$' comes in – the only way a world becomes "actual" is when one applies the operator '$'. Before one applies the operator '$', there is no "actual" world, although of course there is an "active" world – initially, the world of utterance. Accordingly, formulas that make reference to the "actual" world – i.e., formulas of the form ○𝔽 – cannot be evaluated until one has an "actual" world. This only sounds strange if one insists on reading 'actually' as 'at the actual world'. It is probably better to read 'actually' as 'at the special/stored world'. When we start, no world is special/stored.

The rules for □ and ◇ are the same as before in formal appearance. The difference pertains to the second slot, which may be empty/null, which we indicate by '∅'.[4] If there is a second index, then the standard modal rules preserve it. The following are two examples of applying the □O rule.

(1)	□P	/0	/∅	Pr
(2)	P	/01	/∅	1,□O
(3)	□P	/01	/01	Pr
(4)	P	/012	/01	1,□O

Here, the expression '∅' refers to the null index. In devising our system, we could also leave that position blank. We use '∅' for the sake of symmetry in the presentation.

The rules pertaining to '$' look exactly like those for 2D logic.

$O/I

$Φ	/*i*	/*j*
═══		
Φ	/*i*	/*i*

The difference is that, whereas *i* must be a genuine (non-null) index, *j* can be a genuine index or the null index. Alternatively speaking, the line Φ/∅/∅ is semantically inadmissible.

The rules for '○' make specific mention of the null index.

○O/I

○Φ	/*i*	/*j*
═══		
Φ	/*j*	/∅

The presence of the null index in these rules makes a big difference in derivations, as we see later.

In order to see how this system works, we consider a fairly simple example.

[4] The first slot may not be null.

(1)	$\forall x(Fx \rightarrow \Box Gx)$	/0	/∅	Pr
(2)	~~SHOW~~: $\forall x\$\Box(\bigcirc Fx \rightarrow Gx)$	/0	/∅	UD
(3)	~~SHOW~~: $\$\Box(\bigcirc Fa \rightarrow Ga)$	/0	/∅	4,$I
(4)	~~SHOW~~: $\Box(\bigcirc Fa \rightarrow Ga)$	/0	/0	ND
(5)	~~SHOW~~: $\bigcirc Fa \rightarrow Ga$	/01	/0	CD
(6)	| $\bigcirc Fa$	/01	/0	As
(7)	| ~~SHOW~~: Ga	/01	/0	9,□O
(8)	| | Fa	/0	/∅	5,○O
(9)	| | ~~SHOW~~Ga	/0	/∅	1,8,QL

Notice that allowing the second index to be null means that some formulas cannot be evaluated at some index pairs, as exemplified in the following derivation.

(1)	SHOW: : $\bigcirc P \rightarrow P$	/0	/∅	CD	※
(2)	$\bigcirc P$	/0	/∅	As	※
(3)	SHOW: : P	/0	/∅		
(4)	P	/∅	/∅	2,○O	※

Some lines are marked by '※', which we use to mark semantically inadmissible lines. For example, line (4) is inadmissible because it attempts to evaluate P at the pair ∅/∅, which is disallowed. This means that line (2) is also inadmissible, so line (1) is also inadmissible. The formula in question is invalid – not because it can be made false, but because it is semantically inadmissible.

In order for a formula to be semantically well-formed, it cannot have any "naked" occurrences of '○'. So, in order to convert the above formula into a semantically well-formed formula, we must cover the naked occurrence of '○'. This can be done in one of two ways, generating the following lines.

(1)	SHOW: : $\$\bigcirc P \rightarrow P$	/0	/∅
(2)	SHOW: : $\$(\bigcirc P \rightarrow P)$	/0	/∅

The first formula is provable, as seen in the following derivation.

(1)	~~SHOW~~: $\$\bigcirc P \rightarrow P$	/0	/∅	CD
(2)	| $\$\bigcirc P$	/0	/∅	As
(3)	| ~~SHOW~~: P	/0	/∅	DD
(4)	| | $\bigcirc P$	/0	/0	2,$O
(5)	| | P	/0	/∅	4,○O

By contrast, consider the following uncompleted derivation.

(1)	SHOW: : $\$(\bigcirc P \rightarrow P)$	/0	/∅	$I
(2)	SHOW: : $\bigcirc P \rightarrow P$	/0	/0	CD
(3)	$\bigcirc P$	/0	/0	AS
(4)	SHOW: : P	/0	/0	DD
(5)	P	/0	/∅	3,○O

9. How Explicit-Scope Actuality Escapes the Difficulties that Face Rigid Actuality

According to the traditional understanding of 'actually', the ○-operator refers to the actual world, which is understood as the world of utterance – the original world of evaluation. According to that interpretation of '○', the formula

$$\bigcirc P \leftrightarrow P$$

is valid, but unfortunately its necessitation

$$\Box(\bigcirc P \leftrightarrow P)$$

is not valid. This is a serious logical problem, but not a serious semantic problem. However, the traditional reading of 'actually' faces a serious semantic problem, also – it cannot account for embedded modalities [for example, 'Jay believes my yacht could be bigger than it (actually) is'].

Simple 2D-actuality solves the semantic problem of embedded modalities, but it does not solve the logical problem of necessitation.

If we interpret 'actually' from the outset as a scope operator, akin to 'any' and 'either', then we face neither of these problems. In particular, we deny the validity of '○P ↔ P' because we deny its well-formedness. Once we write it properly, as follows (for example),

$○P ↔ P

the result is valid, but so is its necessitation,

□($○P ↔ P)

as can be readily demonstrated.

10. How Explicit-Scope Actuality Escapes the Difficulties that Face Implicit-Scope Actuality

Implicit-scope (flexible) actuality also solves the two major problems that beset rigid actuality, although it faces difficulties when we attempt to expand our logic beyond System K. Recall the following derivation in uncorrected System-T+○.

(1)	~~SHOW~~: P → □P	/0	CD	
(2)	| P	/0	As	
(3)	| ~~SHOW~~: □P	/0	ND	
(4)	| ~~SHOW~~: P	/01	DD	
(5)	| | ○P	/01	2,○I	
(6)	| | ◇○P	/01	5,◇I(t)	※※※
(7)	| | ○P	/012	6,◇O	
(8)	| | P	/01	7,○O	

The way we block this derivation is at line (6), by requiring that ◇I(t) take as input only semantically well-formed formulas. This blocks the inference, but it does not seem particularly elegant.

If we adopt an explicit-scope actuality operator, as formalized in System 2D(3), we see more easily how the above derivation breaks the rules.

(1)	~~SHOW~~: P → □P	/0	/∅	CD	
(2)	| P	/0	/∅	As	
(3)	| ~~SHOW~~: □P	/0	/∅	ND	
(4)	| ~~SHOW~~: P	/01	/∅	DD	
(5)	| | ○P	/01	/∅	2,○I	※※※
(6)	| | ◇○P	/01	/∅	5,◇I(t)	
(7)	| | ○P	/012	/∅	6,◇O	
(8)	| | P	/01	/∅	7,○O	※※※

Notice in particular that lines (5) and (8) are inadmissible in 2D(3). The following is an adjusted derivation that follows the rules.

(1)	SHOW: : P → □P	/0	/∅	CD
(2)	P	/0	/∅	As
(3)	SHOW: : □P	/0	/∅	ND
(4)	SHOW: : P	/01	/∅	???
(5)	○P	/01	/0	2, ○I
(6)	◇○P	/01	/0	5,◇I(t)
(7)	○P	/012	/0	6,◇O
(8)	P	/01	/0	7,○O

Notice that we are trying to show P/01/∅, and all we can achieve is P/01/0.

The other bad derivation from T+○ is the dual to above.

(1)	~~SHOW~~: ◇P → P	/0	CD
(2)	| ◇P	/0	As
(3)	| ~~SHOW~~: P	/0	DD
(4)	| | P	/01	2,◇O
(5)	| | ~~SHOW~~: □○P	/01	ND
(6)	| | ~~SHOW~~: ○P	/012	DD
(7)	| | | ○P	/012	4,○I
(8)	| | ○P	/01	5,□O(t) ※※※
(9)	| | P	/0	8,○O

In our original system, we bust this derivation at line (8) – we prohibit applications of □O(t) that involve a semantically-ill-formed formula.

When we attempt to replicate this derivation in 2D(3), we obtain the following uncompleted derivation.

(1)	SHOW: ◇P → P	/0	/∅	CD
(2)	◇P	/0	/∅	As
(3)	SHOW: P	/0	/∅	DD
(4)	P	/01	/∅	2,◇O
(5)	~~SHOW~~: □○P	/01	/01	ND
(6)	~~SHOW~~: ○P	/012	/01	DD
(7)	| ○P	/012	/01	4,○O
(8)	○P	/01	/01	5,□O(t)
(9)	P	/01	/∅	8,○O

Notice that we are trying to show P/0/∅, and all we can achieve is P/01/∅.

11. Multiple Applications of '$' and '○'

When we iterate '$', with no intervening modal operators, the embedded occurrences of '$' are redundant. For example, we can prove the following about 2D(3).

$\vdash \$\$\Phi \leftrightarrow \$\Phi$
$\vdash \$(\Phi \rightarrow \$\Psi) \leftrightarrow \$(\Phi \rightarrow \Psi)$
$\vdash \$(\Phi \,\&\, \$\Psi) \leftrightarrow \$(\Phi \,\&\, \Psi)$

When there are intervening occurrences of modal operators, things change. For example:

$\nvdash \$\Box\$\Phi \leftrightarrow \$\Box\Phi$

In this particular case, the outer occurrence is redundant, since we have:

$\vdash \$\Box\$\Phi \leftrightarrow \Box\$\Phi$

In the following formula, neither occurrence of '$' is redundant.

$\$\Box(\Phi \rightarrow \$\Box\Psi) \rightarrow \Box(\Phi \rightarrow \$\Box\Psi)$
$\$\Box(\Phi \rightarrow \$\Box\Psi) \rightarrow \$\Box(\Phi \rightarrow \Box\Psi)$

Next, let us consider '○'. First, in 2D(1) and 2D(2), '○○' and '○' are equivalent, as seen in the following schematic derivations.

(1)	○○Φ	/*i*	/*j*	As
(2)	~~SHOW~~: ○Φ	/*i*	/*j*	DD
(3)	○Φ	/*j*	/*j*	1,○O
(4)	Φ	/*j*	/*j*	3,○O
(5)	○Φ	/*i*	/*j*	4,○I
(6)	○Φ	/*i*	/*j*	As
(7)	~~SHOW~~: ○○Φ	/*i*	/*j*	DD
(8)	Φ	/*j*	/*j*	1,○O
(9)	○Φ	/*j*	/*j*	3,○O
(10)	○○Φ	/*i*	/*j*	4,○I

What about 2D(3)? Consider the following derivation attempt.

(1)	○○Φ	/*i*	/*j*	As ☠	
(2)	SHOW: : ○Φ	/*i*	/*j*	DD	
(3)	○Φ	/*j*	/∅	1,○O	※
(4)	Φ	/∅	/∅	3,○O	※

Notice that line (4) is clearly semantically inadmissible. This means that the semantically dependent lines – (3) and (1) – are also semantically inadmissible.

The converse derivation faces a similar problem.

(1)	○Φ	/*i*	/*j*	As	
(2)	SHOW: : ○○Φ	/*i*	/*j*	DD	
(3)	Φ	/*j*	/∅	1,○O	
(4)	○Φ	/*j*	/*j*	3,○I	
(5)	○○Φ	/*i*	/*j*	4,○I	※※※

In this case, line (5) doesn't follow from line (4); double check the ○I rule!

This suggests that our characterization of well-formedness requires adjustment. A further necessary condition on well-formedness is given as follows.

Further Condition for Well-Formedness (Take 2)

Let Φ be a well-formed formula. Suppose *o* is an occurrence of '○' in Φ. Then there is an occurrence *s* of '$' in Φ such that *o* lies inside the scope of *s*, and such that no occurrence of '○' lies between *o* and *s*.