❖ *Proofs and Deductions* ❖

5.13. Quantifiers: Rules of Deduction

We noted earlier that the language of names, predicates, and quantifiers brings two innovations to the construction rules: a predicate letter followed by a name letter or variable, and quantified sentences. Any extensions this language brings to the deductive system of the previous chapter will be just those needed to capture the further valid arguments and logical truths of this extended language.

Since a predicate-letter-followed-by-sentence-letter or -variable forms an atomic formula, with no smaller formulas as parts, their addition to the formal language occasions no new rules of inferences (just as there are no deductive rules devoted to sentence letters). That means any new deductive rules needed will be solely for inferences involving quantifiers.

1. Quantifier Negation. The simplest rule for quantified sentences is an argument form encountered earlier: **Quantifier Negation**, in four forms.[1]

Quantifier Negation (QN)

Inward QN:

$$\frac{\sim\forall x \bullet}{\exists x \sim\bullet} \qquad \frac{\sim\exists x \bullet}{\forall x \sim\bullet}$$

Outward QN:

$$\frac{\exists x \sim\bullet}{\sim\forall x \bullet} \qquad \frac{\forall x \sim\bullet}{\sim\exists x \bullet}$$

[1] This form of inference was first encountered in 5.5. In fact we only need Inward QN as a basic rule here; Outward QN can be treated as a derived rule. See 5.13.1 Problem 1.

This formally implements intuitive arguments such as the following.

Not everything is made of limestone.	Everything is non-physical
∴ Something isn't made of limestone.	∴ Nothing is physical.
Something isn't made of limestone.	Nothing is physical
∴ Not everything is made of limestone.	∴ Everything is non-physical

Already this allows us deduction of the following argument.[2]

1. If everything is made of limestone, then Neko is.
2. (But) Neko's not made of limestone.

∴ Something's not made of limestone.

A: Neko **G__**: is made of limestone

1. $(\forall x\ Gx \rightarrow GA)$
2. $\sim GA$

Get: $\exists x \sim Gx$

3. $\sim\forall x\ Gx$ 1, 2, MT
4. $\exists x \sim Gx$ 3, QN

[2] As we will later show, the first premise here is a logical truth; so the conclusion follows validly from the second premise alone. See the discussion of the Universal Elimination rule, below, and 5.13.1 Problem 2.

2. Universal Elimination. We retain here our semantic focus on full-fledged sentences (rather than mere quasi-sentences), by retaining as well the earlier concept of an **instance** of a quantified sentence. Recall that to build an instance of a quantified sentence we remove the quantifier from the sentence, and then replace the variable in that quantifier according to the following three conditions.[3]

> (A) Replace **all free** occurrences of the variable with name letters.
> (B) Replace **only free** occurrences of that variable with name letters.
> (C) Replace all free occurrences of that variable with the **same** name letter.

For example, to build an instance of "∀x ((Jx ∨ Hx) → ∃x ~Gx)" we remove the quantifier "∀x", and replace all and only the free occurrences of its variable – "x" – by a name letter.

So of the following formulas, only (1) and (2) are instances of "∀x ((Jx ∨ Hx) → ∃x ~Gx)".[4]

(1) ((JA ∨ HA) → ∃x ~Gx)	(A/x)
(2) ((JB ∨ HB) → ∃x ~Gx)	(B/x)
(3) ((JA ∨ Hx) → ∃x ~Gx)	(A/x)
(4) ((JA ∨ HA) → ∃x ~GA)	(A/x)
(5) ((JA ∨ HB) → ∃x ~Gx)	(A,B/x)

(3) violates the '**all free occurrences**' requirement: the occurrence of "x" in "Hx" is left free. (4) violates the '**only free occurrences**' requirement: when "∀x" is removed from "∀x ((Jx ∨ Hx) → ∃x ~Gx)" the occurrence of "x" in "∃x ~Gx" remains bound by "∃x". And (5) violates the '**same name letter**' requirement: when "∀x" is removed, the two free occurrences of "x," in "Jx" and "Hx," aren't replaced by the same name letter.

---

[3] As set out and explained in 5.8.

[4] As in earlier readings, the notation on the right side – for example, "(A/x)" – records which name letter replaces the quantified variable. In the case of (A/x) – pronounced "A for x" – name letter "A" replaces variable "x".

With this understanding of "instance", the inference rule for universal sentences is straightforward.

Universal Elimination ("A-Elim")($\forall -$)[5]

$$\frac{\forall x \bullet}{\bullet_I}$$

where $\bullet_I$ is an **instance** of the scope formula $\bullet$

This rule implements formal counterparts of intuitively valid inferences such as the following.

1. Everything is made of matter.

$\therefore$ The Cathedral of Learning is made of matter.

For instance, armed with $\forall -$ we can demonstrate the validity of the following intuitively valid English argument.

A: Jack **G__**: is a surfer
H__: is an athlete

1. All surfers are athletes.
2. Jack is a surfer.

$\therefore$ Jack is an athlete.

1. $\forall x\,(Gx \rightarrow Hx)$
2. GA

$\therefore$ HA

---

[5] This is sometimes called "Universal Instantiation". We call it "A Elim" for short (rather than "U Elim" for "universal") because the upside-down "A" is being removed. This is in keeping with the naming system of earlier chapters, which featured the rules "Wedge Elim" (rather than "Conjunction Elim") and "Vel Elim" (rather than "Disjunction Elim").

1. $\forall x\ (Gx \rightarrow Hx)$

2. GA

Get: HA

3. $(GA \rightarrow HA)$	1, $\forall-$ (A/x)
4. HA	2, 3, MP

3. Existential Elimination. Matters are trickier with existential sentences. For on the one hand it seems intuitive that here too a quantified sentence licenses an instance: if there exists something G, then there must be some particular object which is truly said to be G. Yet as we saw earlier with truth trees, matters can't be as simple as just instantiating an existential to a single name letter. The following argument is obviously invalid, but will be easy to deduce if existential instantiation is left unconstrained.

☠ Invalid ! ☠

Dr. Slim is a man. Someone stole the crown jewels. (Therefore,) Dr. Slim is a man who stole the crown jewels.

B: Dr. Slim **I**__: is a man
J__: stole the crown jewels

1. IB

2. $\exists x\ Jx$

Get: $(IB \wedge JB)$

3. JB	2, Existential Elimination (B/x)
4. $(IB \wedge JB)$	1, 3, $\wedge$+

Since the name letter "B" appears already on Line (1), it's 'already taken' by someone; so instantiating the existential sentence to that name letter is far from innocent. (Likewise in English: while it's fine to call the jewel thief "Mr. X," or some other name which no one is using, the name "Dr. Slim" was already taken; so it was a mistake to use that name for the culprit.)

Therefore in deductions (as in truth trees) we impose a 'new name' requirement on existential instantiation: the name being instantiated to must be **new** to the deduction – i.e., must **not have appeared on previous lines of the deduction**.

But care is needed when speaking of "previous lines". For instantiating the existential sentence to a name letter can be invalid even if no earlier *numbered* line in the deduction featured that name letter. The following argument, for instance, is clearly invalid.

☠ Invalid ! ☠

> Something is made of marshmallow. (Therefore,) the Cathedral of Learning is made of marshmallow.
>
> **C**: The Cathedral of Learning **K__**: is made of marshmallow
>
> 1. ∃x Kx
>
> ________________________~~Get~~: KC
>
> 2. KC 1, Existential Elimination (C/x)

But the only numbered line before the existential elimination is Line (1), which does not contain the name letter "C"; so "C" is new to the *numbered* lines when the instantiation occurs on Line (2). Since the argument is clearly invalid, **simply requiring the name to be new *among the numbered lines* is not a strong enough constraint on existential elimination**.

Note that "C" does appear before Line (2) – on the "Get" line. We don't count the "Get" line as a line in the deduction, in the sense that we could apply any rule of inference to it; it's just a memo off to the side, reminding us what the deduction is aiming for. Still, if we include the "Get" line as part of the 'new-ness' constraint on instance variables, we correctly block the deductive system from counting the above argument as valid.[6]

So we impose this strengthened 'newness' condition on names in existential elimination.

Existential Elimination (**E-Elim**) (∃–)[7]

$$\frac{\exists x \bullet}{\bullet_I}$$

where (i) $\bullet_I$ is an **instance** of the scope formula $\bullet$,

and

(ii) the **name letter** used in that instance (to replace the quantified variable) is "**new** to the deduction" – that is, does not appear on any previous lines, *including "Get" lines*.

---

[6] So the mere mention of the name letter (in the "Get" memo) rules out its use in ∃– afterwards. Though truth trees don't feature "Get" lines, they impose the same constraint on existential instances: since the first steps of a truth tree are listing the premises (on the left) and conclusion (on the right), any name letters appearing in that conclusion will be barred from appearing in later existential instances, thanks to the newness constraint on the True Existential rule. See

In this respect the "Get" line serves as a filter on existential instances just as the "Show" line does in (Kalish and Montague 1964: 100). While the "Show" line in Kalish and |Montague's system is a numbered line to which (once the word "Show" is crossed off) rules of inference can be applied, it acts as a filter on existential instances **whether or not it's cancelled**. As noted in the second edition (Kalish, Montague, and Mar 1980: 154-155), even a weaker 'newness' constraint blocks the inference both in cases where a prior 'Show' line is already cancelled and in cases where it's not. See likewise the discussion of Problem 69 on (ibid: 159) and Problem 70 on (ibid: 160).

By including the "Get" line in the criterion for 'newness' of a line, without allowing rules of inference to apply to any "Get" line (crossed off or not), the present system retains Kalish and Montague's simplified rule of Existential Instantiation (here, Existential Elimination) while also retaining the top-to-bottom reading order of a Fitch-style deductive system (see e.g. Fitch 1952: xx or Thomason 1970: yy). Further details on the two types of deductive systems can be found in (Pelletier 1999), which provides an extensive discussion of their historical development.

[7] This is sometimes called "Existential Instantiation" (EI).

The following deduction illustrates ∃– and ∀– in combination.

G__: is a cat **H__**: is a fish-eater

1. All cats are fish-eaters.	1. $\forall x\ (Gx \rightarrow Hx)$
2. Cats exist.	2. $\exists x\ Gx$
∴ Fish-eaters exist.	∴ $\exists x\ Hx$

1. $\forall x\ (Gx \rightarrow Hx)$

2. $\exists x\ Gx$

		Get: $\exists x\ Hx$ (ID)
3.	$\sim\exists x\ Hx$	AID
4.	GA	2, ∃– (A/x)
5.	$(GA \rightarrow HA)$	1, ∀– (A/x)
6.	HA	4, 5, MP
7.	$\forall x \sim Hx$	3, QN
8.	$\sim HA$	7, ∀– (A/x)
9.	$\exists x\ Hx$	3, 6, 8, ID

4. Deductive Strategy. It was no coincidence that we used ∃– before ∀– in that last deduction. For suppose instead we apply ∀– to line 1 first.

1. $\forall x\ (Gx \rightarrow Hx)$

2. $\exists x\ Gx$

		Get: $\exists x\ Hx$ (ID)
3.	$\sim\exists x\ Hx$	AID
4.	$(GA \rightarrow HA)$	1, ∀– (A/x)

If we then seek to set up MP on line 4, by getting its antecedent from line 2 by ∃–, we're stuck: since ∃– requires that we instantiate to a **new** name letter, we can't ∃– to "GA" on Line (5) (only, e.g., to "GB" or "GC").

1.	$\forall x\ (Gx \rightarrow Hx)$	
2.	$\exists x\ Gx$	
		Get: $\exists x\ Hx$ (ID)
3.	$\sim\exists x\ Hx$	AID
4.	$(GA \rightarrow HA)$	1, ∀– (A/x)

That points up an important bit of deductive strategy: since ∃– is limited to a new name letter while ∀– isn't, it's shrewd to **use ∃– before using ∀–.**

Deduction Strategy: Use ∃– before using ∀–.

As their names make clear, ∃– and ∀– are **Elim rules**. So these two rules are added to the Elim rules of Chapters Three and Four.

But, as noted already in the truth tree rules for quantifiers, a universally quantified sentence can entail an unlimited number of instances. To avoid such a limitless cascade of sentences, our strategy will be to use ∀– within an indirect deduction to deduce only as many instances of a universal as needed to conflict with instances of existential sentences and other sentences containing name letters. Here, as in truth trees, universals follow up on (instances of) existential sentences and other sentences with name letters.

As regards Quantified Negation, our earlier division of the rule into Inward and Outward versions was for a good cause. We count **Inward QN** as an **Elim rule**, to be used whenever possible.

Inward Quantifier Negation

$$\frac{\sim\forall x\ \bullet}{\exists x \sim\bullet} \qquad \frac{\sim\exists x\ \bullet}{\forall x \sim\bullet}$$

Outward QN is a **setup rule**, used – like the Intro rules – only to construct a needed part of an inference or deduction.

Outward Quantifier Negation

$$\frac{\exists x \sim\bullet}{\sim\forall x\ \bullet} \qquad \frac{\forall x \sim\bullet}{\sim\exists x\ \bullet}$$

Deduction Strategy: Treat **Inward QN** as an **Elim rule**, using it whenever possible. Treat **Outward QN** as a **Setup Rule** (like an **Intro rule**), using it only to get sentences needed to complete a deduction or perform an Elim rule.

Quantifier Deduction Rules

Quantifier Negation (QN)

Inward QN: **Outward QN:**

$$\frac{\sim\forall x\ \bullet}{\exists x \sim\bullet} \qquad \frac{\sim\exists x\ \bullet}{\forall x \sim\bullet} \qquad \frac{\exists x \sim\bullet}{\sim\forall x\ \bullet} \qquad \frac{\forall x \sim\bullet}{\sim\exists x\ \bullet}$$

Universal Elimination ("**A-Elim**")($\forall$**–**)

$$\frac{\forall x\bullet}{\bullet_I}$$

where $\bullet_I$ is an **instance** of the scope formula $\bullet$

Existential Elimination ("**E-Elim**") ($\exists$**–**)

$$\frac{\exists x\bullet}{\bullet_I}$$

where (i) $\bullet_I$ is an **instance** of the scope formula $\bullet$, and (ii) the **name letter** used in that instance (to replace the quantified variable) is **new** to the deduction – that is, does not appear on previous lines, including "Get" lines.

Quantifier Deduction Strategy

- Treat ∀**–** and ∃**–** like **Elim rules**: use whenever possible.
- Use ∃**–** before ∀**–**.
- Treat **Inward QN** as an **Elim rule**: use whenever possible.
- Treat **Outward QN** as a **Setup rule** (like an **Intro rule**): only to get a missing sentence to complete a deduction or perform an Elim rule.