5.8. Quantifier Semantics Revisited: Instances (Simple Version)

1. Instances. So long as we restrict ourselves to sufficiently simple cases[1], what counts as an **instance** of a quantified sentence can likewise be stated simply: (i) **remove the quantifier** from the left of that sentence, then (ii) **replace the variable of quantification** (the variable in that just-removed quantifier) with a **name letter** in the scope formula.

For example, with "∃x Gx" we remove "∃x", leaving just the scope formula "Gx". We then replace the "x" in "Gx" with a name letter – yielding, say, "Ga" or "Gb".

Scope Formula:	**Instances of This Formula**
G**x**	G**a**
	G**b**
	(etc.)

With a sentence containing multiple occurrences of the same variable, we construct an instance by (i) removing the quantifier from the left of that sentence, then (ii) replacing every occurrence of the variable of quantification with a name letter in the (remaining) scope formula.

For example, with "∃x (Gx ∧ Hx)" we remove "∃x," leaving just the scope formula "(Gx ∧ Hx)". Then we replace every occurrence of "x" in "(Gx ∧ Hx)" with a name letter.

Scope Formula:	**Instances of This Formula**
(G**x** ∧ H**x**)	(G**a** ∧ H**a)**
	(G**b** ∧ H**b**)
	(etc.)

To achieve the correct results (in terms of validity and logical truth), two points are important to keep in mind.

---

[1] Specifically: so long as different quantifiers in the sentence don't have overlapping scopes – as would happen in, e.g., the sentences "∀x ∃x Gx" and ∃x (Gx ∧ ∃x ~Gx)".

First, once the quantifier is removed from the left of the sentence, all the variables must be replaced in the scope formula that remains. That's simply to ensure that the resulting instance is indeed a sentence, rather than a (mere) quasi-sentence. So in that last example, beginning with the sentence "∃x (Gx ∧ Hx)" and removing the quantifier "∃x" leaves the scope formula "(Gx ∧ Hx)". If, say, only the first occurrence of "x" were replaced with name letter "a", the result would be the quasi-sentence "(Ga ∧ Hx)" – for which we have no semantic interpretation. Since the point of building instances is to have semantically interpretable sentences which then provide semantics for quantified sentences, the whole purpose of instances is undercut if we don't replace every occurrence of the variable when constructing instances.

> **First requirement** (for building an instance out of the scope formula): replace **every** occurrence of the variable with a name letter.

Second, we must replace all of those variable occurrences with the **same name letter** throughout. For if we don't – replacing one occurrence with one name, another occurrence with some different name letter – we'll get the wrong semantic results.

For example, the following sentence looks like a contradiction.

∃x (Gx ∧ ~Gx)

And as long as the instances we build from this sentence use the same name letter throughout, we achieve the right result: all of its instances are contradiction, hence **in every model** "∃x (Gx ∧ ~Gx)" will have **no true instances**.

Scope Formula:	**Instances of This Formula**
(G**x** ∧ ~G**x**)	(G**a** ∧ ~G**a**) (G**b** ∧ ~G**b**) (etc.)

But if we use different name letters for the different occurrence of "x" in this scope formula, we'll get an instance that **is** true in some model.

☠ Instance of "∃x (Gx ∧ ~Gx)" ? ☠

(Ga ∧ ~Gb)

Here we've replaced the first occurrence of "x" in "(Gx ∧ ~Gx)" with name letter "a", and the second occurrence with "b".

☠ Instance of "(Gx ∧ ~Gx)" ? ☠

(Gx ∧ ~Gx)
↓ ↓
(G<u>a</u> ∧ ~G<u>b</u>)

But "(Ga ∧ ~Gb)" is true in some models – in the following model, for example.

G__: is a cat

𝔻: {**Neko**, **Rex**}

a: **Neko** **G**: {**Neko**}
b: **Rex**

By not using the **same** name letter throughout, when replacing occurrences of "x", the contradictory "∃x (Gx ∧ ~Gx)" is wrongly claimed to have an instance that's true in some model. That's why we impose the further requirement on instances of a quantified sentence.

> **Second requirement** (for building an instance out of the scope formula): replace all occurrences of the variable with **the same name letter**.

2. Quantifier Semantics Revisited. With the official version of "instance" in hand, the semantics for quantified sentences is straightforward.

> An **existential** sentence is **true** in a model if (and only if) it has **at least one true instance** in that model.
>
> An **existential** sentence is **false** in a model if (and only if) **every instance** of that sentence is **false** in the model.
>
> A **universal sentence** is **true** in a model if (and only if) **every instance** of that sentence is **true** in the model.
>
> A **universal sentence** is **false** in a model if (and only if) it has **at least one false instance** in that model.

Summary: Instances and Quantifier Semantics

- **Instance of a Quantified Sentence**: an instance of a quantified sentence is the result of (i) removing the (left-most) quantifier from that sentence and then (ii) in the scope formula that remains, replacing the variable of quantification (the variable appearing in that quantifier) with a name letter, according to the following three constraints.

  (1) Replace **all free** occurrences of the variable.
  (2) Replace occurrences of the variable by the **same** name letter throughout.

- **Existential Semantics**:

  An **existential** sentence is **true** in a model if (and only if) it has **at least one true instance** in that model.

  An **existential** sentence is **false** in a model if (and only if) **every instance** of that sentence is **false** in the model.

- **Universal Semantics**:

  A **universal sentence** is **true** in a model if (and only if) **every instance** of that sentence is **true** in the model.

  A **universal sentence** is **false** in a model if (and only if) it has **at least one false instance** in that model.