Propositional Calculus II:

Two more Rules of Inference/Inference Schemata

(`conditional elim` = *modus ponens;*

proof by cases = *disjunction elimination*),

Applying Them to Additional Motivating Problems

Selmer Bringsjord • James Oswald

Rensselaer AI & Reasoning (RAIR) Lab
Department of Cognitive Science
Department of Computer Science
Lally School of Management & Technology
Rensselaer Polytechnic Institute (RPI)
Troy, New York 12180 USA

Intro to Logic
1/27/2025

RAIR
Rensselaer AI and Reasoning Lab

Logic-and-AI in the news

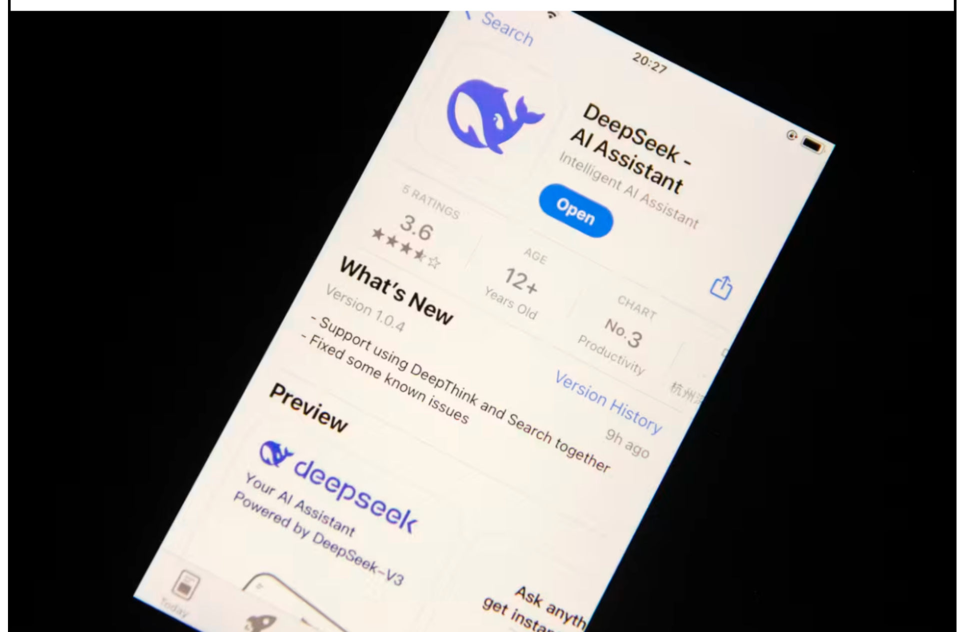
…

Apropos of Selmer's Claim re. Commodification

TECH Following

Silicon Valley Is Raving About a Made-in-China AI Model

DeepSeek is called 'amazing and impressive' despite working with less-advanced chips



A chatbot app developed by the Chinese AI company DeepSeek. (PHOTO: RAFFAELE HUANG/WSJ)

By Raffaele Huang Follow

Updated Jan 26, 2025 12:00 a.m. ET

Listen to this article

7 minutes

SINGAPORE—A Chinese artificial-intelligence company has Silicon Valley marveling at how its programmers nearly matched American rivals despite using inferior chips.

Logistics, again …

The Starting Code to Purchase in Bookstore

Your code for starting the registration process is:

To access HyperGrader, HyperSlate, the license agreement, and to obtain the textbook LAMA-BDLA, go to::

https://rpi.logicamodernapproach.com

The Starting Code to Purchase in Bookstore

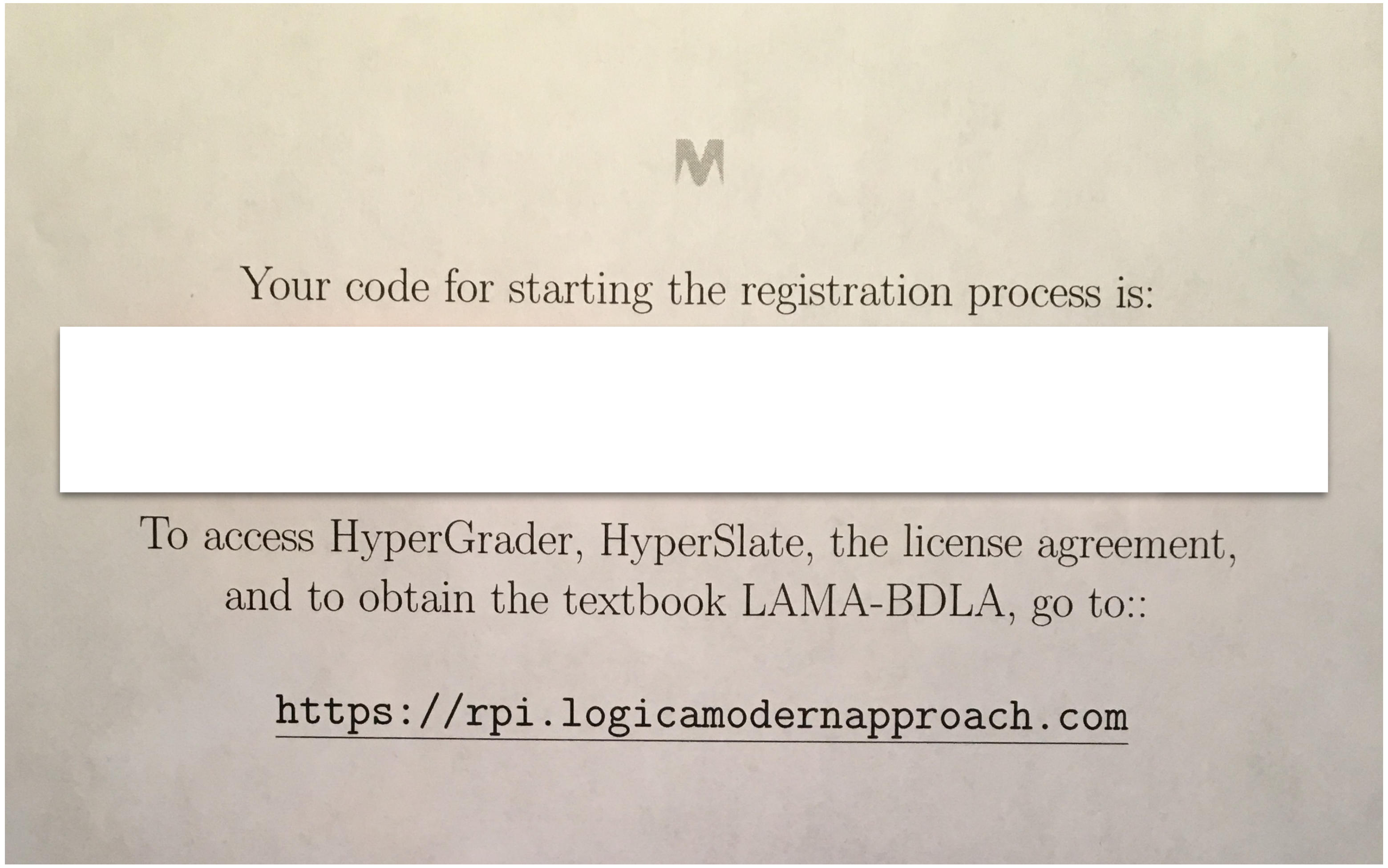
Your code for starting the registration process is:

To access HyperGrader, HyperSlate, the license agreement, and to obtain the textbook LAMA-BDLA, go to::

https://rpi.logicamodernapproach.com

Once seal broken on envelope, no return. Remember from first class, any reservations, opt for "Stanford" paradigm, with its software instead of LAMA® paradigm!

The Starting Code to Purchase in Bookstore


Your code for starting the registration process is:

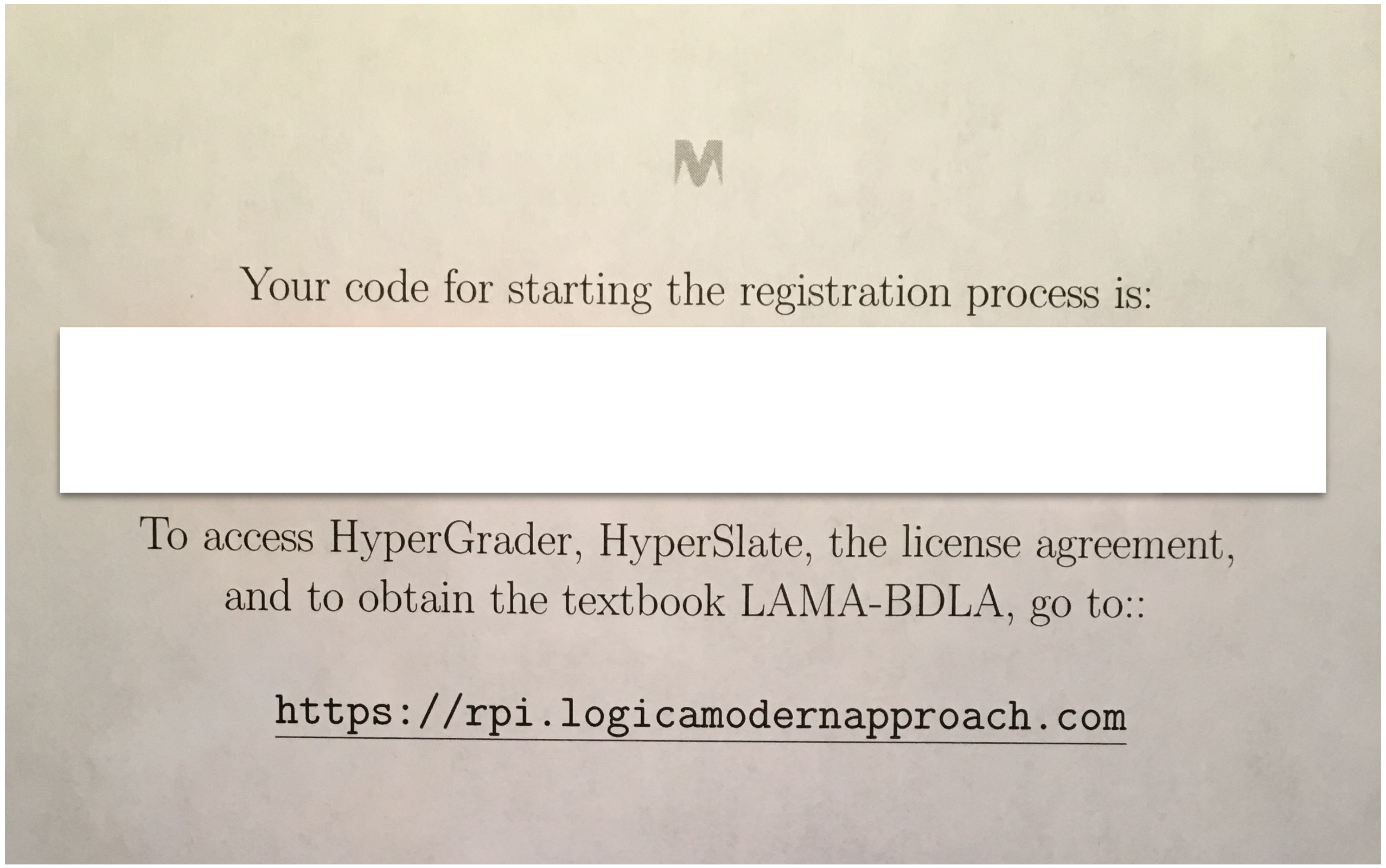
To access HyperGrader, HyperSlate, the license agreement, and to obtain the textbook LAMA-BDLA, go to::

https://rpi.logicamodernapproach.com

Once seal broken on envelope, no return. Remember from first class, any reservations, opt for "Stanford" paradigm, with its software instead of LAMA® paradigm!

The email address you enter is case-sensitive!

The Starting Code to Purchase in Bookstore


Your code for starting the registration process is:

To access HyperGrader, HyperSlate, the license agreement, and to obtain the textbook LAMA-BDLA, go to::

https://rpi.logicamodernapproach.com

Once seal broken on envelope, no return. Remember from first class, any reservations, opt for "Stanford" paradigm, with its software instead of LAMA® paradigm!

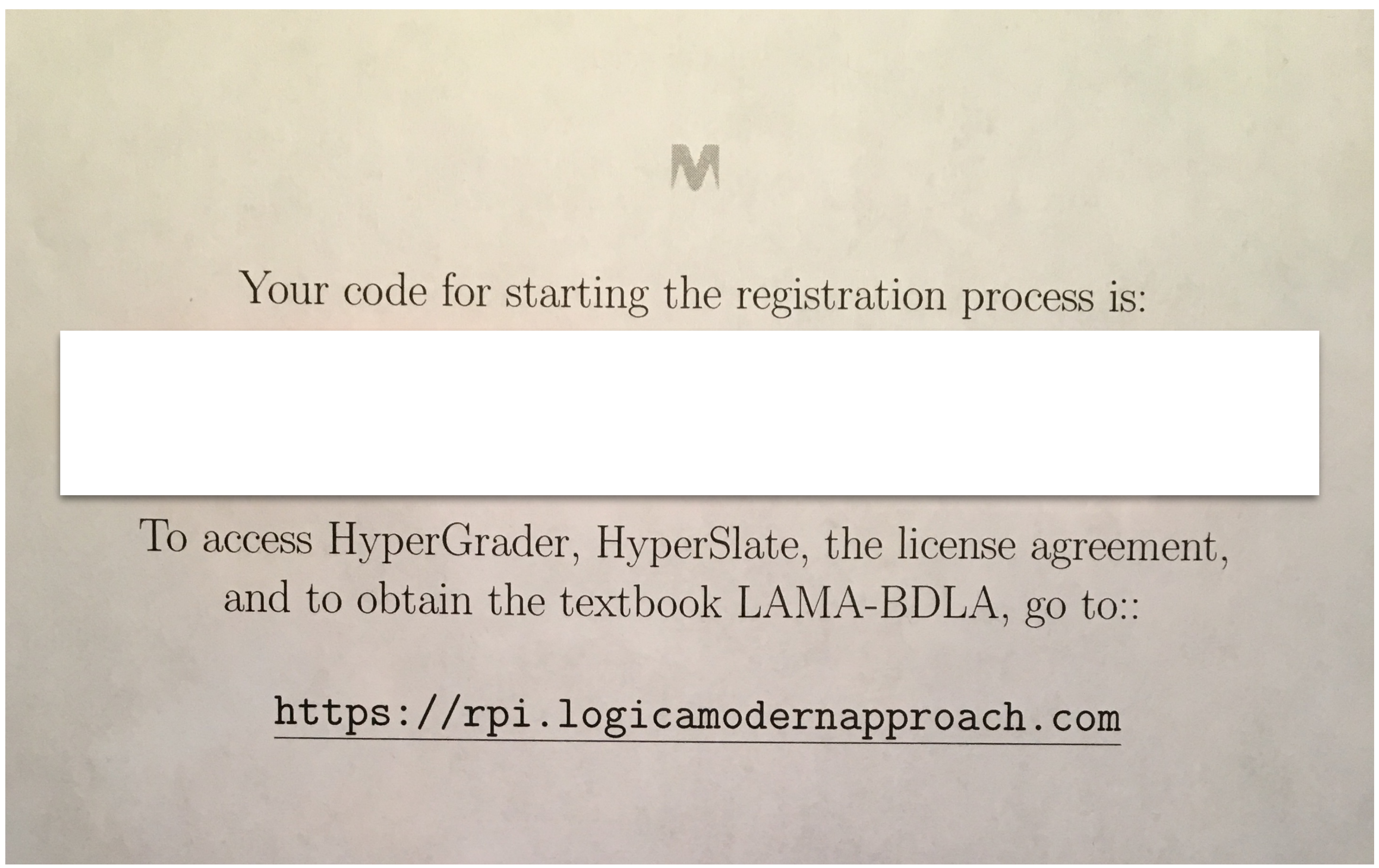
The email address you enter is case-sensitive!

Your OS and browser must be fully up-to-date; Chrome is the best choice, browser-wise (though I use Safari).

The Starting Code Purchased in Bookstore Should By Now've Been Used to Register & Subsequently Sign In

First two Prop-Calc Practice problems:

`switching_conjuncts_fine`, `switching_disjuncts_fine`



Your code for starting the registration process is:

To access HyperGrader, HyperSlate, the license agreement, and to obtain the textbook LAMA-BDLA, go to::

https://rpi.logicamodernapproach.com

E-Housekeeping Pts (again)

E-Housekeeping Pts (again)

- Must input your RIN. (This is your “University ID.”)

E-Housekeeping Pts (again)

- Must input your RIN. (This is your "University ID.")
- Make sure OS fully up-to-date.

E-Housekeeping Pts (again)

- Must input your RIN. (This is your "University ID.")
- Make sure OS fully up-to-date.
- Make sure browser fully up-to-date.

E-Housekeeping Pts (again)

- Must input your RIN. (This is your "University ID.")
- Make sure OS fully up-to-date.
- Make sure browser fully up-to-date.
- Chrome best (but I use Safari).

E-Housekeeping Pts (again)

- Must input your RIN. (This is your "University ID.")
- Make sure OS fully up-to-date.
- Make sure browser fully up-to-date.
- Chrome best (but I use Safari).
- Always work in the same browser window with multiple tabs; must do this with email and HyperGrader® & HyperSlate®.

Propositional Calculus II:

Two more Rules of Inference/Inference Schemata
(`conditional elim` = *modus ponens;*
proof by cases = *disjunction elimination*),
Applying Them to Additional Motivating Problems

Selmer Bringsjord • James Oswald

Rensselaer AI & Reasoning (RAIR) Lab
Department of Cognitive Science
Department of Computer Science
Lally School of Management & Technology
Rensselaer Polytechnic Institute (RPI)
Troy, New York 12180 USA

Intro to Logic
1/27/2025

Last time we introduced and and lauded the power of **oracles**, and questions … and now … picking up where we left off …

“NYS 3” Revisited

Given the statements

$\neg\neg c$

$c \rightarrow a$

$\neg a \vee b$

$b \rightarrow d$

$\neg(d \vee e)$

which one of the following statements must also be true?

$\neg c$

e

h

$\neg a$

all of the above

"NYS 3" Revisited

Given the statements

$\neg\neg c$

$c \rightarrow a$

$\neg a \vee b$

$b \rightarrow d$

$\neg(d \vee e)$

which one of the following statements must also be true?

$\neg c$

e

h

$\neg a$

all of the above

“NYS 3” Revisited

Given the statements

$\neg\neg c$

$c \rightarrow a$

$\neg a \vee b$

$b \rightarrow d$

$\neg(d \vee e)$

After last class, should have explored if you are registered … Show in HyperSlate® as I did that each of the first four options can be proved using the PC entailment (= provability) oracle.

which one of the following statements must also be true?

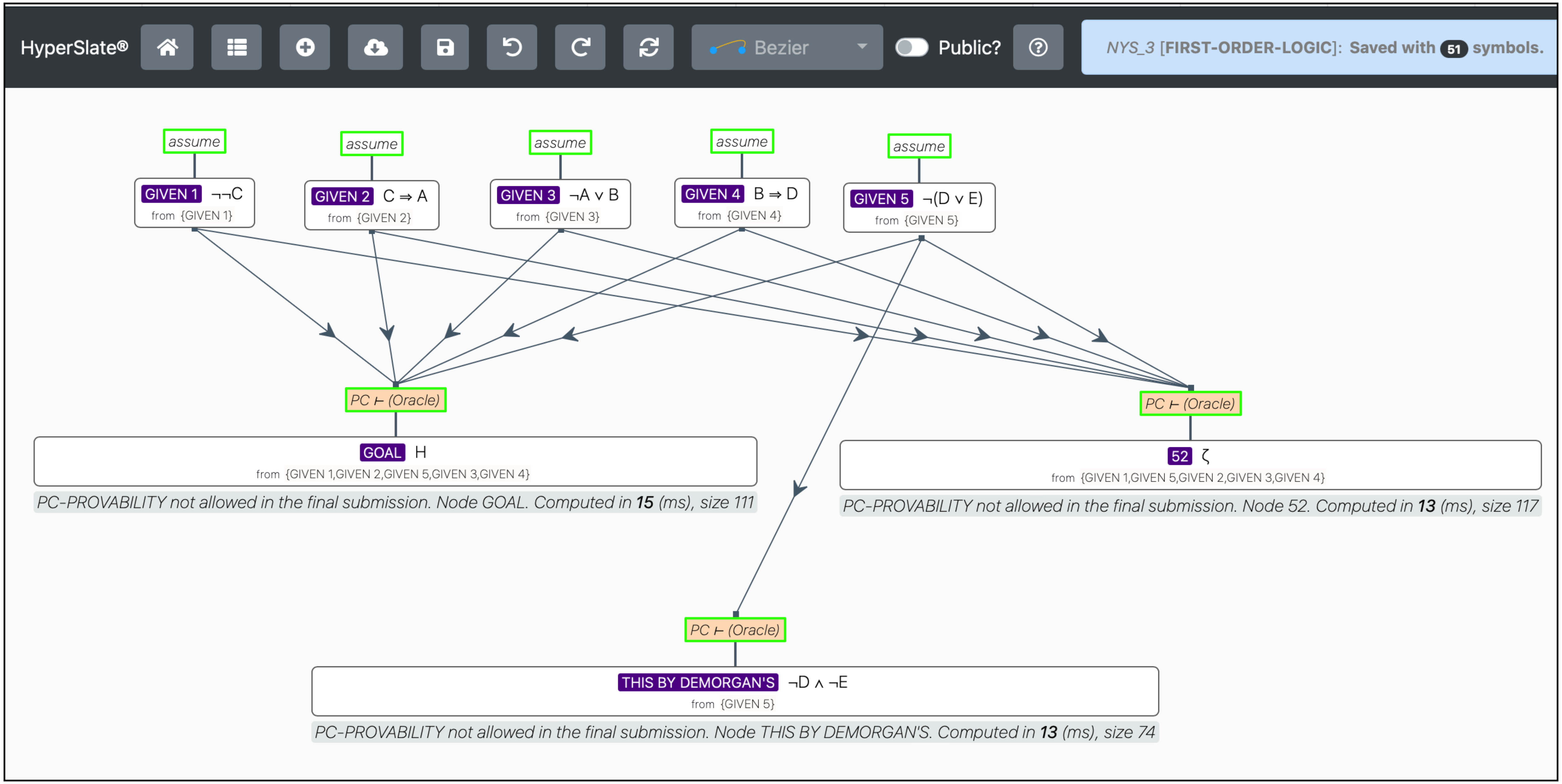
$\neg c$

e

h

$\neg a$

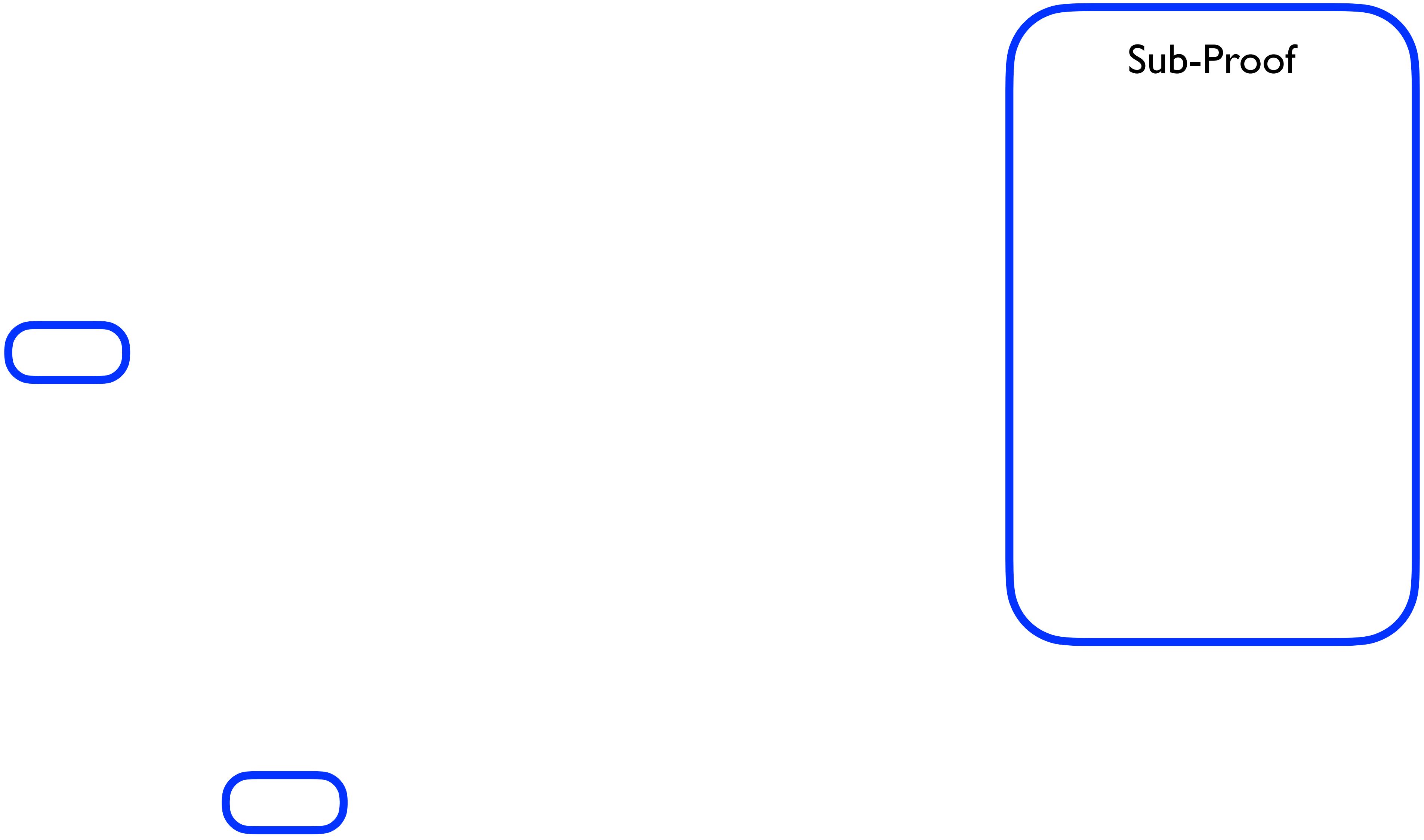
all of the above


HyperSlate®
Bezier
Public?
NYS_3 [FIRST-ORDER-LOGIC]: Saved with 51 symbols.
assume
assume
assume
assume
assume
GIVEN 1 ¬¬C
from {GIVEN 1}
GIVEN 2 C ⇒ A
from {GIVEN 2}
GIVEN 3 ¬A ∨ B
from {GIVEN 3}
GIVEN 4 B ⇒ D
from {GIVEN 4}
GIVEN 5 ¬(D ∨ E)
from {GIVEN 5}
PC ⊢ (Oracle)
GOAL H
from {GIVEN 1,GIVEN 2,GIVEN 5,GIVEN 3,GIVEN 4}
PC-PROVABILITY not allowed in the final submission. Node GOAL. Computed in 15 (ms), size 111
PC ⊢ (Oracle)
52 ζ
from {GIVEN 1,GIVEN 5,GIVEN 2,GIVEN 3,GIVEN 4}
PC-PROVABILITY not allowed in the final submission. Node 52. Computed in 13 (ms), size 117
PC ⊢ (Oracle)
THIS BY DEMORGAN'S ¬D ∧ ¬E
from {GIVEN 5}
PC-PROVABILITY not allowed in the final submission. Node THIS BY DEMORGAN'S. Computed in 13 (ms), size 74

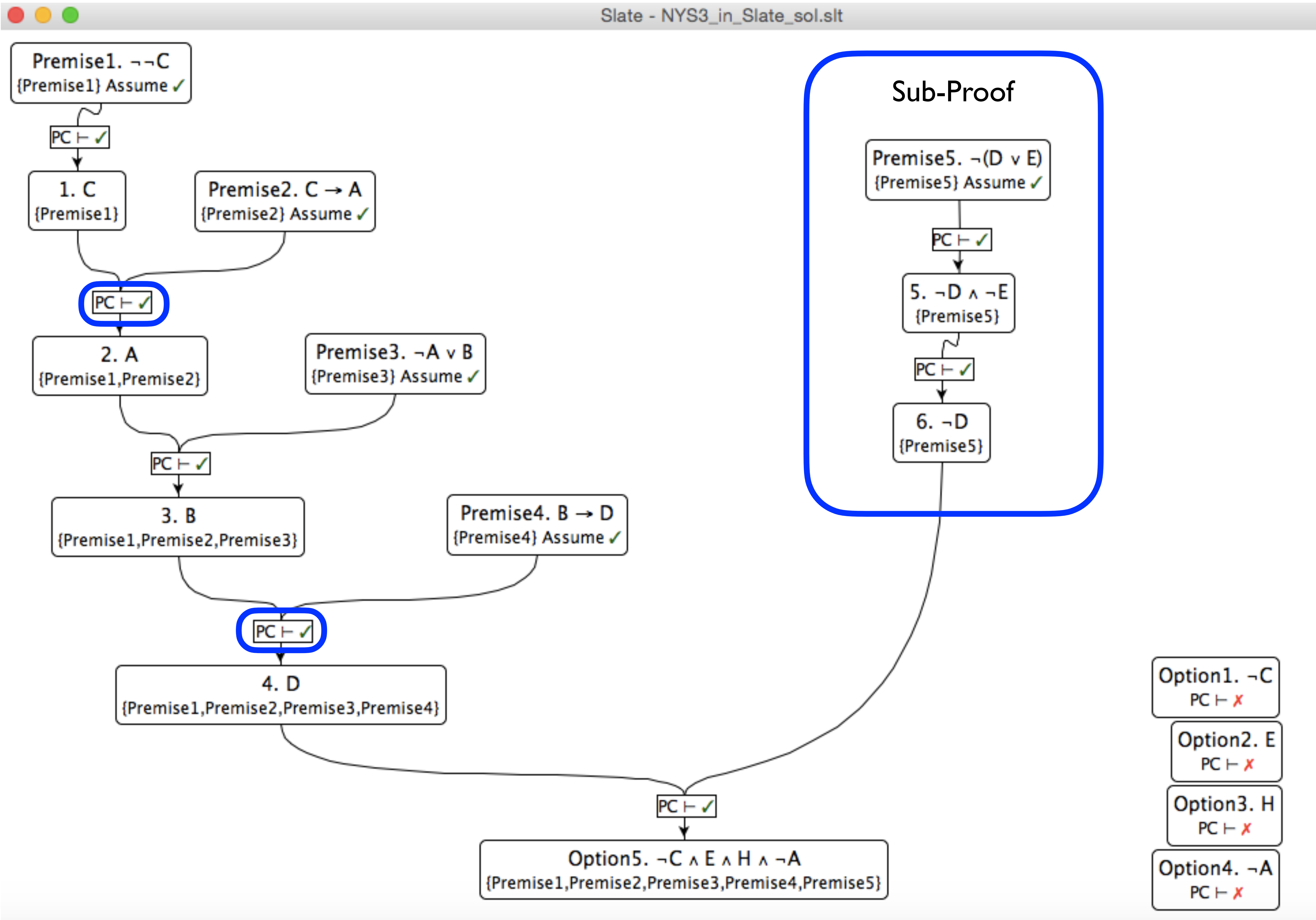
Proof Plan …

Proof Plan …

Proof Plan . . .

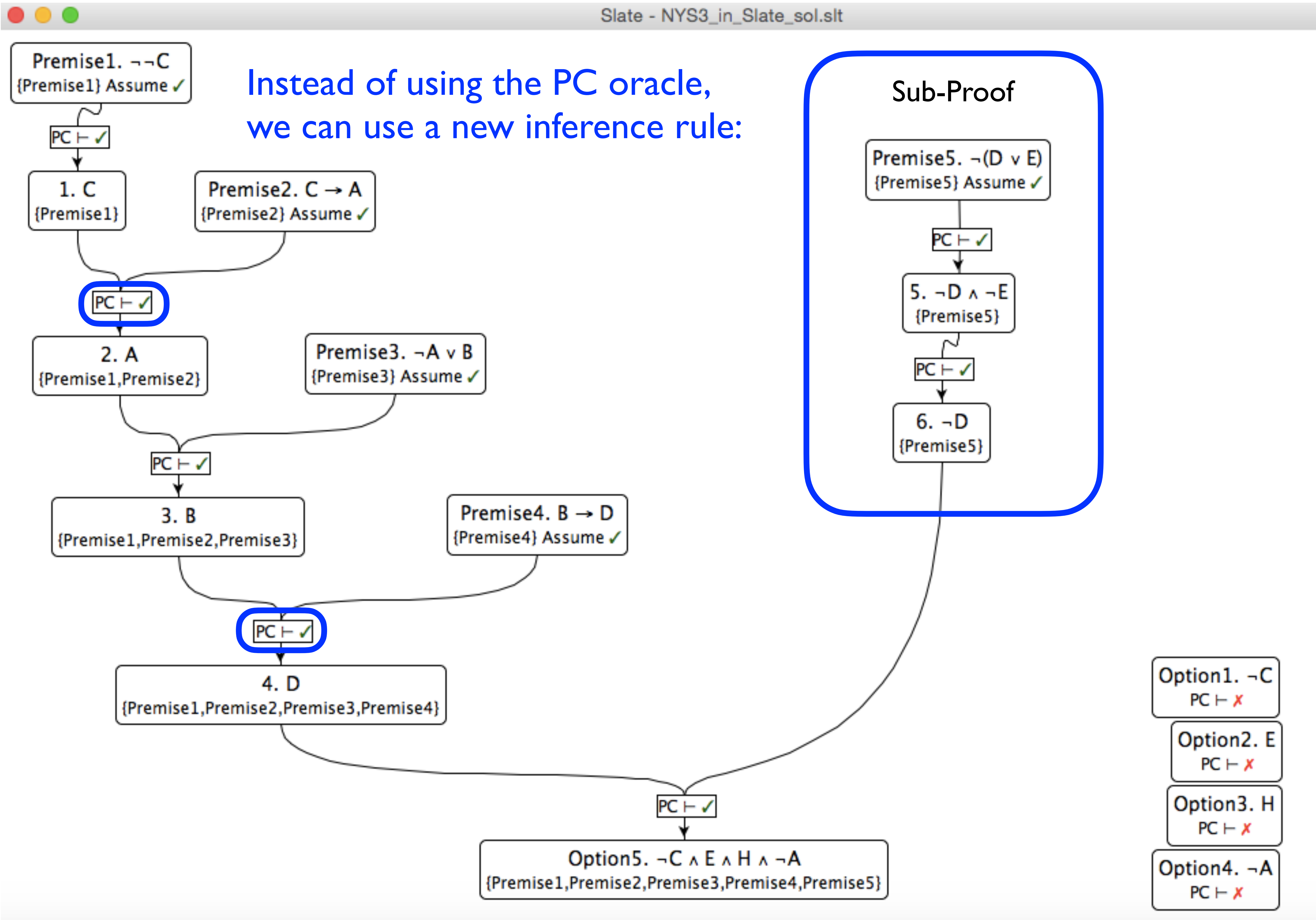


Proof Plan …



Slate - NYS3_in_Slate_sol.slt
Premise1. ¬¬C
{Premise1} Assume ✓
PC ⊢ ✓
1. C
{Premise1}
Premise2. C → A
{Premise2} Assume ✓
PC ⊢ ✓
2. A
{Premise1,Premise2}
Premise3. ¬A ∨ B
{Premise3} Assume ✓
PC ⊢ ✓
3. B
{Premise1,Premise2,Premise3}
Premise4. B → D
{Premise4} Assume ✓
PC ⊢ ✓
4. D
{Premise1,Premise2,Premise3,Premise4}
Sub-Proof
Premise5. ¬(D ∨ E)
{Premise5} Assume ✓
PC ⊢ ✓
5. ¬D ∧ ¬E
{Premise5}
PC ⊢ ✓
6. ¬D
{Premise5}
PC ⊢ ✓
Option5. ¬C ∧ E ∧ H ∧ ¬A
{Premise1,Premise2,Premise3,Premise4,Premise5}
Option1. ¬C
PC ⊢ ✗
Option2. E
PC ⊢ ✗
Option3. H
PC ⊢ ✗
Option4. ¬A
PC ⊢ ✗

Proof Plan …



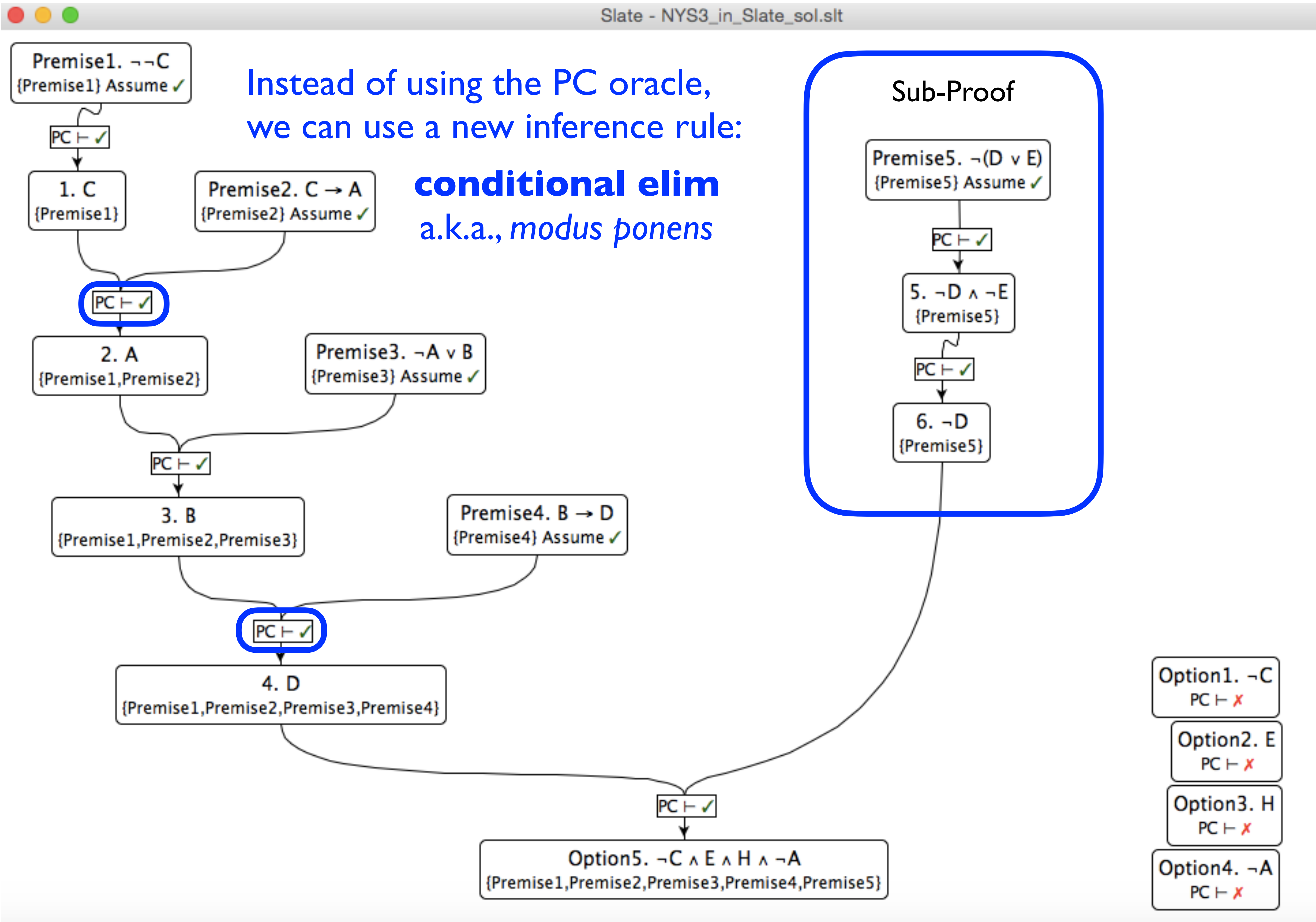
Slate - NYS3_in_Slate_sol.slt
Instead of using the PC oracle,
we can use a new inference rule:
Premise1. ¬¬C
{Premise1} Assume ✓
PC ⊢ ✓
1. C
{Premise1}
Premise2. C → A
{Premise2} Assume ✓
PC ⊢ ✓
2. A
{Premise1,Premise2}
Premise3. ¬A ∨ B
{Premise3} Assume ✓
PC ⊢ ✓
3. B
{Premise1,Premise2,Premise3}
Premise4. B → D
{Premise4} Assume ✓
PC ⊢ ✓
4. D
{Premise1,Premise2,Premise3,Premise4}
Sub-Proof
Premise5. ¬(D ∨ E)
{Premise5} Assume ✓
PC ⊢ ✓
5. ¬D ∧ ¬E
{Premise5}
PC ⊢ ✓
6. ¬D
{Premise5}
PC ⊢ ✓
Option5. ¬C ∧ E ∧ H ∧ ¬A
{Premise1,Premise2,Premise3,Premise4,Premise5}
Option1. ¬C
PC ⊢ ✗
Option2. E
PC ⊢ ✗
Option3. H
PC ⊢ ✗
Option4. ¬A
PC ⊢ ✗

Proof Plan …

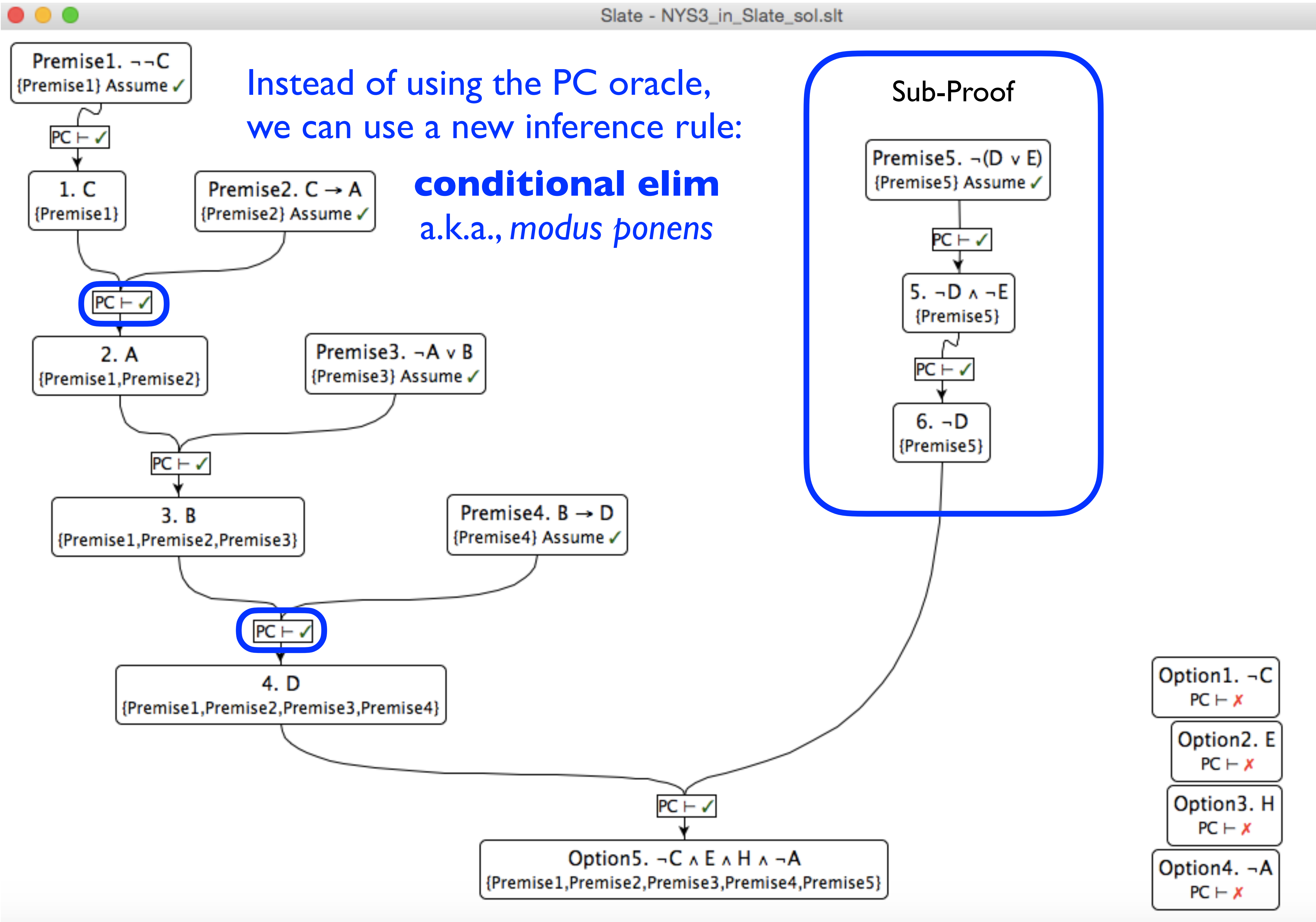
Instead of using the PC oracle, we can use a new inference rule:

conditional elim

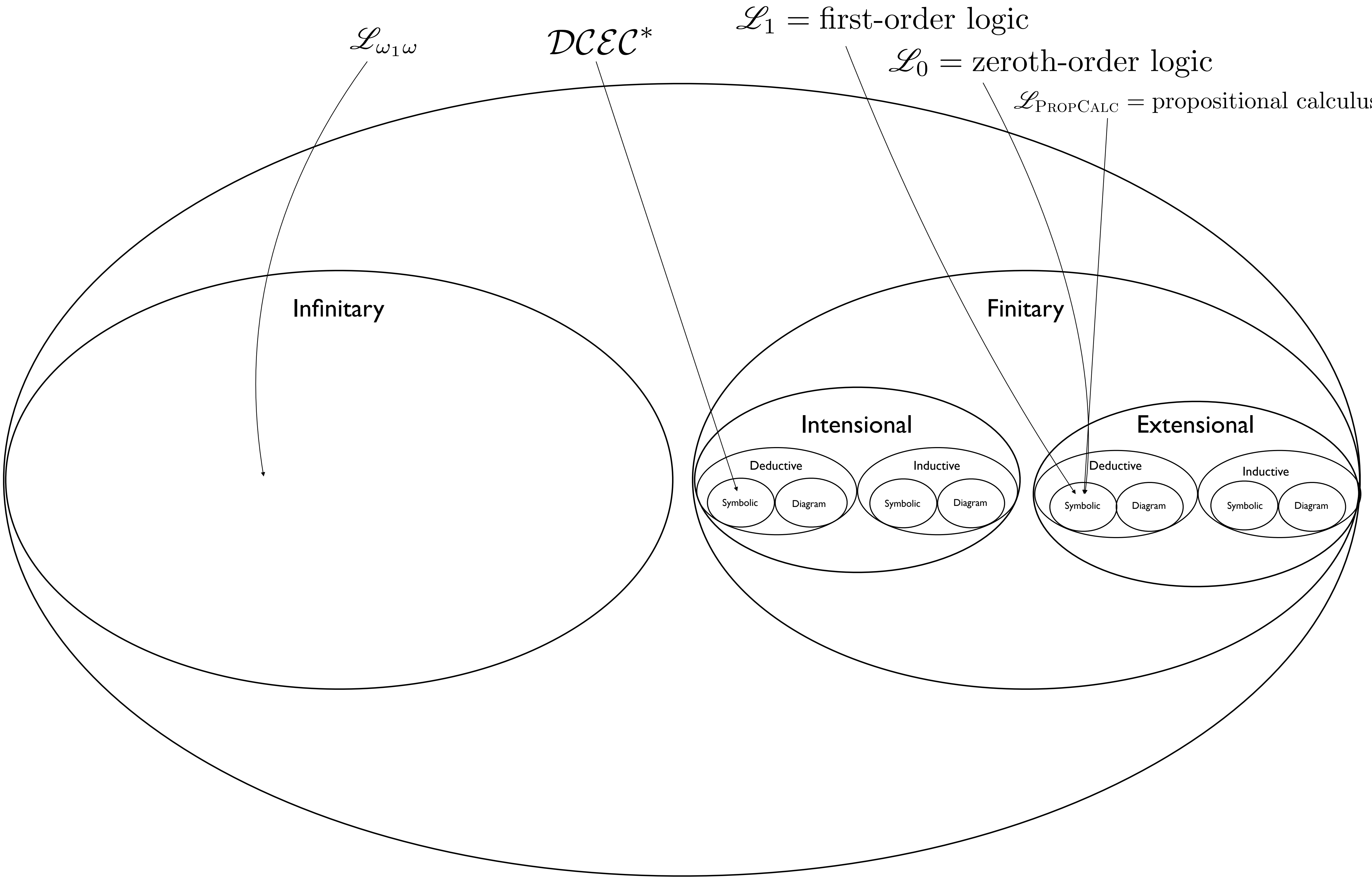
a.k.a., *modus ponens*



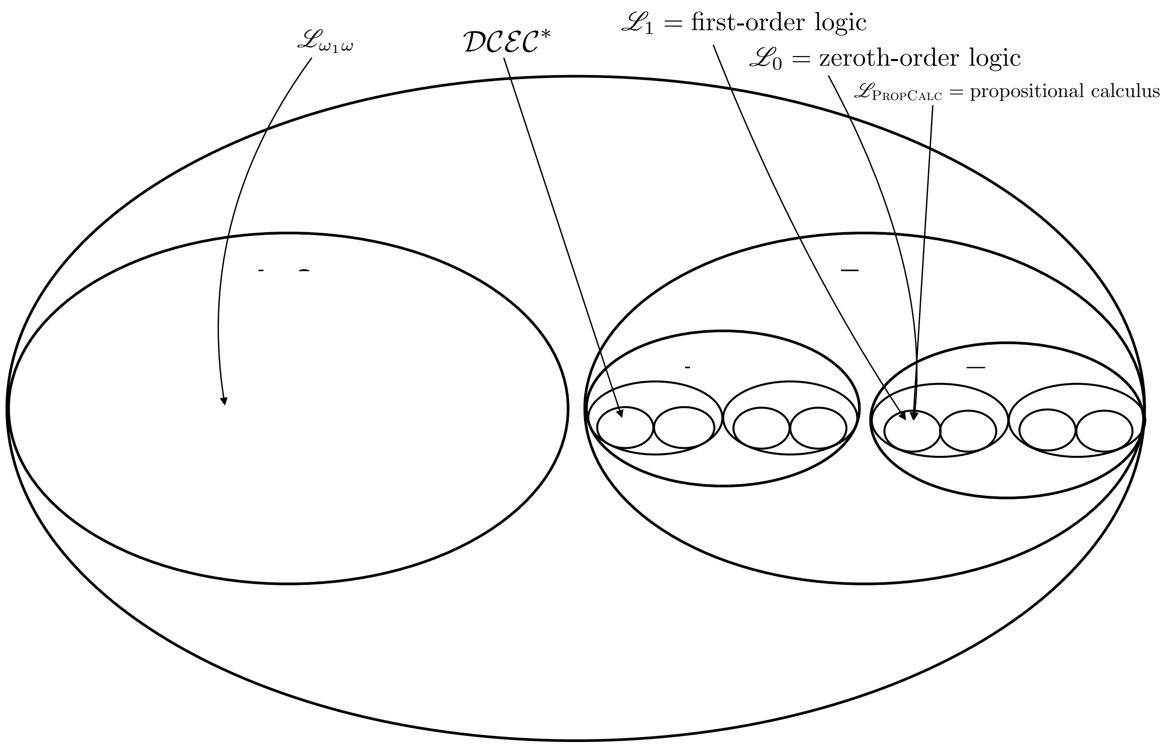
Proof Plan …


Slate - NYS3_in_Slate_sol.slt
Instead of using the PC oracle, we can use a new inference rule:
conditional elim
a.k.a., *modus ponens*
Sub-Proof
Premise1. ¬¬C
{Premise1} Assume ✓
PC ⊢ ✓
1. C
{Premise1}
Premise2. C → A
{Premise2} Assume ✓
PC ⊢ ✓
2. A
{Premise1,Premise2}
Premise3. ¬A ∨ B
{Premise3} Assume ✓
PC ⊢ ✓
3. B
{Premise1,Premise2,Premise3}
Premise4. B → D
{Premise4} Assume ✓
PC ⊢ ✓
4. D
{Premise1,Premise2,Premise3,Premise4}
Premise5. ¬(D ∨ E)
{Premise5} Assume ✓
PC ⊢ ✓
5. ¬D ∧ ¬E
{Premise5}
PC ⊢ ✓
6. ¬D
{Premise5}
PC ⊢ ✓
Option5. ¬C ∧ E ∧ H ∧ ¬A
{Premise1,Premise2,Premise3,Premise4,Premise5}
Option1. ¬C
PC ⊢ ✗
Option2. E
PC ⊢ ✗
Option3. H
PC ⊢ ✗
Option4. ¬A
PC ⊢ ✗

The Universe of Logics

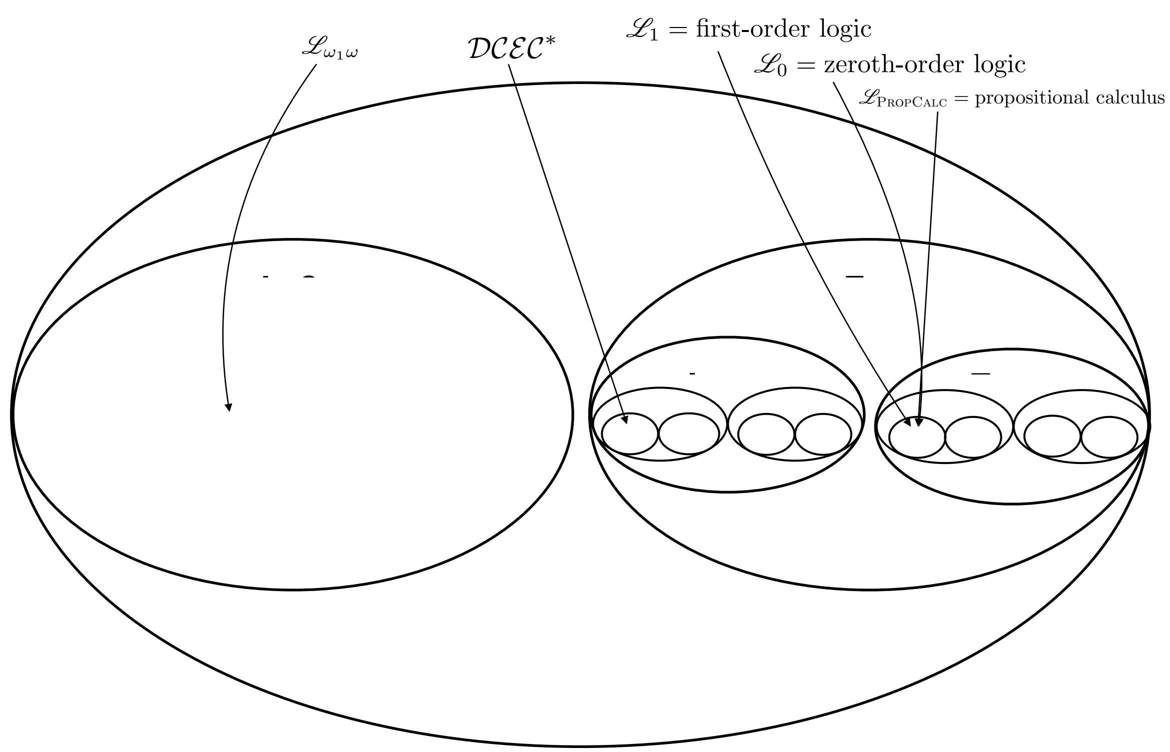


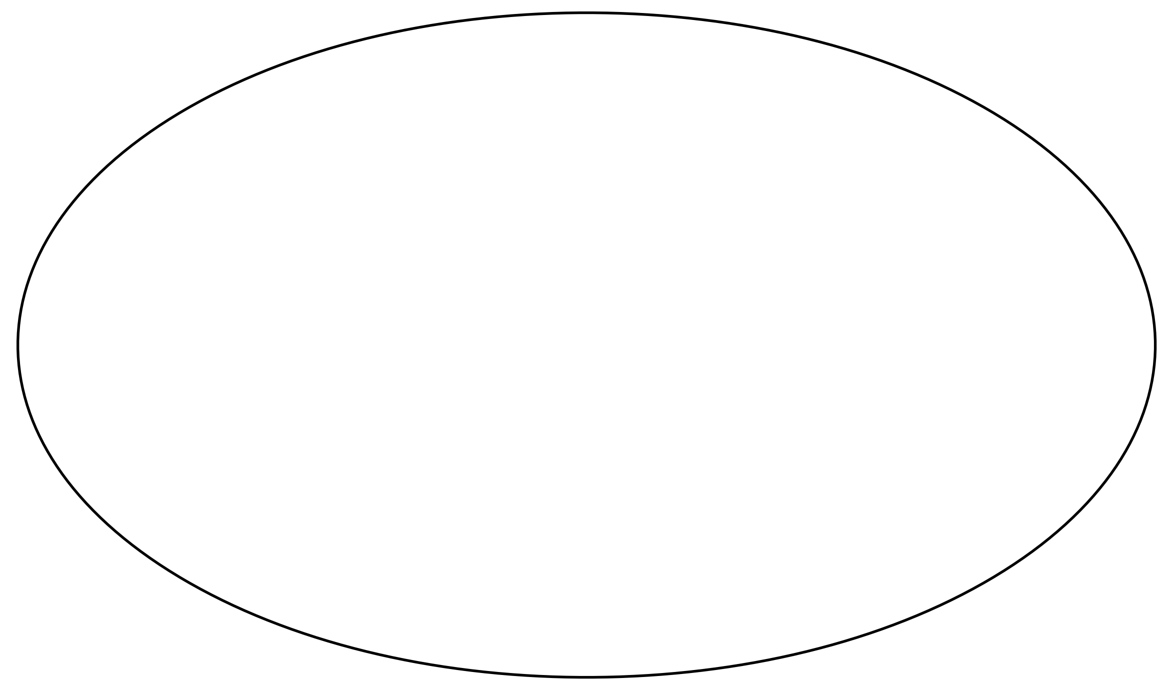
The Universe of Logics



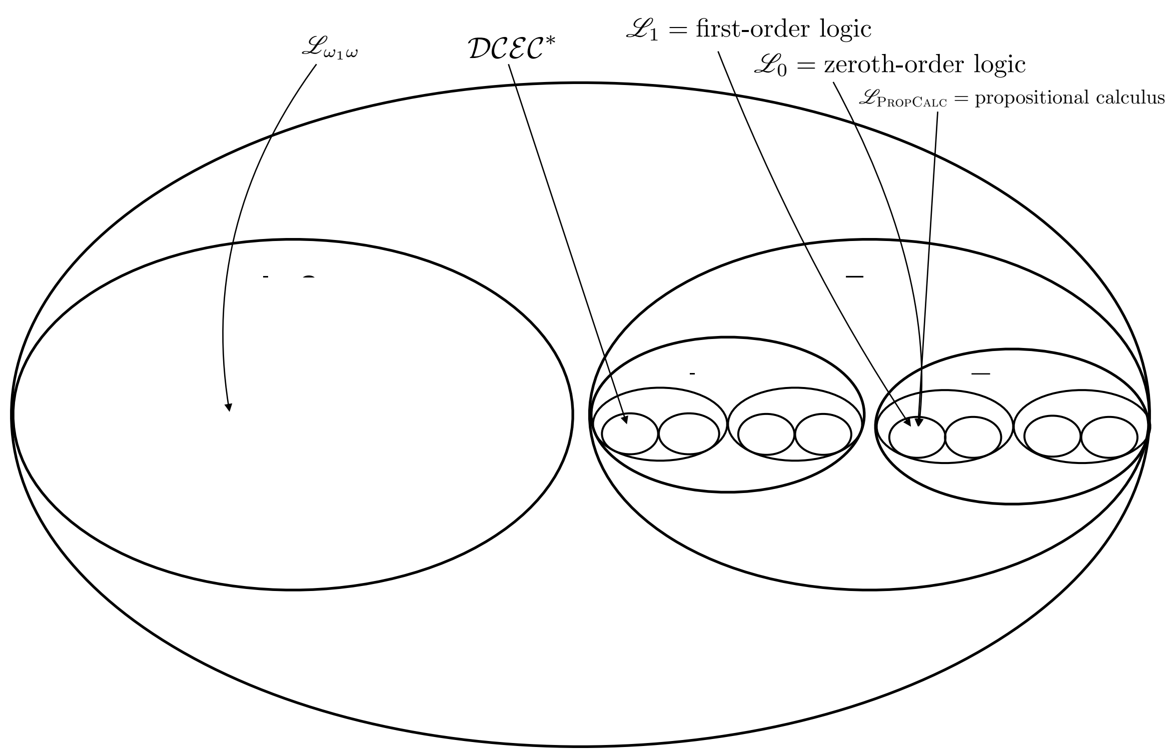
The Universe of Logics

The Physical Universe

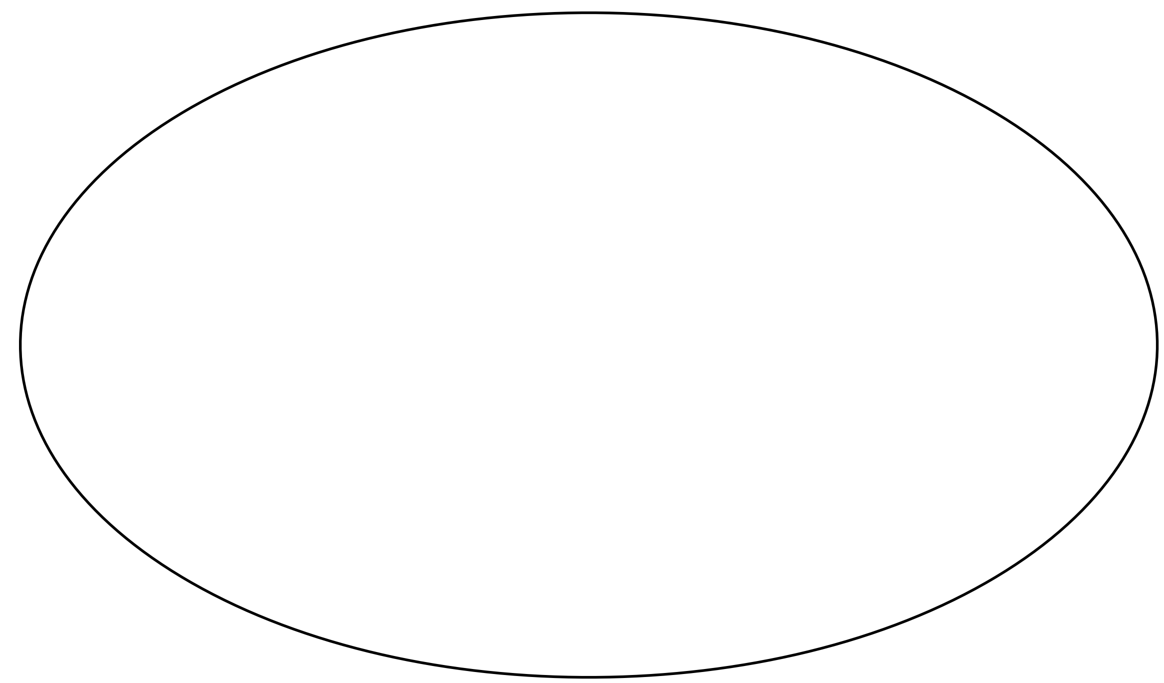




The Universe of Logics

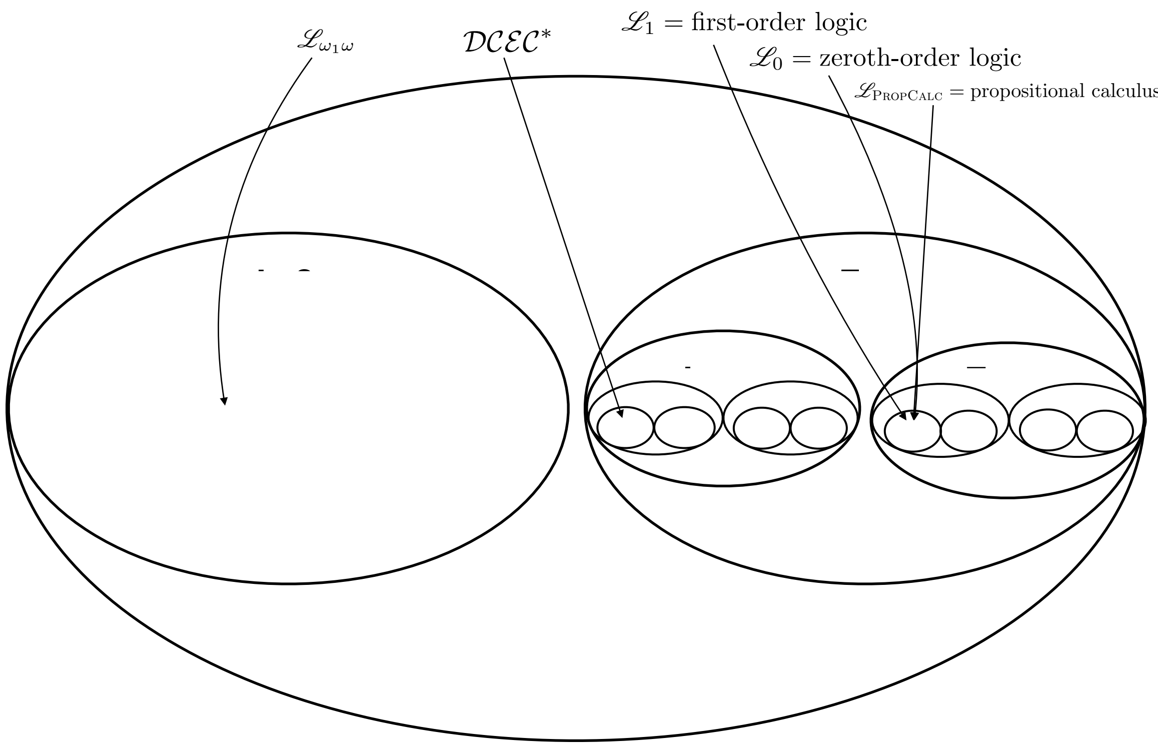


The Physical Universe

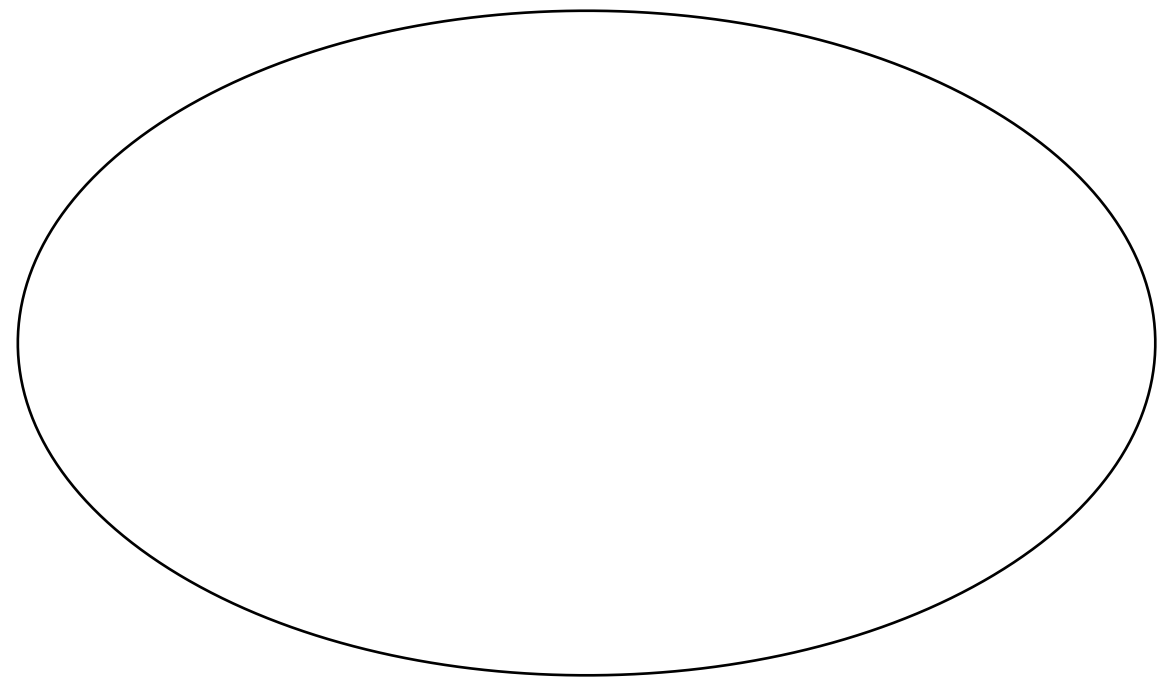


Non-Physical

The Universe of Logics



The Physical Universe

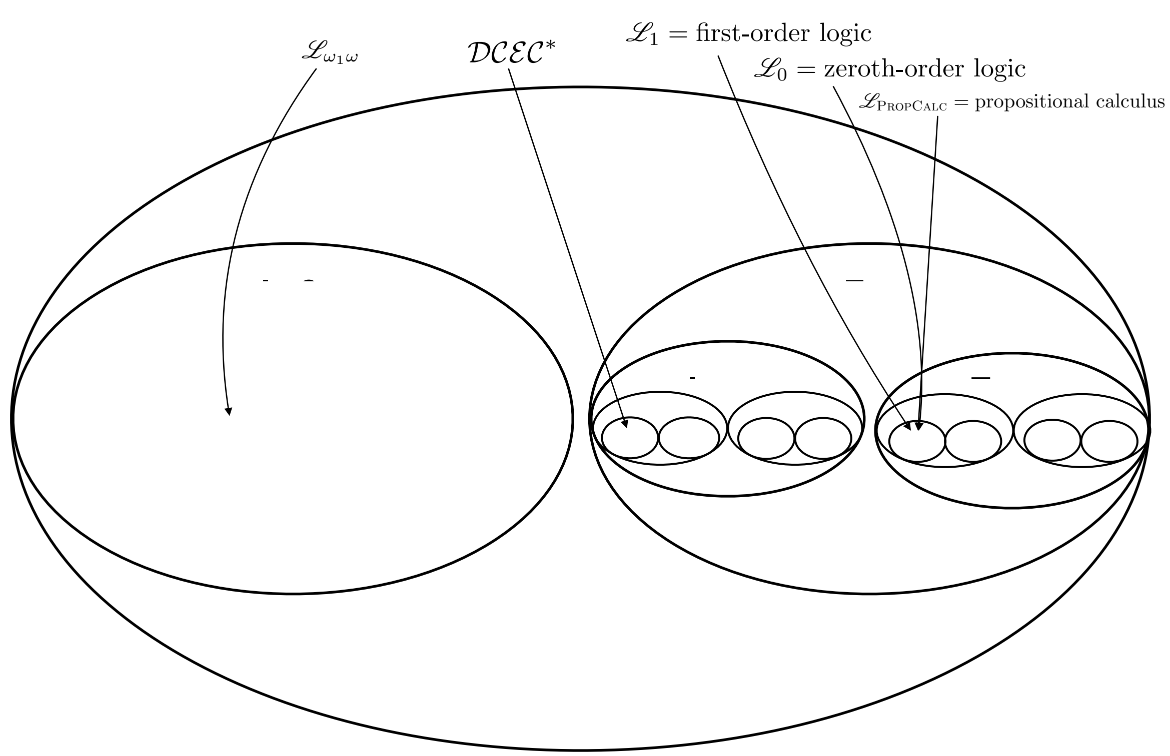


∞

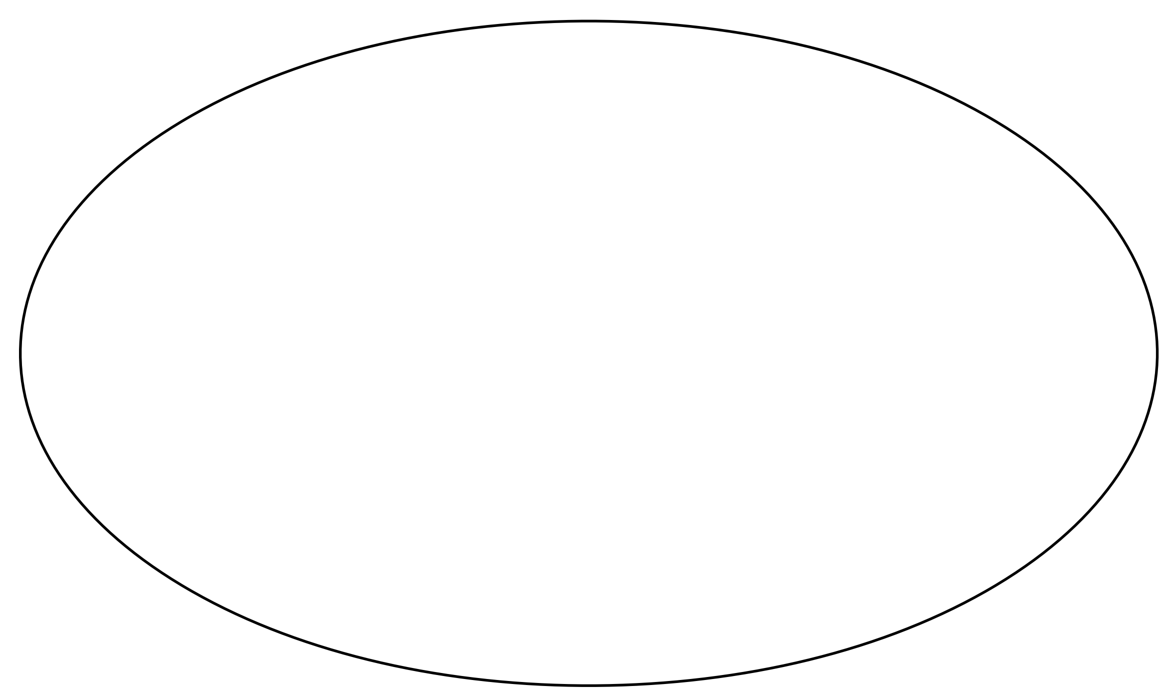
$\mathbb{R}$

$\mathbb{N}$

The Universe of Logics



The Physical Universe



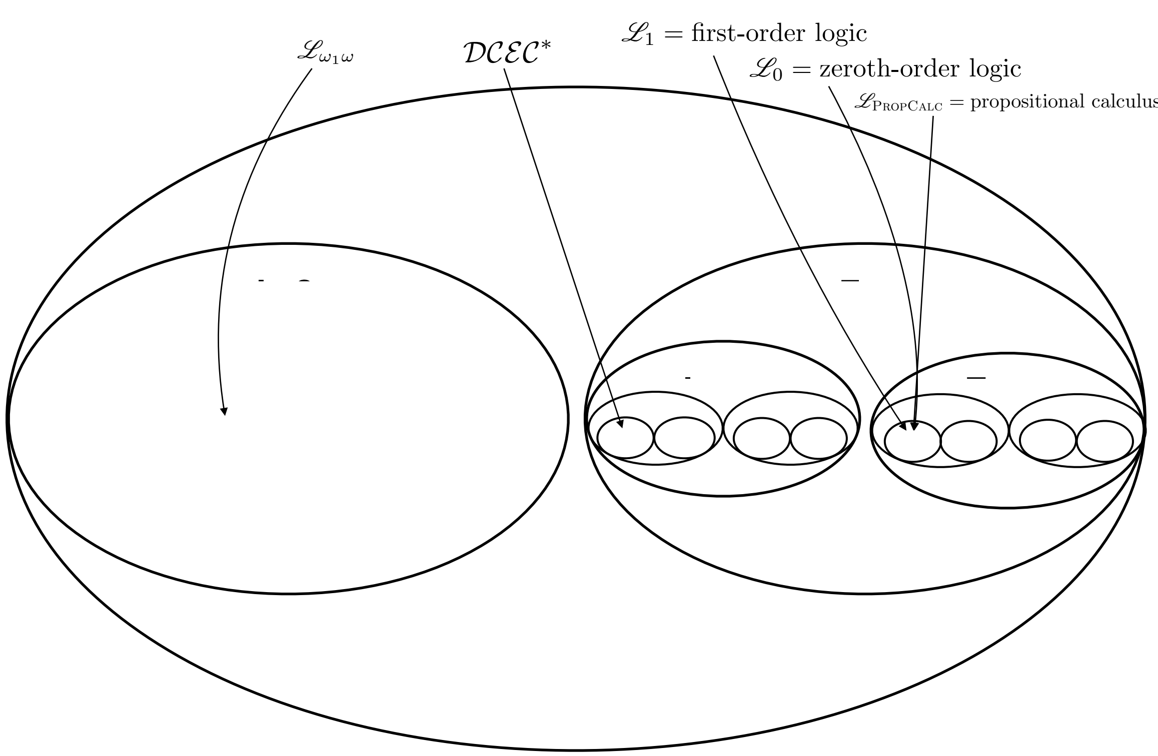
∞

$\mathbb{R}$

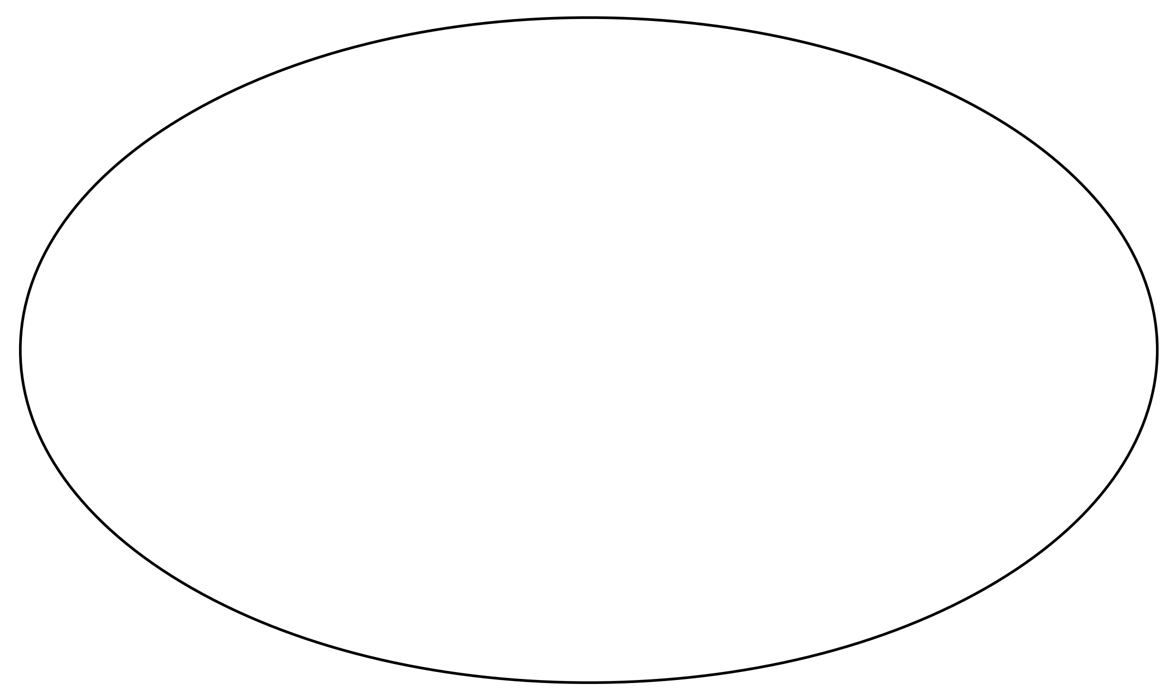
$\mathbb{N}$

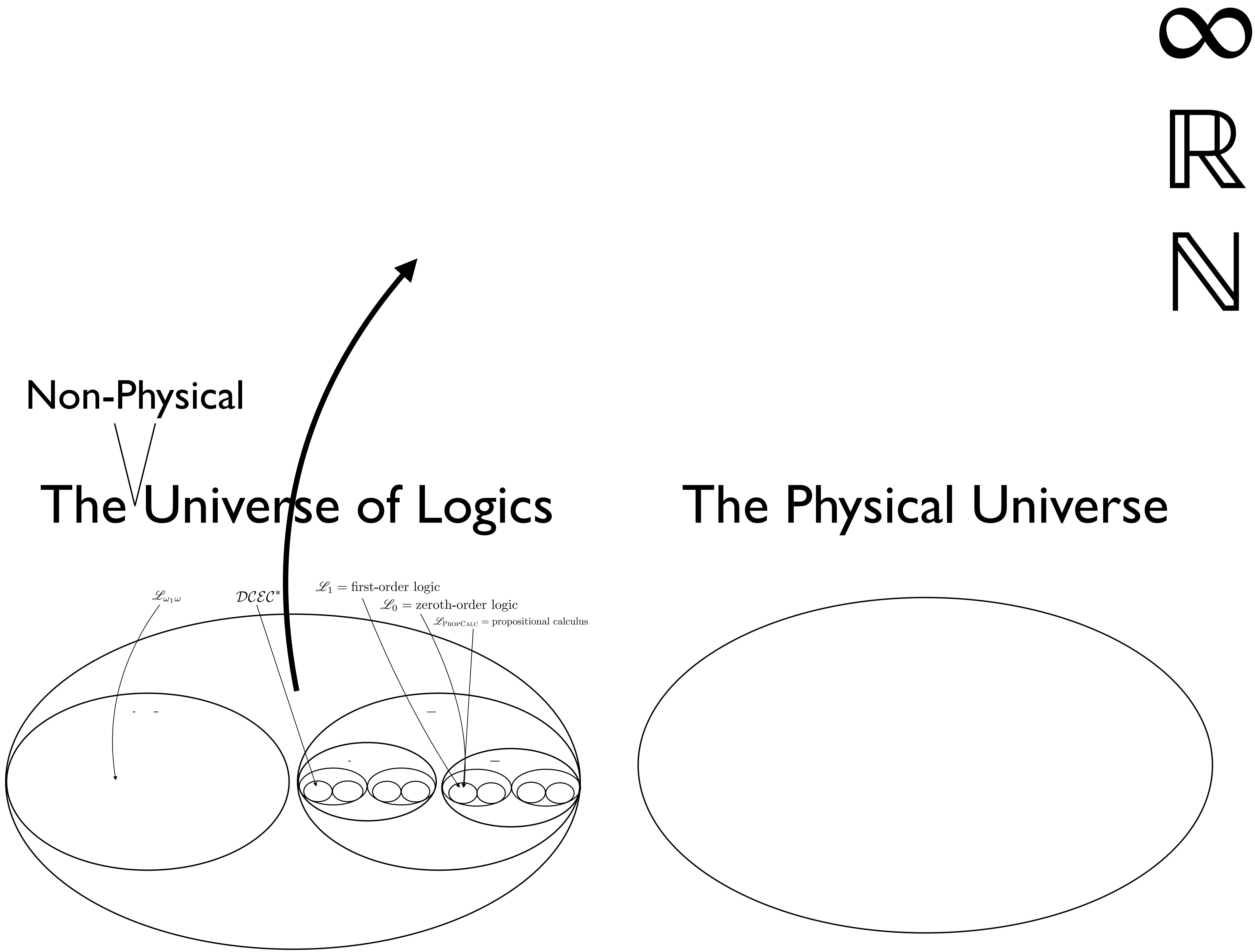
Non-Physical

The Universe of Logics

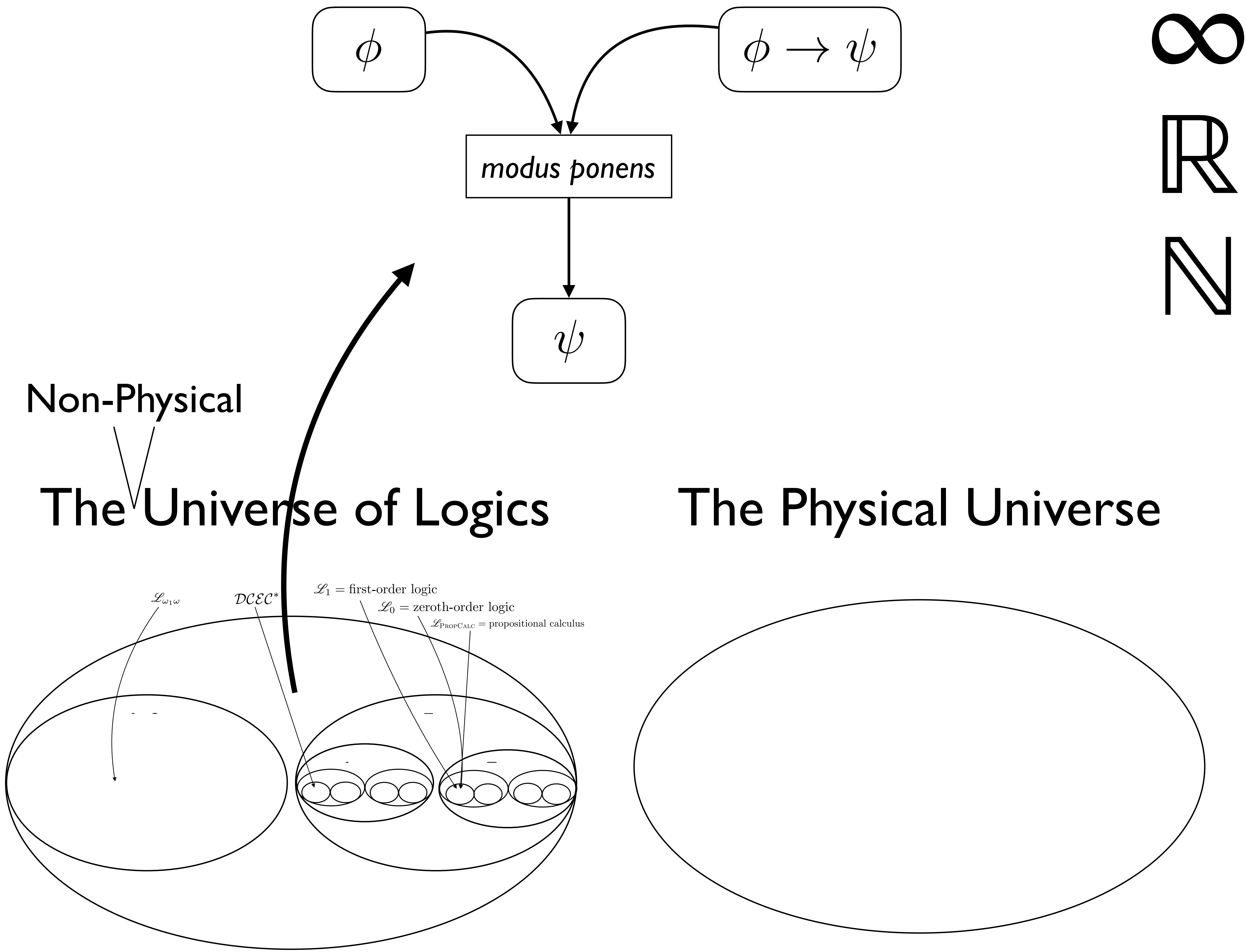


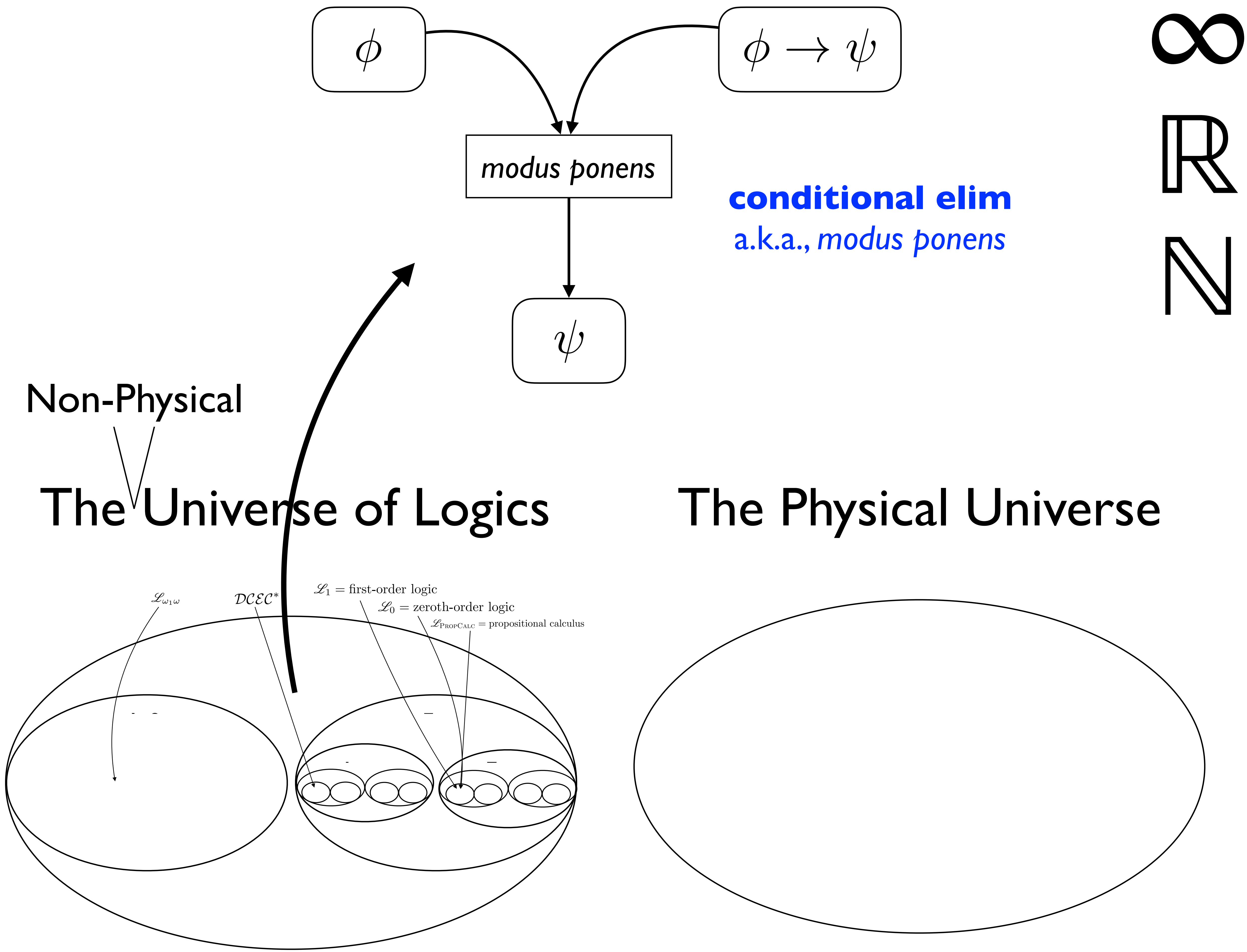
The Physical Universe



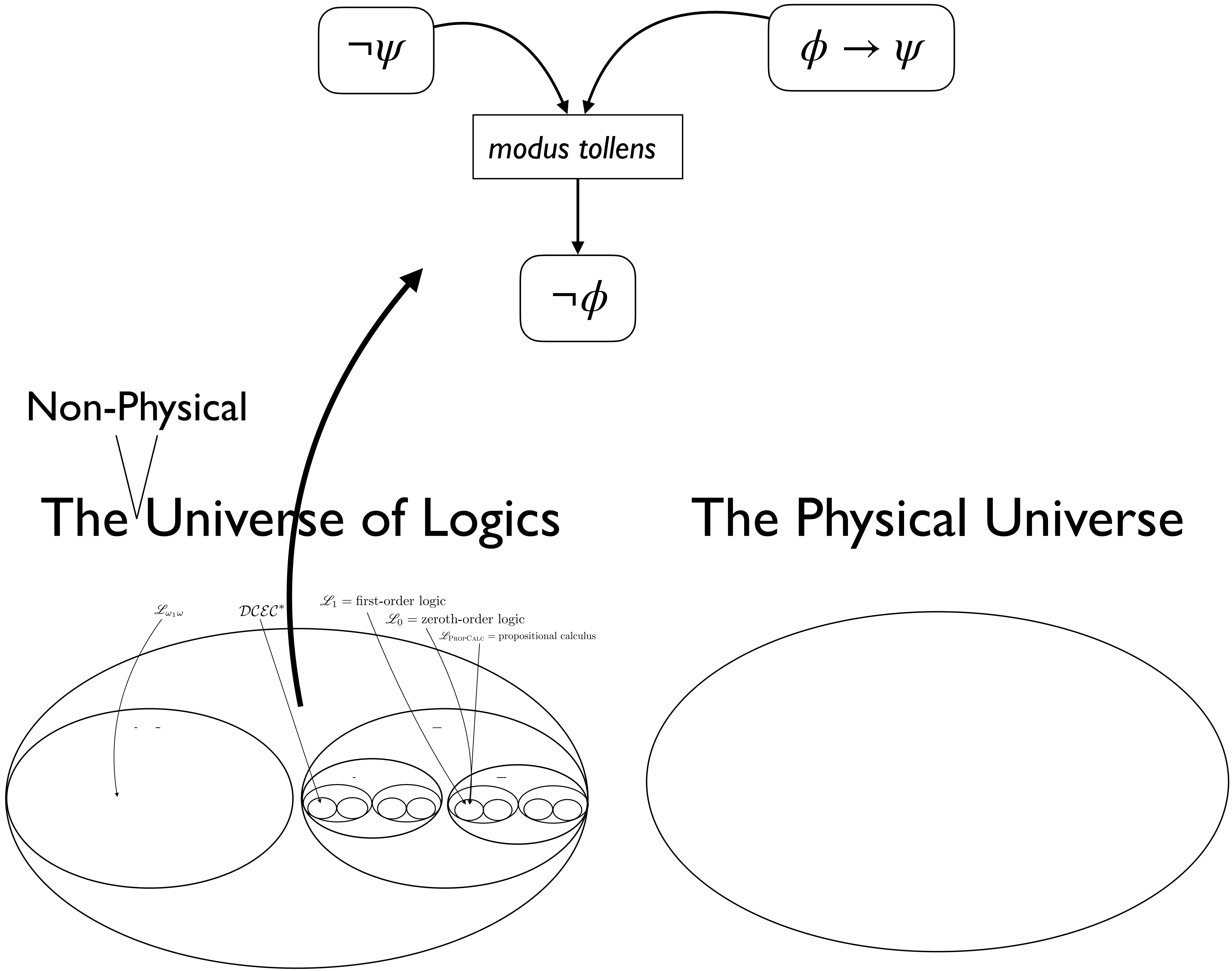


∞
ℝ
ℕ
Non-Physical
The Universe of Logics
The Physical Universe
$\mathscr{L}_{\omega_1\omega}$
$\mathcal{DCEC}^*$
$\mathscr{L}_1$ = first-order logic
$\mathscr{L}_0$ = zeroth-order logic
$\mathscr{L}_{\text{PropCalc}}$ = propositional calculus


ϕ
$\phi \to \psi$
modus ponens
ψ
∞
$\mathbb{R}$
$\mathbb{N}$
Non-Physical
The Universe of Logics
The Physical Universe
$\mathscr{L}_{\omega_1\omega}$
$\mathcal{DCEC}^*$
$\mathscr{L}_1$ = first-order logic
$\mathscr{L}_0$ = zeroth-order logic
$\mathscr{L}_{\text{PropCalc}}$ = propositional calculus



ϕ
$\phi \rightarrow \psi$
modus ponens
ψ
conditional elim
a.k.a., *modus ponens*
∞
$\mathbb{R}$
$\mathbb{N}$
Non-Physical
The Universe of Logics
The Physical Universe
$\mathscr{L}_{\omega_1\omega}$
$\mathcal{DCEC}^*$
$\mathscr{L}_1$ = first-order logic
$\mathscr{L}_0$ = zeroth-order logic
$\mathscr{L}_{\text{PropCalc}}$ = propositional calculus



$\neg\psi$
$\phi \rightarrow \psi$
modus tollens
$\neg\phi$
Non-Physical
The Universe of Logics
The Physical Universe
$\mathscr{L}_{\omega_1\omega}$
$\mathcal{DCEC}^*$
$\mathscr{L}_1$ = first-order logic
$\mathscr{L}_0$ = zeroth-order logic
$\mathscr{L}_{\text{PropCalc}}$ = propositional calculus

Mathematical Objects Are Non-Physical, So We Are Too

Selmer Bringsjord & Naveen Sundar Govindarajulu

`version 0125221730NY`

Mathematical Objects Are Non-Physical, So We Are Too

Selmer Bringsjord & Naveen Sundar Govindarajulu

`version 0125221730NY`

itself is at bottom a simple recursive one. (There are now numerous variants, but we ignore this for efficiency.) The algorithm is to receive an array of ordered objects, for example

$$\left\langle \boxed{5}\ \ \boxed{9}\ \ \boxed{10}\ \ \boxed{7}\ \ \boxed{4}\ \ \boxed{3}\ \ \boxed{11}\ \ \boxed{8}\ \ \boxed{6} \right\rangle,$$

and to then produce as output the sorted version of this input, which in this case is:

$$\left\langle \boxed{3}\ \ \boxed{4}\ \ \boxed{5}\ \ \boxed{6}\ \ \boxed{7}\ \ \boxed{8}\ \ \boxed{9}\ \ \boxed{10}\ \ \boxed{11} \right\rangle.$$

So, what's the algorithm? In order to answer this question, we can't avoid resorting to what we can call *embodiments* or *tokens* of the general and abstract *type* $\mathcal{Q}$.[9] This terminology, and the associated concrete practice, is easy to grasp. For an example, we give one high-level embodiment/token $\hat{\mathcal{Q}}_1$ of $\mathcal{Q}$ that views the algorithm as a three-stage one.[10] Before supplying the example in question, we draw the reader's attention to what we just did with a bit of suggestive notation: We used "hat" $\hat{O}$ to indicate that what is being referred to an an embodiment of the thing O (in this case, of course, an algorithm). Hence, the hat in '$\hat{\mathcal{Q}}_1$' says that we have here an embodiment of the algorithm $\mathcal{Q}$ itself. Very well, and now to the embodiment in question itself:

- I Pick the rightmost element in the array as the *pivot*.
- II Partition the array so that all elements in the array less than the pivot are before it, and all elements greater than the pivot are placed after it.
- III Recursively apply both I and II to the sub-array now before the pivot, as well as to the sub-array now after the pivot.

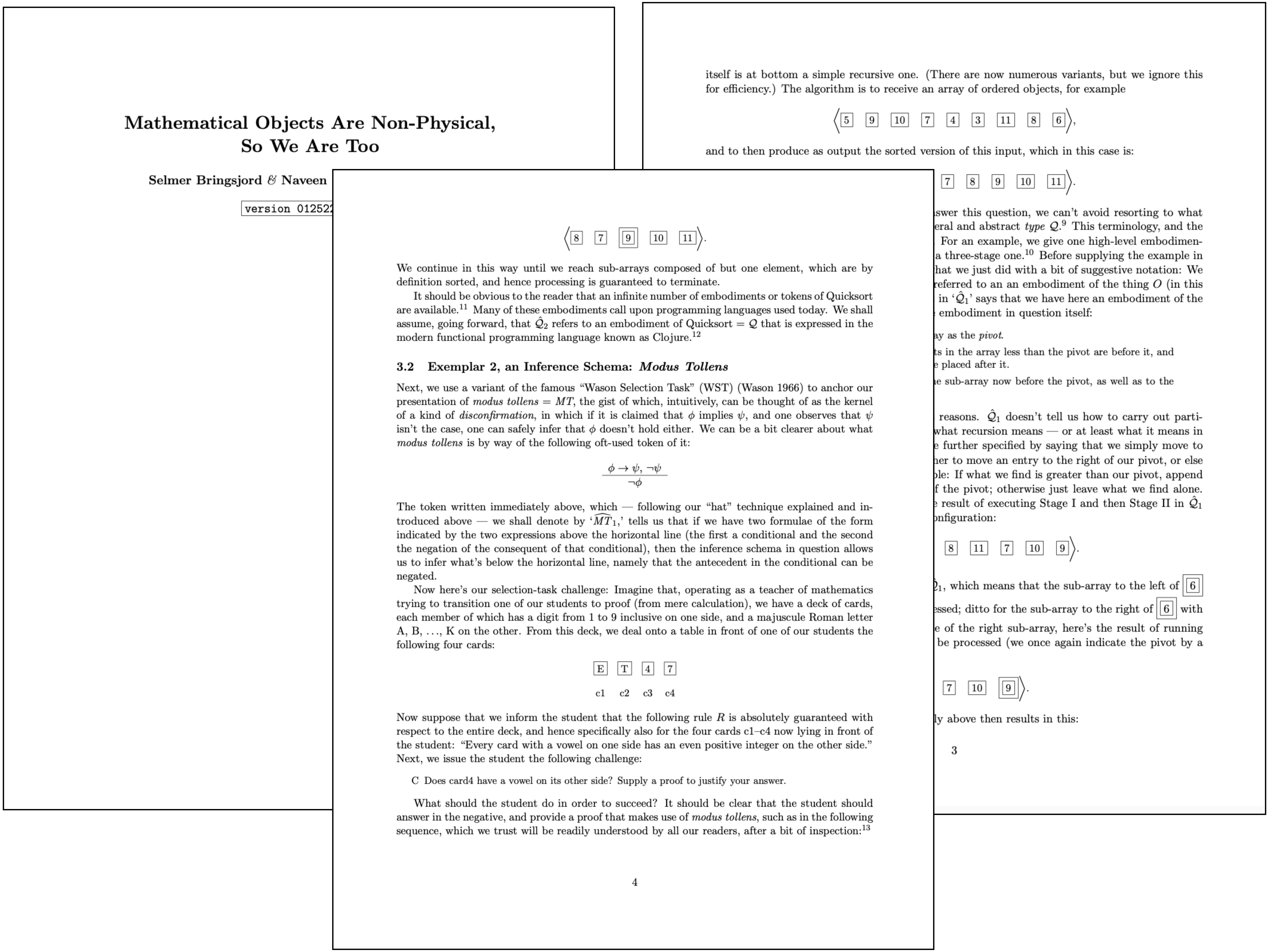
This is said to be 'high-level' for obvious reasons. $\hat{\mathcal{Q}}_1$ doesn't tell us how to carry out partitioning, and it relies on an understanding of what recursion means — or at least what it means in this context. But no worries: Stage II can be further specified by saying that we simply move to the left one entry at a time, and decide whether to move an entry to the right of our pivot, or else leave it where it is. And how to decide? Simple: If what we find is greater than our pivot, append it to whatever sub-sequence is to the right of the pivot; otherwise just leave what we find alone. Using a double-box to indicate our pivot, the result of executing Stage I and then Stage II in $\hat{\mathcal{Q}}_1$ on the initial input array will result in this configuration:

$$\left\langle \boxed{5}\ \ \boxed{4}\ \ \boxed{3}\ \ \boxed{\boxed{6}}\ \ \boxed{8}\ \ \boxed{11}\ \ \boxed{7}\ \ \boxed{10}\ \ \boxed{9} \right\rangle.$$

Now the algorithm calls for Stage III in $\hat{\mathcal{Q}}_1$, which means that the sub-array to the left of $\boxed{\boxed{6}}$ with $\boxed{3}$ as the pivot of this sub-array is processed; ditto for the sub-array to the right of $\boxed{\boxed{6}}$ with $\boxed{9}$ as the pivot of this sub-array. In the case of the right sub-array, here's the result of running Stage I, which is to be passed to Stage II to be processed (we once again indicate the pivot by a double-box):

$$\left\langle \boxed{8}\ \ \boxed{11}\ \ \boxed{7}\ \ \boxed{10}\ \ \boxed{\boxed{9}} \right\rangle.$$

Stage II applied to the input to it immediately above then results in this:



Mathematical Objects Are Non-Physical, So We Are Too

Selmer Bringsjord & Naveen

version 01252

itself is at bottom a simple recursive one. (There are now numerous variants, but we ignore this for efficiency.) The algorithm is to receive an array of ordered objects, for example

⟨ 5 9 10 7 4 3 11 8 6 ⟩,

and to then produce as output the sorted version of this input, which in this case is:

7 8 9 10 11 ⟩.

swer this question, we can't avoid resorting to what
eral and abstract *type* $\mathcal{Q}$.[9] This terminology, and the
For an example, we give one high-level embodimen-
a three-stage one.[10] Before supplying the example in
hat we just did with a bit of suggestive notation: We
eferred to an an embodiment of the thing O (in this
in '$\hat{\mathcal{Q}}_1$' says that we have here an embodiment of the
embodiment in question itself:

ay as the *pivot*.
ts in the array less than the pivot are before it, and
placed after it.
e sub-array now before the pivot, as well as to the

reasons. $\hat{\mathcal{Q}}_1$ doesn't tell us how to carry out parti-
what recursion means — or at least what it means in
e further specified by saying that we simply move to
er to move an entry to the right of our pivot, or else
le: If what we find is greater than our pivot, append
f the pivot; otherwise just leave what we find alone.
e result of executing Stage I and then Stage II in $\hat{\mathcal{Q}}_1$
nfiguration:

8 11 7 10 9 ⟩.

$_1$, which means that the sub-array to the left of 6
ssed; ditto for the sub-array to the right of 6 with
e of the right sub-array, here's the result of running
be processed (we once again indicate the pivot by a

7 10 9 ⟩.

ly above then results in this:

3

⟨ 8 7 9 10 11 ⟩.

We continue in this way until we reach sub-arrays composed of but one element, which are by definition sorted, and hence processing is guaranteed to terminate.

It should be obvious to the reader that an infinite number of embodiments or tokens of Quicksort are available.[11] Many of these embodiments call upon programming languages used today. We shall assume, going forward, that $\hat{\mathcal{Q}}_2$ refers to an embodiment of Quicksort = $\mathcal{Q}$ that is expressed in the modern functional programming language known as Clojure.[12]

3.2 Exemplar 2, an Inference Schema: *Modus Tollens*

Next, we use a variant of the famous "Wason Selection Task" (WST) (Wason 1966) to anchor our presentation of *modus tollens* = MT, the gist of which, intuitively, can be thought of as the kernel of a kind of *disconfirmation*, in which if it is claimed that ϕ implies ψ, and one observes that ψ isn't the case, one can safely infer that ϕ doesn't hold either. We can be a bit clearer about what *modus tollens* is by way of the following oft-used token of it:

$$\frac{\phi \rightarrow \psi, \neg\psi}{\neg\phi}$$

The token written immediately above, which — following our "hat" technique explained and introduced above — we shall denote by '$\widehat{MT}_1$,' tells us that if we have two formulae of the form indicated by the two expressions above the horizontal line (the first a conditional and the second the negation of the consequent of that conditional), then the inference schema in question allows us to infer what's below the horizontal line, namely that the antecedent in the conditional can be negated.

Now here's our selection-task challenge: Imagine that, operating as a teacher of mathematics trying to transition one of our students to proof (from mere calculation), we have a deck of cards, each member of which has a digit from 1 to 9 inclusive on one side, and a majuscule Roman letter A, B, ..., K on the other. From this deck, we deal onto a table in front of one of our students the following four cards:

E	T	4	7
c1	c2	c3	c4

Now suppose that we inform the student that the following rule R is absolutely guaranteed with respect to the entire deck, and hence specifically also for the four cards c1–c4 now lying in front of the student: "Every card with a vowel on one side has an even positive integer on the other side." Next, we issue the student the following challenge:

C Does card4 have a vowel on its other side? Supply a proof to justify your answer.

What should the student do in order to succeed? It should be clear that the student should answer in the negative, and provide a proof that makes use of *modus tollens*, such as in the following sequence, which we trust will be readily understood by all our readers, after a bit of inspection:[13]

4

Mathematical Objects Are Non-Physical, So We Are Too

Selmer Bringsjord & Naveen

version 01252

itself is at bottom a simple recursive one. (There are now numerous variants, but we ignore this for efficiency.) The algorithm is to receive an array of ordered objects, for example

$$\langle \boxed{5}\ \boxed{9}\ \boxed{10}\ \boxed{7}\ \boxed{4}\ \boxed{3}\ \boxed{11}\ \boxed{8}\ \boxed{6} \rangle,$$

and to then produce as output the sorted version of this input, which in this case is:

$$\boxed{7}\ \boxed{8}\ \boxed{9}\ \boxed{10}\ \boxed{11} \rangle.$$

swer this question, we can't avoid resorting to what
eral and abstract *type* $\mathcal{Q}$.[9] This terminology, and the
For an example, we give one high-level embodimen-
a three-stage one.[10] Before supplying the example in
hat we just did with a bit of suggestive notation: We
eferred to an an embodiment of the thing O (in this
in '$\hat{\mathcal{Q}}_1$' says that we have here an embodiment of the
embodiment in question itself:

ay as the *pivot*.
ts in the array less than the pivot are before it, and
e placed after it.
e sub-array now before the pivot, as well as to the

reasons. $\hat{\mathcal{Q}}_1$ doesn't tell us how to carry out parti-
what recursion means — or at least what it means in
e further specified by saying that we simply move to
her to move an entry to the right of our pivot, or else
ble: If what we find is greater than our pivot, append
f the pivot; otherwise just leave what we find alone.
e result of executing Stage I and then Stage II in $\hat{\mathcal{Q}}_1$
onfiguration:

$$\boxed{8}\ \boxed{11}\ \boxed{7}\ \boxed{10}\ \boxed{9} \rangle.$$

$_1$, which means that the sub-array to the left of $\boxed{6}$
essed; ditto for the sub-array to the right of $\boxed{6}$ with
e of the right sub-array, here's the result of running
be processed (we once again indicate the pivot by a

$$\boxed{7}\ \boxed{10}\ \boxed{\boxed{9}} \rangle.$$

ly above then results in this:

$$\langle \boxed{8}\ \boxed{7}\ \boxed{\boxed{9}}\ \boxed{10}\ \boxed{11} \rangle.$$

We continue in this way until we reach sub-arrays composed of but one element, which are by definition sorted, and hence processing is guaranteed to terminate.

It should be obvious to the reader that an infinite number of embodiments or tokens of Quicksort are available.[11] Many of these embodiments call upon programming languages used today. We shall assume, going forward, that $\hat{\mathcal{Q}}_2$ refers to an embodiment of Quicksort $= \mathcal{Q}$ that is expressed in the modern functional programming language known as Clojure.[12]

3.2 Exemplar 2, an Inference Schema: *Modus Tollens*

Next, we use a variant of the famous "Wason Selection Task" (WST) (Wason 1966) to anchor our presentation of *modus tollens* $= MT$, the gist of which, intuitively, can be thought of as the kernel of a kind of *disconfirmation*, in which if it is claimed that ϕ implies ψ, and one observes that ψ isn't the case, one can safely infer that ϕ doesn't hold either. We can be a bit clearer about what *modus tollens* is by way of the following oft-used token of it:

$$\frac{\phi \rightarrow \psi,\ \neg\psi}{\neg\phi}$$

The token written immediately above, which — following our "hat" technique explained and introduced above — we shall denote by '$\widehat{MT}_1$,' tells us that if we have two formulae of the form indicated by the two expressions above the horizontal line (the first a conditional and the second the negation of the consequent of that conditional), then the inference schema in question allows us to infer what's below the horizontal line, namely that the antecedent in the conditional can be negated.

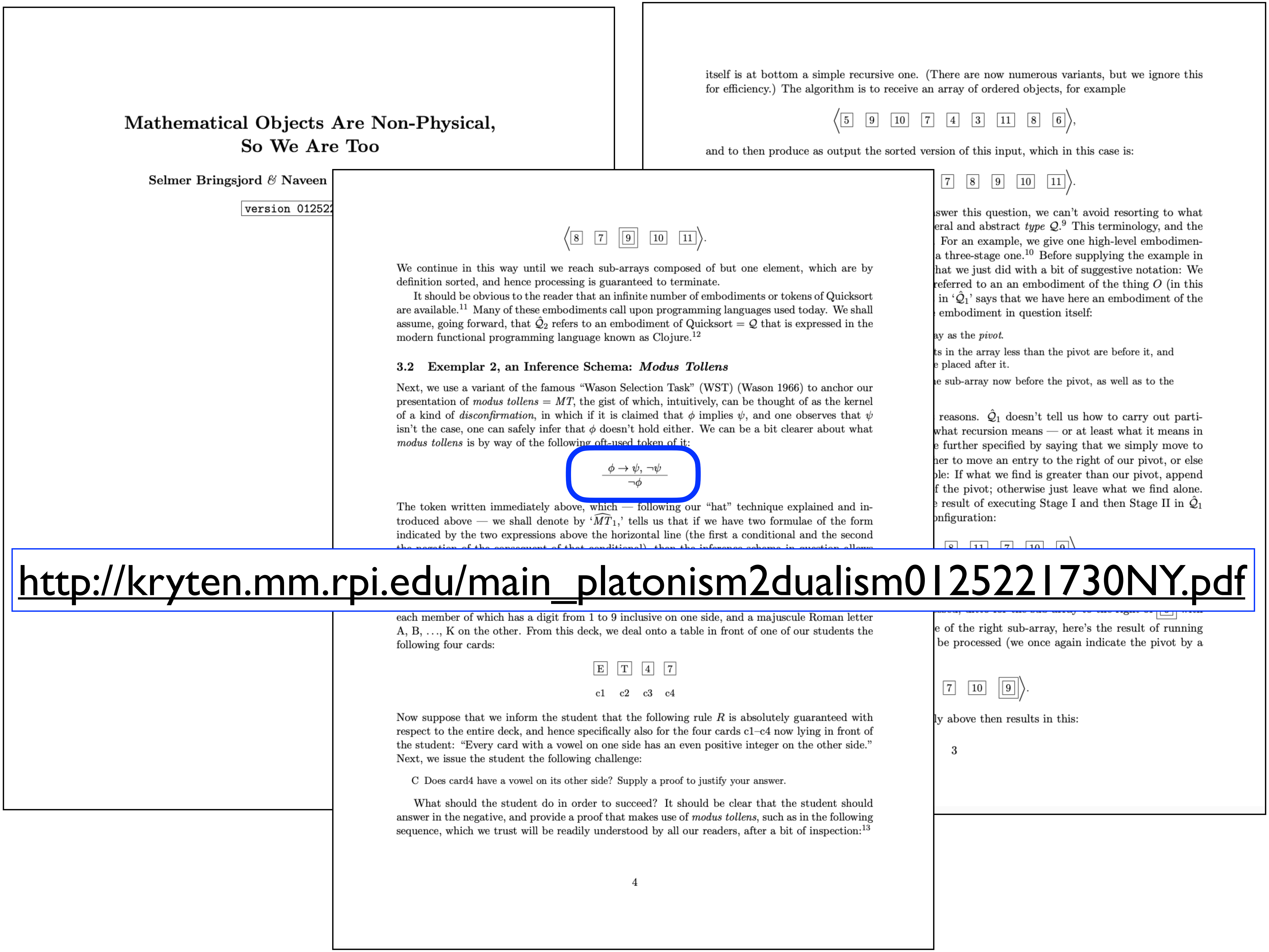
Now here's our selection-task challenge: Imagine that, operating as a teacher of mathematics trying to transition one of our students to proof (from mere calculation), we have a deck of cards, each member of which has a digit from 1 to 9 inclusive on one side, and a majuscule Roman letter A, B, ..., K on the other. From this deck, we deal onto a table in front of one of our students the following four cards:

E	T	4	7
c1	c2	c3	c4

Now suppose that we inform the student that the following rule R is absolutely guaranteed with respect to the entire deck, and hence specifically also for the four cards c1–c4 now lying in front of the student: "Every card with a vowel on one side has an even positive integer on the other side." Next, we issue the student the following challenge:

C Does card4 have a vowel on its other side? Supply a proof to justify your answer.

What should the student do in order to succeed? It should be clear that the student should answer in the negative, and provide a proof that makes use of *modus tollens*, such as in the following sequence, which we trust will be readily understood by all our readers, after a bit of inspection:[13]



Mathematical Objects Are Non-Physical,
So We Are Too

Selmer Bringsjord & Naveen

version 01252

itself is at bottom a simple recursive one. (There are now numerous variants, but we ignore this for efficiency.) The algorithm is to receive an array of ordered objects, for example

⟨5 9 10 7 4 3 11 8 6⟩,

and to then produce as output the sorted version of this input, which in this case is:

⟨8 7 9 10 11⟩.

We continue in this way until we reach sub-arrays composed of but one element, which are by definition sorted, and hence processing is guaranteed to terminate.

It should be obvious to the reader that an infinite number of embodiments or tokens of Quicksort are available.[11] Many of these embodiments call upon programming languages used today. We shall assume, going forward, that $\hat{\mathcal{Q}}_2$ refers to an embodiment of Quicksort $= \mathcal{Q}$ that is expressed in the modern functional programming language known as Clojure.[12]

3.2 Exemplar 2, an Inference Schema: *Modus Tollens*

Next, we use a variant of the famous "Wason Selection Task" (WST) (Wason 1966) to anchor our presentation of *modus tollens* $= MT$, the gist of which, intuitively, can be thought of as the kernel of a kind of *disconfirmation*, in which if it is claimed that ϕ implies ψ, and one observes that ψ isn't the case, one can safely infer that ϕ doesn't hold either. We can be a bit clearer about what *modus tollens* is by way of the following oft-used token of it:

$$\frac{\phi \rightarrow \psi, \neg\psi}{\neg\phi}$$

The token written immediately above, which — following our "hat" technique explained and introduced above — we shall denote by '$\widehat{MT}_1$,' tells us that if we have two formulae of the form indicated by the two expressions above the horizontal line (the first a conditional and the second

http://kryten.mm.rpi.edu/main_platonism2dualism0125221730NY.pdf

each member of which has a digit from 1 to 9 inclusive on one side, and a majuscule Roman letter A, B, …, K on the other. From this deck, we deal onto a table in front of one of our students the following four cards:

E T 4 7

c1 c2 c3 c4

Now suppose that we inform the student that the following rule R is absolutely guaranteed with respect to the entire deck, and hence specifically also for the four cards c1–c4 now lying in front of the student: "Every card with a vowel on one side has an even positive integer on the other side." Next, we issue the student the following challenge:

C Does card4 have a vowel on its other side? Supply a proof to justify your answer.

What should the student do in order to succeed? It should be clear that the student should answer in the negative, and provide a proof that makes use of *modus tollens*, such as in the following sequence, which we trust will be readily understood by all our readers, after a bit of inspection:[13]

4

Next problem (King-Ace) …

King-Ace 2

Suppose that the following premise is true:

If there is a king in the hand, then there is an ace in the hand; or if there isn't a king in the hand, then there is an ace; but not both of these if-then statements are true.

What can you infer from this premise?

King-Ace 2

Suppose that the following premise is true:

If there is a king in the hand, then there is an ace in the hand; or if there isn't a king in the hand, then there is an ace; but not both of these if-then statements are true.

What can you infer from this premise?

There is an ace in the hand.

King-Ace 2

Suppose that the following premise is true:

If there is a king in the hand, then there is an ace in the hand; or if there isn't a king in the hand, then there is an ace; but not both of these if-then statements are true.

What can you infer from this premise?

~~There is an ace in the hand.~~

King-Ace 2

Suppose that the following premise is true:

If there is a king in the hand, then there is an ace in the hand; or if there isn't a king in the hand, then there is an ace; but not both of these if-then statements are true.

What can you infer from this premise?

NO! ~~There is an ace in the hand.~~

King-Ace 2

Suppose that the following premise is true:

If there is a king in the hand, then there is an ace in the hand; or if there isn't a king in the hand, then there is an ace; but not both of these if-then statements are true.

What can you infer from this premise?

NO! ~~There is an ace in the hand.~~ NO!

King-Ace 2

Suppose that the following premise is true:

If there is a king in the hand, then there is an ace in the hand; or if there isn't a king in the hand, then there is an ace; but not both of these if-then statements are true.

What can you infer from this premise?

NO! ~~There is an ace in the hand.~~ NO!

In fact, what you *can* infer is that there *isn't* an ace in the hand!

King-Ace Solved

Proposition: There is *not* an ace in the hand.

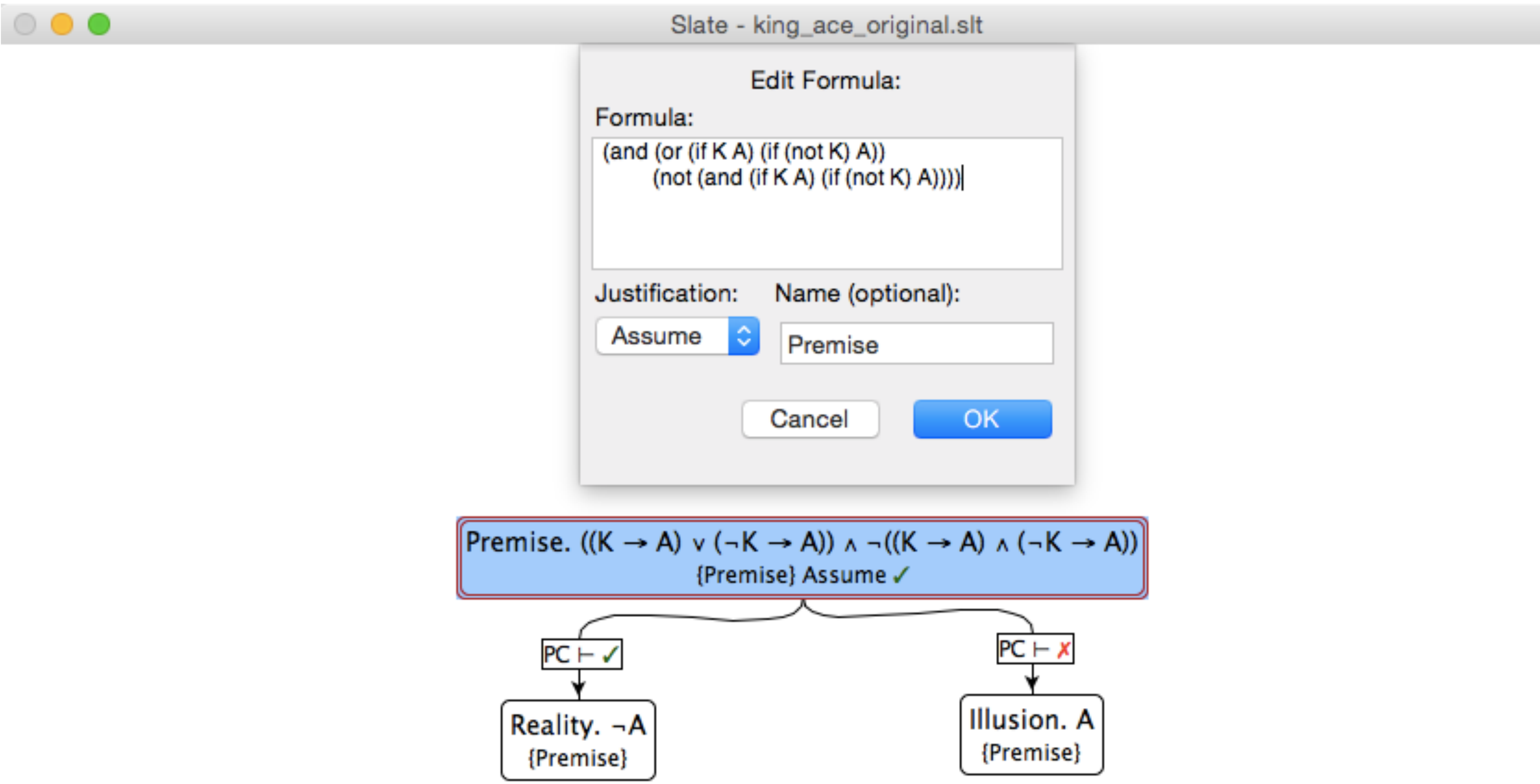
Proof: We know that at least one of the if-thens (i.e., at least one of the **conditionals**) is false. So we have two cases to consider, viz., that *K* => *A* is false, and that ¬*K* => *A* is false. Take first the first case; accordingly, suppose that *K* => *A* is false. Then it follows that *K* is true (since when a conditional is false, its antecedent holds but its consequent doesn't), and *A* is false. Now consider the second case, which consists in ¬*K* => *A* being false. Here, in a direct parallel, we know ¬*K* and, once again, ¬*A*. In both of our two cases, which are exhaustive, there is no ace in the hand. The proposition is established. **QED**

King-Ace Solved

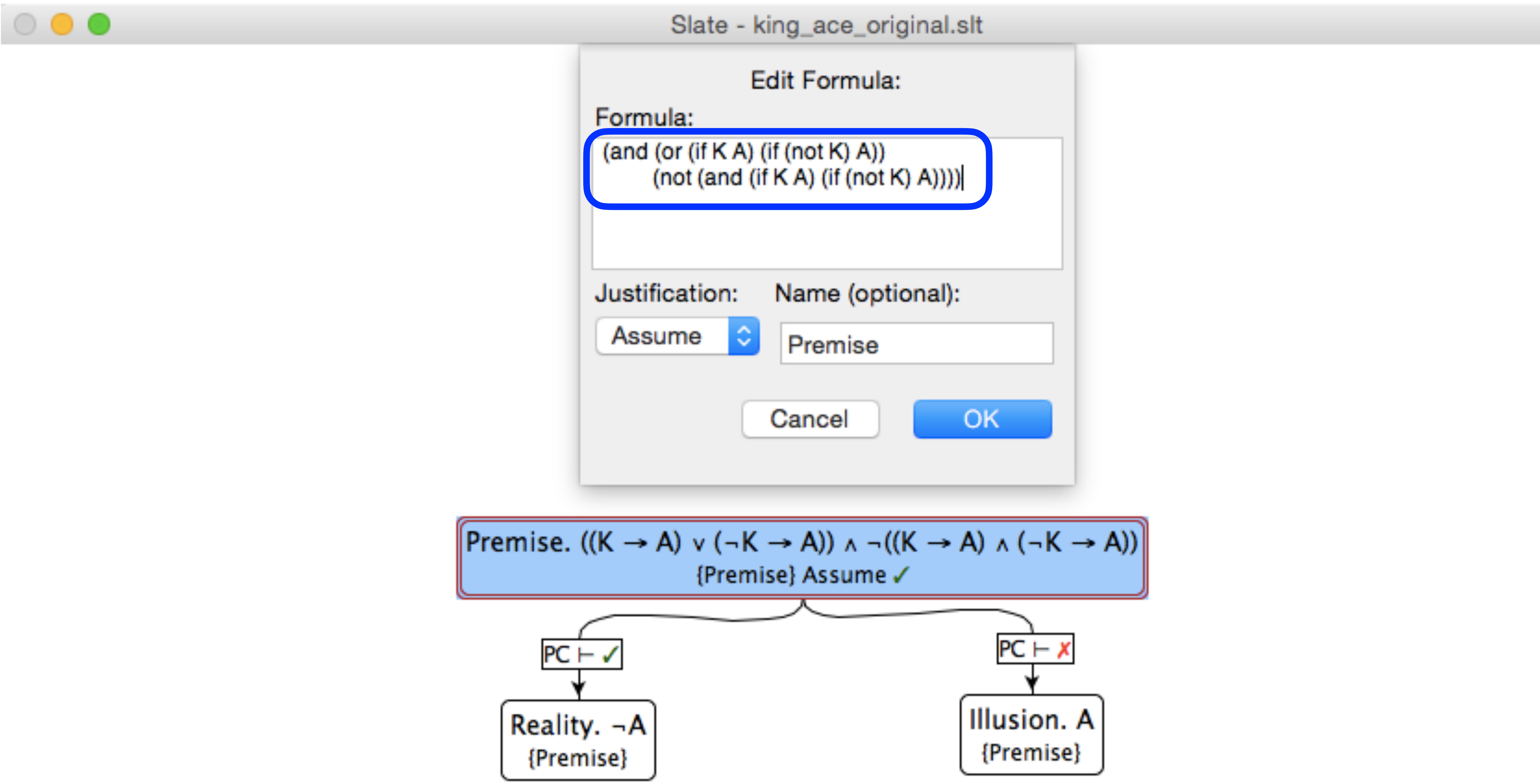
Proposition: There is *not* an ace in the hand.

Proof: We know that at least one of the if-thens (i.e., at least one of the **conditionals**) is false. So we have two cases to consider, viz., that *K* => *A* is false, and that ¬*K* => *A* is false. Take first the first case; accordingly, suppose that *K* => *A* is false. Then it follows that *K* is true (since when a conditional is false, its antecedent holds but its consequent doesn't), and *A* is false. Now consider the second case, which consists in ¬*K* => *A* being false. Here, in a direct parallel, we know ¬*K* and, once again, ¬*A*. In both of our two cases, which are exhaustive, there is no ace in the hand. The proposition is established. **QED**

Study the S-expression



Study the S-expression



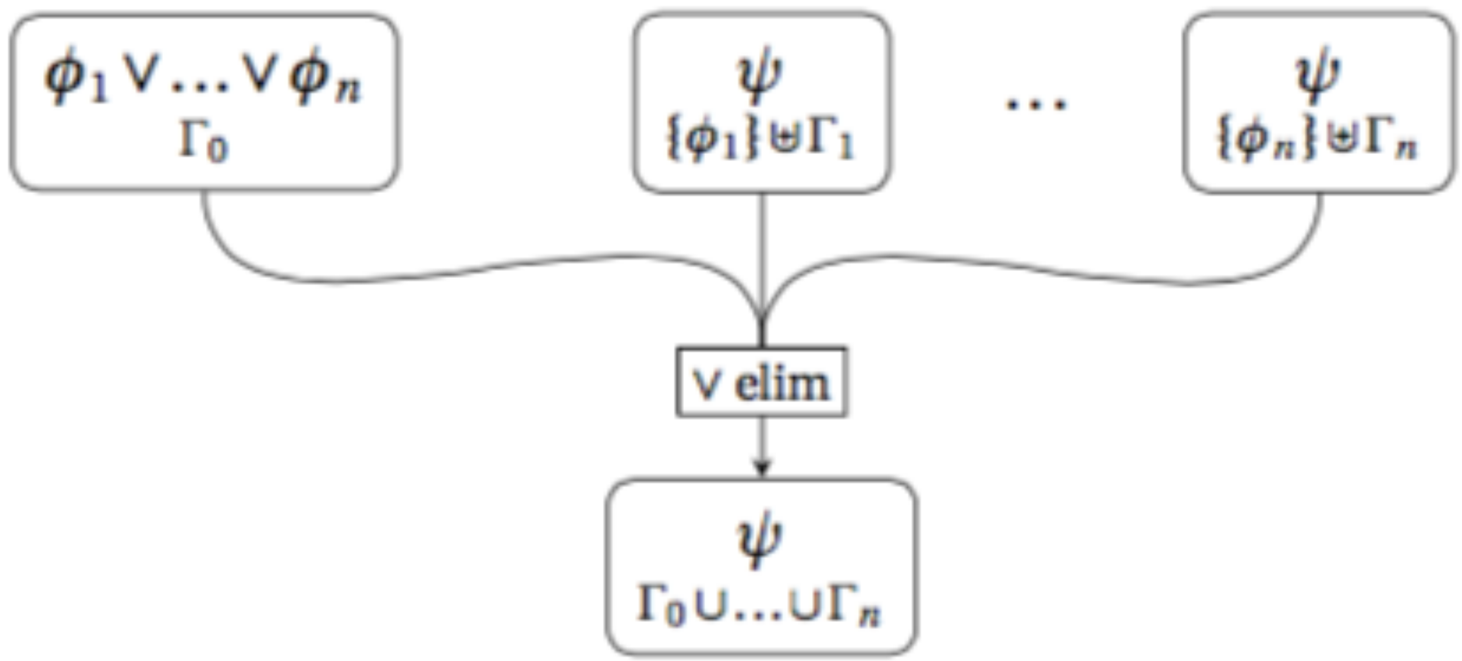
We need another rule of inference to crack this problem … …

We need another rule of inference to crack this problem … …

`disjunction elimination`

From ~ p. 54 in LAMA-BDLA

from each ϕ_i, then we may conclude ψ. That is, if we can, for each ϕ_i, assume ϕ_i and show that ψ follows, then we may conclude ψ from the disjunction $\phi_1 \vee \ldots \vee \phi_n$ and the derivations of ψ. There is one more subtle point, however. In the days-of-the-week example above, the conclusion that Susan has class on a weekday should not be in the scope of both the assumptions that she has class on Monday and that she has class on Tuesday; these assumptions are *discharged*. Disjunction elimination discharges each assumption ϕ_i from the line of reasoning that corresponds to that case.

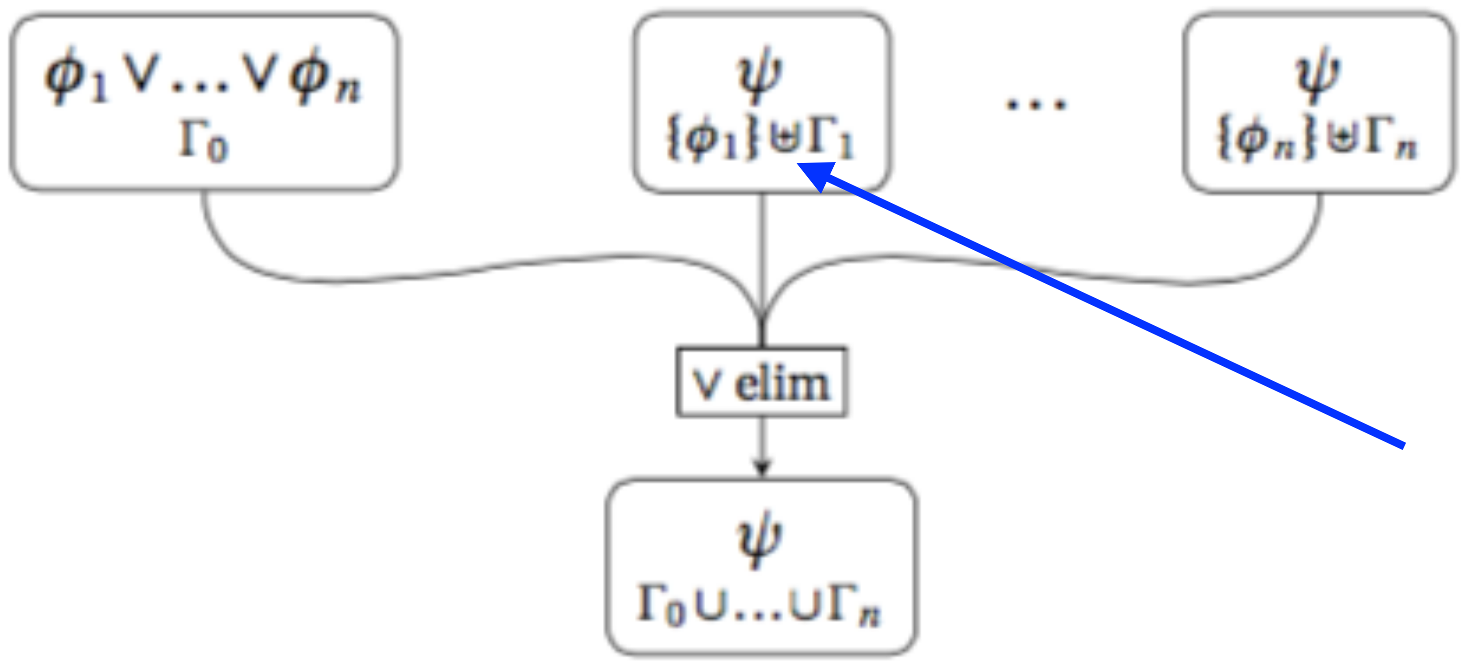


(2.25)

The various Γ_i on the premises of disjunction elimination might make this rule seem more complicated than it really is. Their presence makes it clear that the only assumptions discharged from each line of reasoning is the assumption corresponding to that particular case.

From ~ p. 54 in LAMA-BDLA

from each ϕ_i, then we may conclude ψ. That is, if we can, for each ϕ_i, assume ϕ_i and show that ψ follows, then we may conclude ψ from the disjunction $\phi_1 \vee \ldots \vee \phi_n$ and the derivations of ψ. There is one more subtle point, however. In the days-of-the-week example above, the conclusion that Susan has class on a weekday should not be in the scope of both the assumptions that she has class on Monday and that she has class on Tuesday; these assumptions are *discharged*. Disjunction elimination discharges each assumption ϕ_i from the line of reasoning that corresponds to that case.

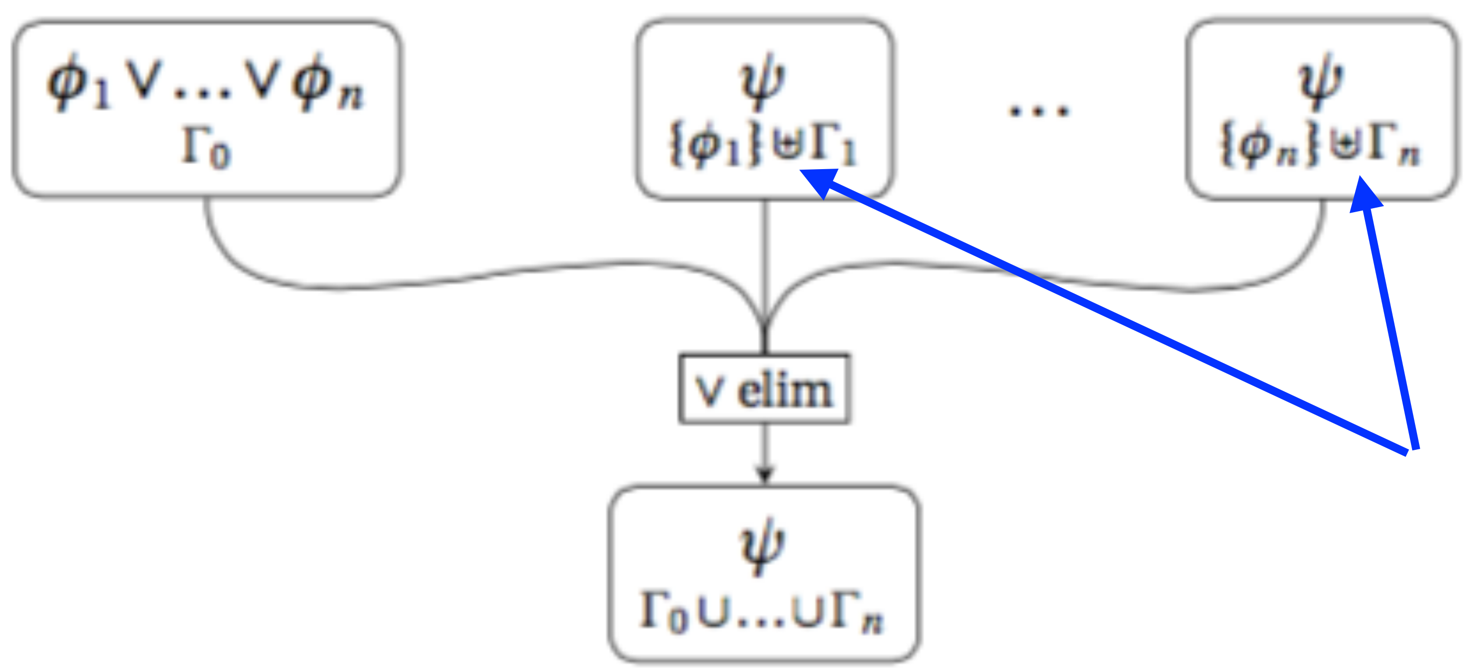


(2.25)

The various Γ_i on the premises of disjunction elimination might make this rule seem more complicated than it really is. Their presence makes it clear that the only assumptions discharged from each line of reasoning is the assumption corresponding to that particular case.

From ~ p. 54 in LAMA-BDLA

from each ϕ_i, then we may conclude ψ. That is, if we can, for each ϕ_i, assume ϕ_i and show that ψ follows, then we may conclude ψ from the disjunction $\phi_1 \vee \ldots \vee \phi_n$ and the derivations of ψ. There is one more subtle point, however. In the days-of-the-week example above, the conclusion that Susan has class on a weekday should not be in the scope of both the assumptions that she has class on Monday and that she has class on Tuesday; these assumptions are *discharged*. Disjunction elimination discharges each assumption ϕ_i from the line of reasoning that corresponds to that case.

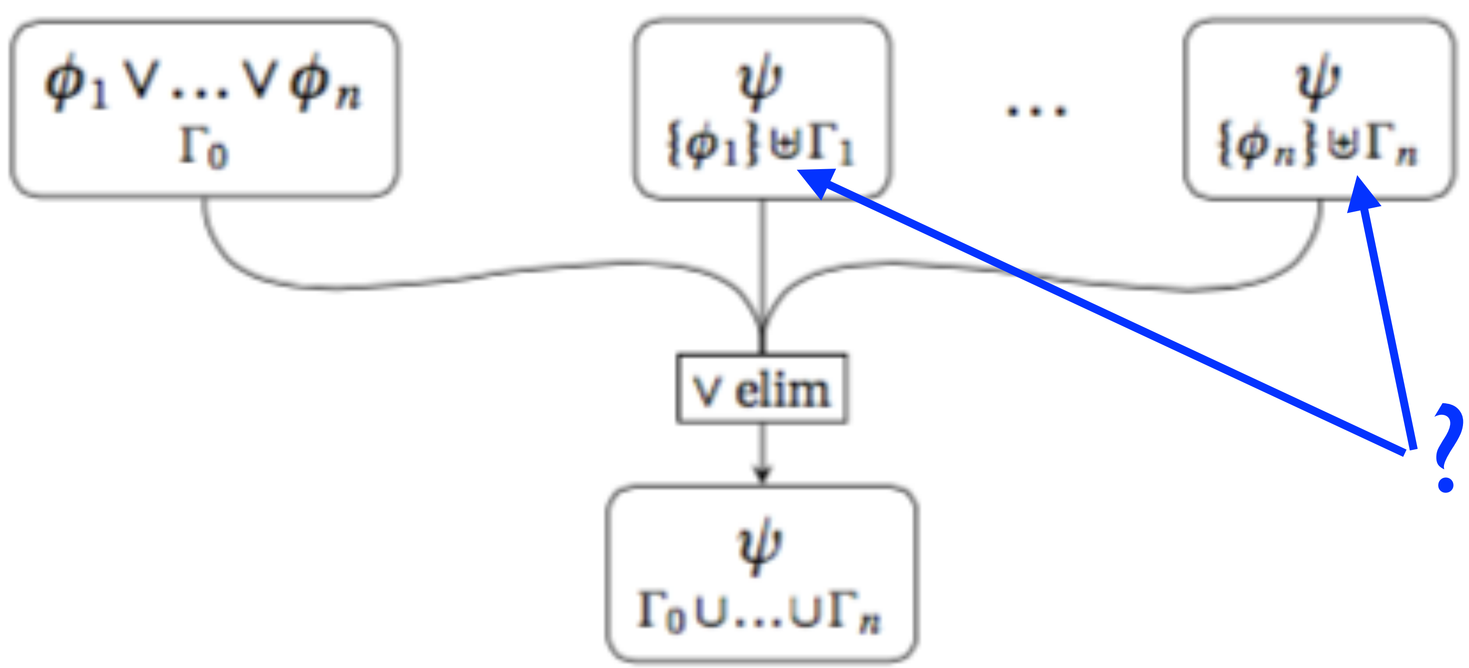


(2.25)

The various Γ_i on the premises of disjunction elimination might make this rule seem more complicated than it really is. Their presence makes it clear that the only assumptions discharged from each line of reasoning is the assumption corresponding to that particular case.

From ~ p. 54 in LAMA-BDLA

from each ϕ_i, then we may conclude ψ. That is, if we can, for each ϕ_i, assume ϕ_i and show that ψ follows, then we may conclude ψ from the disjunction $\phi_1 \vee \ldots \vee \phi_n$ and the derivations of ψ. There is one more subtle point, however. In the days-of-the-week example above, the conclusion that Susan has class on a weekday should not be in the scope of both the assumptions that she has class on Monday and that she has class on Tuesday; these assumptions are *discharged*. Disjunction elimination discharges each assumption ϕ_i from the line of reasoning that corresponds to that case.

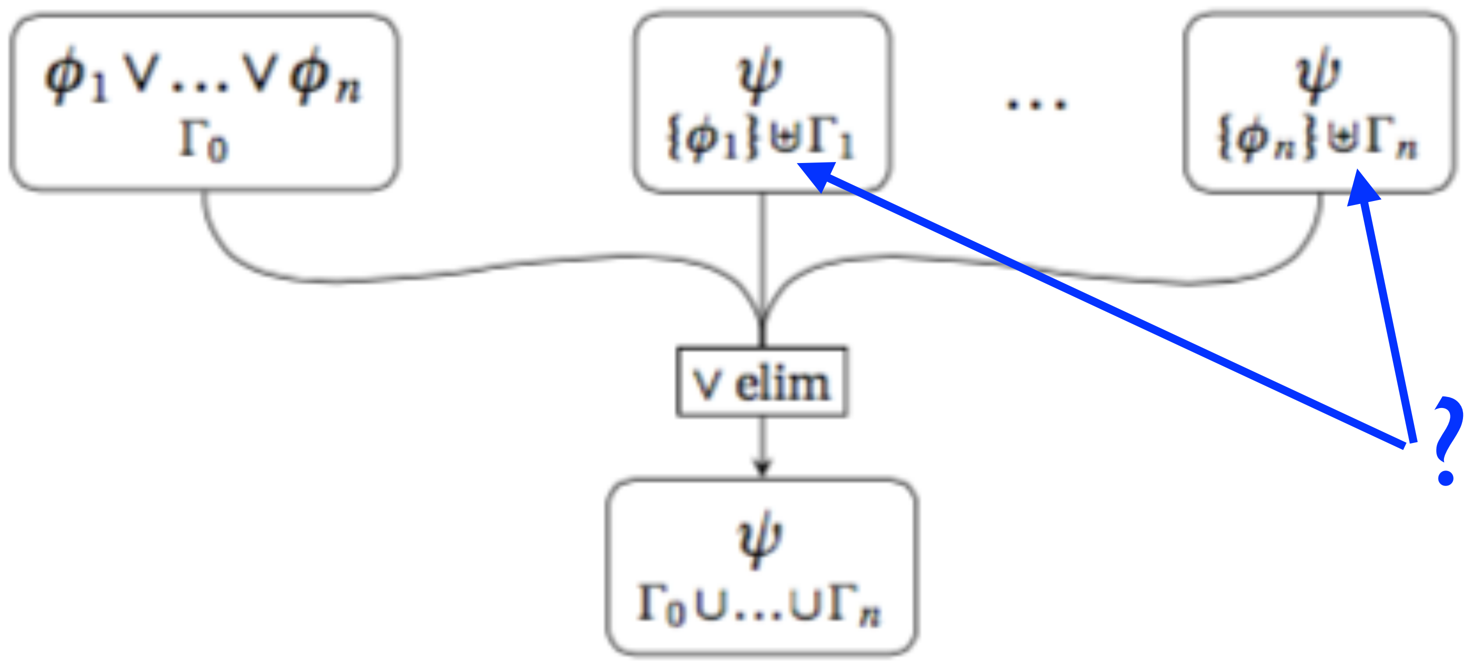


(2.25)

The various Γ_i on the premises of disjunction elimination might make this rule seem more complicated than it really is. Their presence makes it clear that the only assumptions discharged from each line of reasoning is the assumption corresponding to that particular case.

From ~ p. 54 in LAMA-BDLA

from each ϕ_i, then we may conclude ψ. That is, if we can, for each ϕ_i, assume ϕ_i and show that ψ follows, then we may conclude ψ from the disjunction $\phi_1 \vee \ldots \vee \phi_n$ and the derivations of ψ. There is one more subtle point, however. In the days-of-the-week example above, the conclusion that Susan has class on a weekday should not be in the scope of both the assumptions that she has class on Monday and that she has class on Tuesday; these assumptions are *discharged*. Disjunction elimination discharges each assumption ϕ_i from the line of reasoning that corresponds to that case.

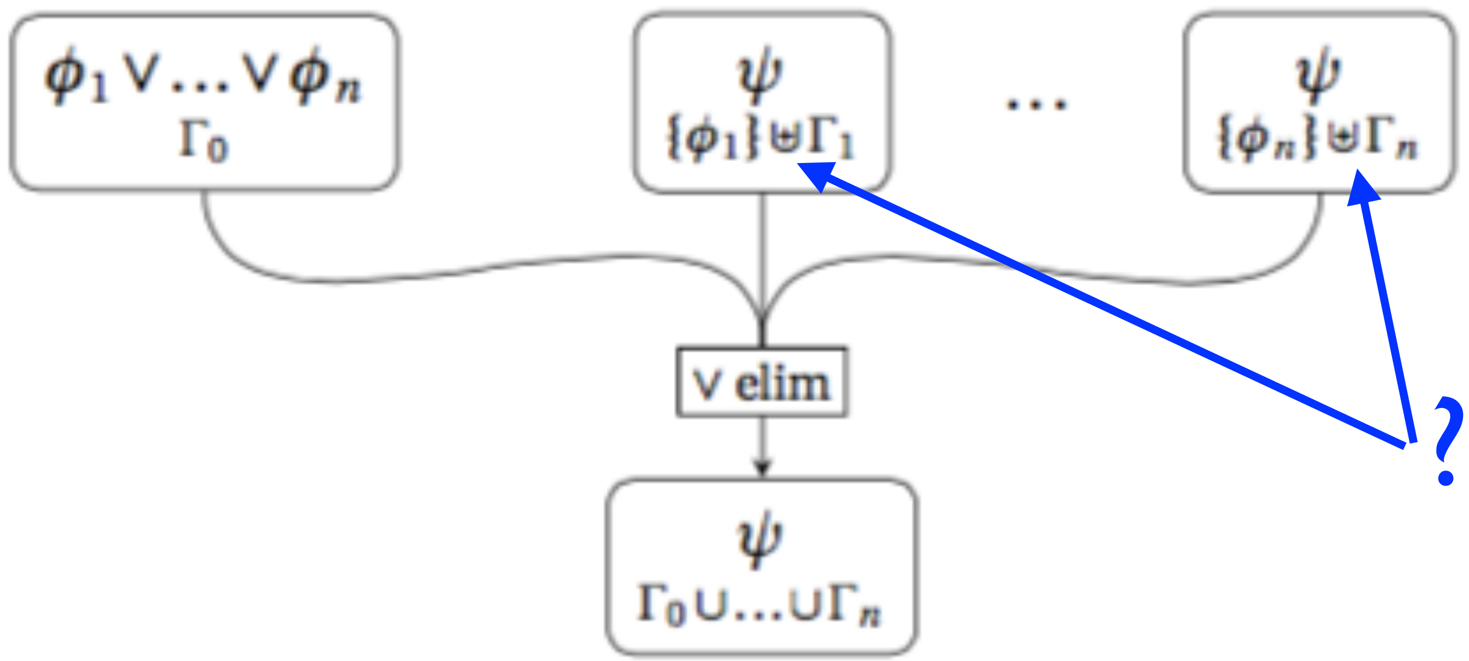


(2.25)

The various Γ_i on the premises of disjunction elimination might make this rule seem more complicated than it really is. Their presence makes it clear that the only assumptions discharged from each line of reasoning is the assumption corresponding to that particular case.

From ~ p. 54 in LAMA-BDLA

from each ϕ_i, then we may conclude ψ. That is, if we can, for each ϕ_i, assume ϕ_i and show that ψ follows, then we may conclude ψ from the disjunction $\phi_1 \vee \ldots \vee \phi_n$ and the derivations of ψ. There is one more subtle point, however. In the days-of-the-week example above, the conclusion that Susan has class on a weekday should not be in the scope of both the assumptions that she has class on Monday and that she has class on Tuesday; these assumptions are *discharged*. Disjunction elimination discharges each assumption ϕ_i from the line of reasoning that corresponds to that case.



(2.25)

The various Γ_i on the premises of disjunction elimination might make this rule seem more complicated than it really is. Their presence makes it clear that the only assumptions discharged from each line of reasoning is the assumption corresponding to that particular case.

King-Ace 2

Suppose that the following premise is true:

If there is a king in the hand, then there is an ace in the hand; or if there isn't a king in the hand, then there is an ace; but not both of these if-then statements are true.

What can you infer from this premise?

NO! ~~There is an ace in the hand.~~ NO!

In fact, what you *can* infer is that there *isn't* an ace in the hand!

Future Required problem (on HyperGrader®): You will need to finish the proof in HyperSlate® — with no remaining use of an oracle.

Det er en ære å lære formell logikk!

(Det er en ære å lære formell logikk!)

(Det er en ære å lære formell logikk!)

Further hands-on interaction in Part II of Class?