5.X. Pure Sentences, Instances, and Quantifier Semantics Revisited

1. Pure Formulas. Our first encounter with instances of a formula was in providing semantics for a small subset of the Chapter Five language, where quantifiers were applied only to predicate basics – a predicate atom or its negation. That much allowed us to construct instances of the following sorts of formulas.

$$\forall x\, Gx \qquad \exists y\, Hy \qquad \forall x\, \sim Ix$$

We focused just on instances of these simple sorts of formulas because the account of instances for any quantified formula in the language turns out to be rather complex. By contrast, instances for these simple sorts of quantified formulas are simple: remove the quantifier from the left of the sentence, and in the scope formula that remains replace the variable with a name letter. So instances of "$\forall x\, \sim Ix$," for instance, include the following.

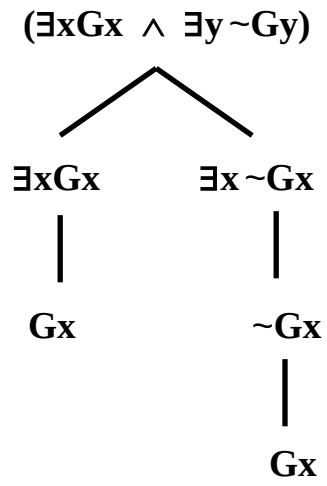
> Formula: $\forall x\, \sim Ix$
> Instances: $\sim Ia$, $\sim Ib$, $\sim Ic$, (etc.)

In our ascent to semantics for the full formula language we pause here, for similar reasons, to explore another 'sub-language': the family of **pure formulas**.[1] For here again we'll find that the simplicity of these formulas yields a likewise simple account of instances.

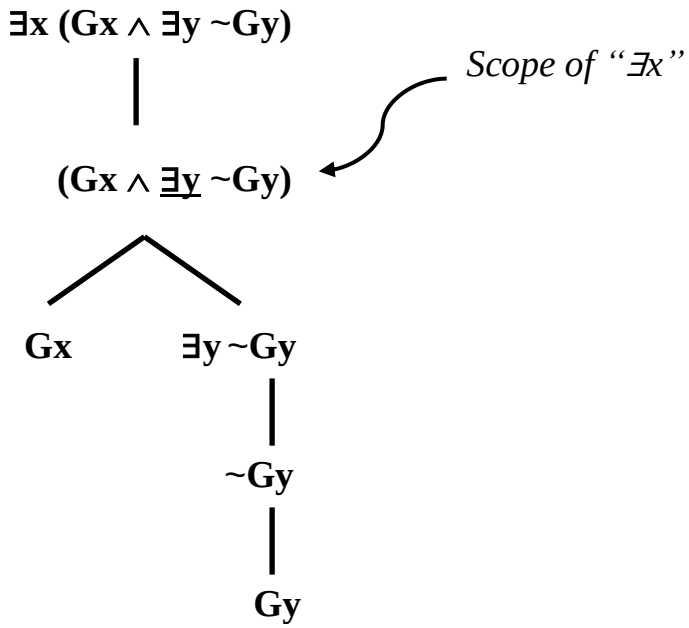
And pure formulas of Chapter Five have a further marvelous feature: as we'll later prove, every formula in the Chapter Five formal language is semantically equivalent to a pure formula.[2] So in (briefly) restricting ourselves to pure formulas we sacrifice no expressive power.

[1] Adopting "pure" from (Quine 1982: 152).
[2] As proven in 5.X.

A formula is pure if no quantifier shows up within the scope of another quantifier. So "**(∃xGx ∧ ∃y ~Gy)**" **is a pure formula**, because though it has two quantifiers, neither appears within the scope of the other. (As the construction tree makes clear, the scope of "**∃xGx**" is just "Gx," and the scope of "**∃y ~Gy**" is "~Gy".)

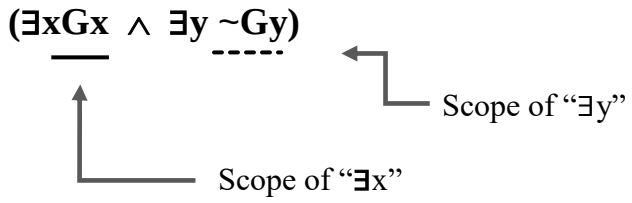


By contrast, "**∃x (Gx ∧ ∃y ~Gy)**" **isn't a pure formula**, because the quantifier "**∃y**" appear within the scope of "**∃x**".

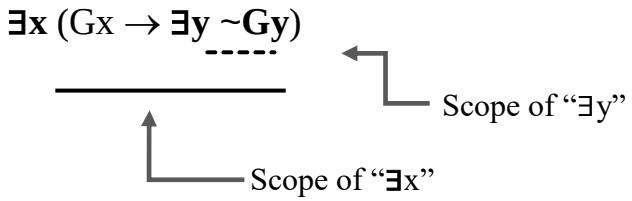


> **Pure formula**: a formula where no quantifier appears within the scope of another quantifier (that is: where quantifier scopes don't overlap).

Phrasing the definition in terms of scope overlap provides another quick way of understanding the concept. For note that in the pure formula "**(∃xGx ∧ ∃y ~Gy)**" the scopes of "**∃x**" and "**∃x**" don't overlap.



Whereas in the impure formula "**∃x (Gx ∧ ∃y ~Gy)**" the scopes of "**∃x**" and "**∃x**" do overlap.



So all of the following formulas are thus pure formulas.

$\forall x\,(Gx \wedge Hy)$ $\quad$ $(\forall x\,Gx \rightarrow (Gw \vee \exists y\,Jy))$
$(\exists z\,(Gz \rightarrow Hx) \wedge \forall x\,(Hx \wedge Ix))$

But none of the following are pure formulas.

$\forall x\,(Gx \wedge \exists y\,Hy)$ $\quad$ $\exists z\,(Gz \rightarrow \forall x\,(Hx \wedge Ix))$
$\forall x\,\exists y\,(Gx \wedge Hy)$ $\quad$ $(\forall x\,Gx \rightarrow \forall w\,(Gw \vee \exists y\,Jy))$

Since this is the only constraint placed on Chapter Five formulas to count as a pure formula, a small change to our construction rules suffices to stake out the pure formulas. The Chapter Five construction rules ran like so.

Atomic Formulas:

A1. Sentence letters are atomic formulas
A2a. A predicate letter followed by a name letter is an atomic formula.
A2b. A predicate letter followed by a variable is an atomic formula.

Formulas:

1. Atomic formulas are formulas.
2. If ● is a formula, then ~● is a formula.
3. If ● and ▲ are formulas, then (● ∧ ▲) is a formula.
4. If ● and ▲ are formulas, then (● ∨ ▲) is a formula.
5. If ● and ▲ are formulas, then (● →▲) is a formula.
6. If ● and ▲ are formulas, then (● ↔▲) is a formula.
7a. If ★ is a variable and ● is a formula, then ∃★ ● is a formula.
7b. If ★ is a variable and ● is a formula, then ∀★ ● is a formula.

To restrict these to pure formulas we only need to change the quantifier construction rules (7a) and (7b).

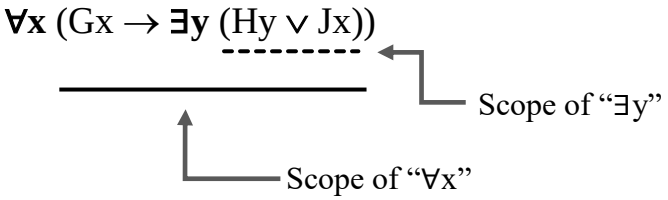
P7a. If ★ is a variable and ● is a **formula containing no quantifiers**, then ∃★ ● is a pure formula.

P7b. If ★ is a variable and ● is a **formula containing no quantifiers**, then ∀★ ● is a pure formula.

2. Pure Formulas: Instances.

*****

But it's convenient to focus on a special sub-species of these formulas – the **pure formulas** – built by restricting the quantifier construction rules (7a and 7b).[3] As they stand, (7a) and (7b) allow us to attach a quantifier to the left of any old formula – even a formula which itself already contains quantifiers. That permits the scopes of different quantifiers to overlap. In the next formula, for instance, "∀x" and "∃y" have **overlapping scopes**.



Pure formulas avoid this overlap of quantifier scope by only attaching a quantifier to the left of a formula that's quantifier-free.
So the construction rules for pure formulas are just the existing construction rules, except with this added requirement on the quantifier rules.

> P7a. If ★ is a variable and ● is a **formula containing no quantifiers**, then ∃★ ● is a pure formula.
>
> P7b. If ★ is a variable and ● is a **formula containing no quantifiers**, then ∀★ ● is a pure formula.

All of the following formulas are thus pure formulas.

> ∀x (Gx ∧ Hy) (∀x Gx → (Gw ∨ ∃y Jy))
> (∃z (Gz → Hx) ∧ ∀x (Hx ∧ Ix))

But none of the following are pure formulas.

[3] Adopting "pure" from (Quine 1982: 152).

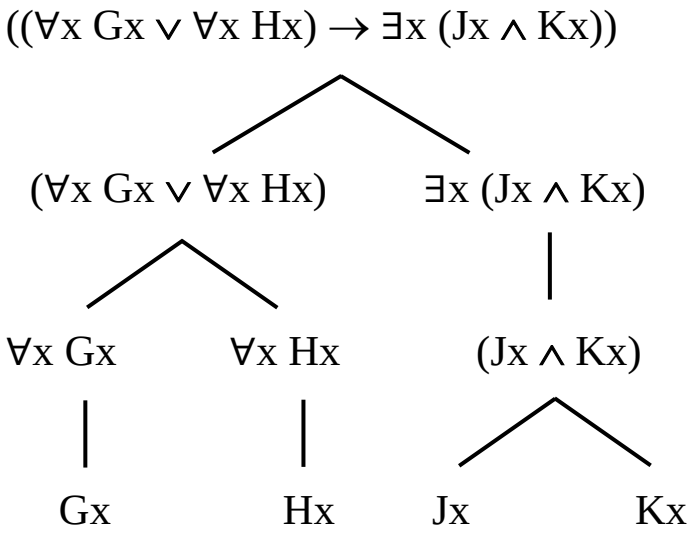
$\forall x\ (Gx \wedge \exists y\ Hy)$ $\quad\quad$ $\exists z\ (Gz \rightarrow \forall x\ (Hx \wedge Ix))$
$\forall x\ \exists y\ (Gx \wedge Hy)$ $\quad\quad$ $(\forall x\ Gx \rightarrow \forall w\ (Gw \vee \exists y\ Jy))$

As we see in the previous examples, a pure formula can contain more than one quantifier – so long as no quantifier appears within the scope of any of the others. So the following qualifies as a pure formula.[4]

$((\forall x\ Gx \vee \forall x\ Hx) \rightarrow \exists x\ (Jx \wedge Kx))$

The scope formula of the first universal quantifier is "Gx"; the scope of the second universal is "Hx"; and the scope of the existential quantifier is "$(Jx \wedge Kx)$". Since none of these scope formulas contain quantifiers, this formula is indeed a pure formula.

[4] In construction of a pure formula, the output of the quantifier construction rules, (P7a) and (P7b), can't be fed back into one of those quantifier rules as fresh input. So to get more than one quantifier in a pure formula, the output(s) of the quantifier rules must instead be fed as input into a rule for a two-placed connective – the conjunction, disjunction, conditional, or biconditional rules (3)-(6) – as in the following example.



Just as a sentence is a formula with no free variables, **a pure sentence** is a pure formula with no free variables; and a **pure quasi-sentence** is a pure formula with a free variable. As before, for purposes of translating English sentences and building their semantic profiles we're really only interested in sentences.

Pure sentences are interesting for these purposes because (a) it turns out every sentence in the Chapter Five language is logically equivalent to a pure sentence, yet (b) the semantics for pure sentences is easier to explain and to use.[5] That semantics comes via the **instances** of pure universal and existential sentences.

2. Pure Instances. We earlier encountered **instances** of the simplest sort of quantified sentence, such as "∀x Gx" and "∃x ~Hx," where the scope formula of the quantifiers was a predicate atom or negation of a predicate atom.[6] In that sort of case it was easy to build an instance of the quantified sentence: we (1) remove the quantifier from the left of the sentence, then (2) replace the variable of quantification (the variable used in that just-removed quantifier) with a name letter.

[5] The equivalence comes through the **rules of passage**, discussed in 5.X.
[6] In 5.X.

For instance, to construct instances of the quantified sentence "∃x ~Hx" we remove "∃x" from the left, leaving its scope formula "~Hx".

~~∃x~~ ~H**x**

Then we replace the "x" in the scope formula with (an occurrence of) a name letter – say, "a" or "b" or "c".

Scope Formula:	**Instances of This Formula**
~H**x**	~H**a** ~H**b** ~H**c** (etc.)

But for this method of building instances to scale up to larger pure quantified sentences, we must follow certain technical constraints – on pain of yielding the wrong semantic results. In general, an **instance of a pure quantified sentence** (however complex) must meet the following two conditions.

a) **All** free variable occurrences must be replaced with a name letter.

b) Occurrences of free variables must be replaced by the **same name letter** throughout the scope formula.

Call them the "**all**" and "**same**" requirements.

Why we follow the "all" requirement is obvious: we want an instance to be a **sentence**, capable of truth or falsehood (rather than a mere quasi-sentence). But if we don't replace every free variable occurrence with a name letter, a free variable occurrence will remain in the resulting formula – making that formula a quasi-sentence rather than a sentence.

To see why we follow the "same" requirement, consider: the following pure quantified sentence looks like a **contradiction**. Intuitively, there should be **no possible way for this sentence to be true**.

$$\exists x\,(Gx \wedge \sim Gx)$$

To construct an instance of this sentence, we remove the quantifier "∃x", and replace every free occurrence of "x" in scope formula "(Gx ∧ ~Gx)" with a name letter. Now if, following the "same" condition, we use the same name letter throughout, we end up with an instance of the following sort.

(Ga ∧ ~Ga)
(Gb ∧ ~Gb)
(Gc ∧ ~Gc)
(etc.)

Since all of these are contradictions, true in no model, the same holds for the original quantified sentence – **the correct result**.[7]

But suppose we disregard the "all" condition, and replace different occurrences of "x" in "(Gx ∧ ~Gx)" with different name letters. Then the following would be treated as an instance of "∃x (Gx ∧ ~Gx)".

☠ Instance of "∃x (Gx ∧ ~Gx)" ? ☠

(Ga ∧ ~Gb)

Now, "(Ga ∧ ~Gb)" can certainly be true in a model – for example, in a a model with just two objects, cat Neko and non-cat Rex.

(9) (G**a** ∧ ~G**b**) **G__**: is a cat

$\mathbb{D}$: {**Neko**, **Rex**}

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

[7] So a contradiction is a sentence all of whose instances are also contradictions – just as in sentence logic (as noted in 2.17.1 Problem F).

If "(Ga ∧ ~Gb)" counts as an instance of "∃x (Gx ∧ ~Gx)", then "∃x (Gx ∧ ~Gx)" has at least one true instance in a model, and so wouldn't count as a contradiction after all.

That seems like **the wrong result**. Hence our insistence on the "same" condition: when replacing free occurrences of a variable in the scope formula, use the **same name letter** throughout.

***

[Later can note that *pure instances* all qualify as *instances*. (since pure instance is an instance of a pure sentence). So "all" and "same" conditions alone won't always yield an instance of a formula (specifically when there's quantifier overlap). But those conditions applied to a pure sentence will always yield an instance – since pure sentences by their natural can't violate the "only free" requirement that's later added.]

***

Summary: Sentences, Formulas, and Binding

- For a **quantifier** to **bind** an **occurrence of a variable**:

  (1) the variable-occurrence being bound must be the same variable appearing in that quantifier,

  and

  (2) the variable-occurrence being bound must appear within the *scope* of that quantifier.

- A **free** variable-occurrence is one that is not bound.

- A **variable** is **free** if it has a free occurrence; a **variable** is **bound** if it has a bound occurrence. (So a variable can be both bound and free. Each variable-occurrence, however, is bound or free, but not both.)

- A **quasi-sentence** is a formula with at least one free variable.

- A **sentence** is a formula with no free variables.