

5.9. Quantifier Truth Trees

We adapt our earlier truth tree rules to the expanded formal semantics.

We've added only two new aspects to the semantics of previous chapters: a truth rule for a **name atom** (predicate-letter-plus-name-letter) and truth rules for **universal and existential sentences**.

Precisely because they're atoms, name atoms poses no new challenges for truth trees. They're like the other semantic atoms – sentence letters – in not being further broken down in a truth tree.

For universal and existential sentences, truth comes by way of instances acting as semantic stand-ins. Based on the truth value of the instances, the quantifier yields a value for the whole quantified sentence. So while the truth value of a quantified sentence does depend on the truth value of a smaller sentence, that smaller sentence is not, technically, a part of that larger whole. (The instance “Ga,” for example, isn't a construction part of the sentence “ $\forall x Gx$ ”.)

That marks something unprecedented in truth trees. For previously truth tree rules ‘un-built’ a sentence the same way a construction tree would (only with truth values added). But in the semantics developed so far for existential and universal sentences don't break such a sentence into its immediate part, but into close relatives of that part: one or more instances.

That new development poses a new challenge: since there's no end to the number of instances we can build for a quantified sentence, truth trees threaten to spin out of control unless the rules for quantified sentences are constrained. To ensure a finite test, truth tree rules will need to focus on just those instances useful either to constructing a counterexample for the argument, or to showing that no such counterexample exists.

1. True Existentials, False Universals. Consider first a **true existential** sentence such as “Something is made of steel,” translated into the formal language as follows.

“Something is made of steel.”

G: __ is made of steel.

$\exists x Gx$

For this sentence to be true there must be at least one true instance of “Gx” – for example, “Ga”.

Instance of Gx: **Ga** (a/x)

Our truth tree rule for true existentials reflects this.

$\checkmark \exists x Gx$	
Ga	

In general: if an existential sentence is true, some instance of it is true.

True Existential (*First Draft*)

$\checkmark \exists x \bullet$	
\bullet_I	

(Here \bullet_I is an instance of \bullet .)

But we need to restrict this rule, to avoid mistaking invalid arguments for valid ones. The following argument, for instance, is glaringly **invalid**.

1. Something is made of steel.

\therefore Rex is made of steel.

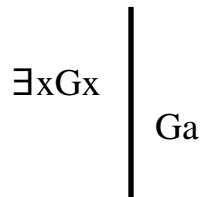
Adapting the previous translation key, we translate like so.

a: Rex **G:** __ is made of steel.

1. $\exists x Gx$

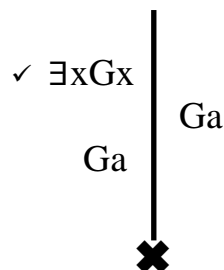
$\therefore Ga$

The tree test begins as usual: premise(s) on the left, conclusion on the right.



But if we now build an instance “Ga,” the tree closes – because the atom “Ga” is on both the left and right of the line.

✂ A Bad Tree ✂



In closing every path, the tree judges the original argument to be **valid** – clearly the **wrong** result.

Our misstep came in using the instance “Ga” – in English, “Rex is made of steel”. The true existential sentence “Something is made of steel” certainly promised us that some-object-or-other is made of steel. But it was a bold and invalid leap to assume that the object in question was Rex.

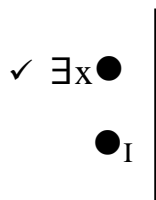
The same sort of mistake is illustrated in the following passage.

Someone broke into the museum between midnight and 2 AM and stole the crown jewels. Let’s call that person “Rex”. So then: Rex broke into the museum and stole the jewels. Quick – grab Rex before he gets away!

While it’s fine to call the thief “Mr. X,” or some other name that no one’s using, the name “Rex” was already taken. So it was far from innocent to use that name for the burglar.

To avoid this sort of error, we insist that with a true existential the name used in its instance can’t **already** be used by something in the model – or, more specifically, on that tree path. For true existential sentences the name used in the instance must be **new** to that path – i.e., a name not appearing previously on that tree path.

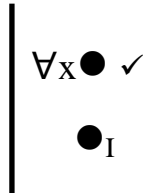
True Existential



(Restriction: \bullet_I must use a new name letter)

False universal sentences present a parallel case. For a universal sentence to be true, all of its instances must be true. So a false universal sentence is sure to have at least one false instance.

False Universal (*First Draft*)



But again we'd better restrict the name in that instance if we don't want the tree test to yield mistaken judgments of validity. The following English argument is clearly **invalid**.

1. The Cathedral of Learning is made of limestone.

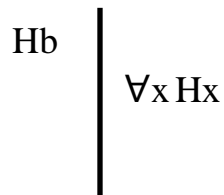
\therefore Everything is made of limestone.

b: The Cathedral of Learning **H:** __ is made of limestone.

1. Hb

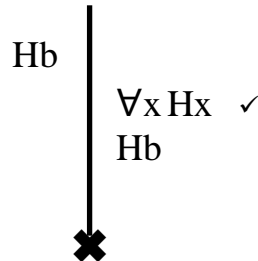
$\therefore \forall x Hx$

Testing the formal argument for validity begins as usual.



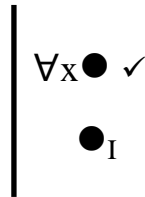
But the false instance “Hb” leaves that atom on both the left and right sides of the path – marking the argument *valid*.

⚡ A Bad Tree ⚡



Again the error lay in allowing the instance to use a name *already* appearing on that tree path. We thus impose the same restriction on the False Universal Rule.

False Universal



(Restriction: \bullet_I must use a new name letter)

2. True Universals, False Existentials: ‘Star Rules’. Compared with the truth tree rule for the false universal, the rule for a **true universal** sentence is simple: a true universal licenses a **true instance for every name used in the model**.

So for this model, featuring name letters “a,” “b,” and “c,” the true universal “ $\forall x Gx$ ” brings in its wake a true instance for each.

\mathbb{D} : {**2, 3, 4**}

a: **2** G: {**2, 3, 4**}

b: **3**

c: **4**

Instances of “ $\forall x Gx$ ”:

Ga: **1** (a/x)

Gb: **1** (b/x)

Gc: **1** (c/x)

That means that, unlike the previous tree rules for quantified sentences, the rule for a true universal **won’t** be finished with just one instance. And while we used a three-name model here for illustration, in building a tree we won’t know in advance how many objects (or names) might be needed for a validity counterexample.

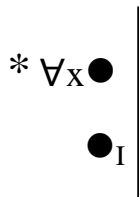
Indeed, in the course of a truth tree for an invalid argument we back our way into the appropriate model – just as, in previous chapters, we backed into a valuation that acted as a counterexample. Instead of beginning with a model (then constructing the appropriate instances), in a truth tree we have the model in hand only once the tree is completed. Yet without a ready-made model to appeal to, we won’t know how many instances are called for.

Still, it would be madness for the tree rule to require **every possible** instance. For with an infinite number of name letters in the formal language, an infinite number of instances are possible for a quantified sentence. A tree rule requiring an infinite number of steps results in a truth tree test that **never** ends, for any (finite) number of steps – and that’s no test at all.

True universals thus pull us in opposite directions: potentially requiring any number of instances, yet also requiring a cap on that number to keep the test from spinning out of control.

We address the first requirement with an addition to our bookkeeping notation. Whereas all previous tree rules **checked** a molecular sentence, and never returned to it, when extracting a true instance from a true universal we instead **star** the universal sentence – thereby noting that we may return to it later for further instances.

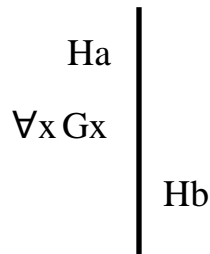
True Universal



We cap the potential explosion of instances with two restrictions.

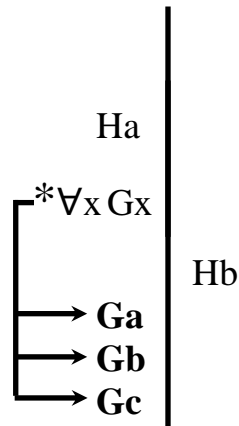
First, so long as there are already one or more name letters on that tree path, the True Universal rule builds instances **only** for those names. That is: the True Universal rule will **never introduce new name letters** (provided there's already a name letter on that path). Whereas we applied the True Existential and False Universal rules only to new name letters, here we do basically the opposite: whenever possible applying the True Universal rule only to **'old' name letters**.

In the following tree, with name letters “a” and “b” already appearing, we build two instances for “ $\forall x Gx$ ”: “Ga” and “Gb”.



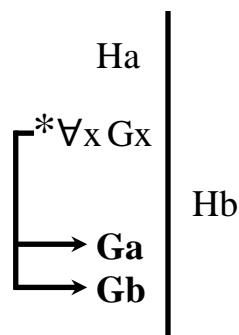
So in the following three applications of the True Universal rule, the third is improper, since it builds an instance for **new** name letter “c”.

☠ A Bad Tree ☠



We will, however, apply the True Universal Rule to ‘old’ (i.e. pre-existing) names **whenever** possible – specifically, to **every** ‘old’ name letter on that path. Since two name letters, “a” and “b,” were already present in that last tree, it would have been equally improper to build **only** “Ga” with the True Universal rule; for in leaving out instance “Gb” we overlook ‘old’ name “b”.

The proper application of the True Universal Rule in this case is as follows.



Second, in applying truth tree rules we push the True Universal rule to the back of the line. That is: we perform such a ‘**star**’ rule (a rule starring its sentence, rather than checking it) **only after** completing all the available ‘**check**’ rules. That way, any name letters to be added by check rules (True Existential, False Universal) will already be on the scene.

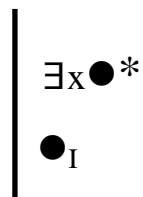
In practice those two restrictions work like so: beginning with a tree that may already contain name letters, we milk out as many more name letters as possible with the check rules (True Existential and False Universal). Only **after** that do we apply the True Universal rule – building an instance for each of the names already found on that path of the tree.

The rule for **false existential** sentences is the mirror image of the True Universal rule. Since an existential sentence is true so long as even one instance is true, it's false only if **every** one of its instances is false.

Again we face a potential infinity of instances (here, false ones): if the sentence “Something is a unicorn” is false, then also false will be the sentences “Letitia is a unicorn,” “Lucretia is a unicorn,” and so on. The consequences are the same as with a true universal. First: no one instance exhausts the potential of a false existential. Second: any attempt to build all possible instances is doomed to infinite regress.

Our solution is the same as well. In recognition of its unlimited potential, a false existential sentence is starred, not checked – leaving open the possibility of revisiting it later for further instances.

False Existential



And once again we apply this star rule only **after** all available ‘check’ rules are used – seeking, whenever possible, to build an instance for (all and) **only** the names already present on that tree path.

A final point completes our treatment of the ‘star’ rules. We said earlier that ‘star’ rules apply only to ‘old’ names **so long as there are already (one or more) name letters on that tree path**. We appreciate the importance of that final proviso (“*so long as...*”) when considering a case where **no** names appear, and **only** ‘star’ sentences remain to be broken down. The following **valid** argument provides an example.

1. Nothing is a unicorn.

\therefore Not everything is a unicorn.

G: __ is a unicorn

1. $\sim \exists x Gx$

$\therefore \sim \forall x Gx$

The truth tree test of validity begins as always; and we can easily dispatch the tildes in “ $\sim \exists x Gx$ ” and “ $\sim \forall x Gx$ ” with the True Negation and False Negation rules.

$\checkmark \sim \exists x Gx$		
		$\sim \forall x Gx \checkmark$
$\forall x Gx$		$\exists x Gx$

But then we hit a wall. Since false existentials and true universals are ‘star’ sentences, we seek to build instances of them using only names **already present** on that tree path. Yet no names are present, and no ‘check’ rules can be applied to supply a name. Counting the tree as finished, however, would wrongly judge the argument invalid.

We need a single name here to break the logjam. So we add: if, in breaking break down a star sentence, there are **no** name letters present on that tree path, nor further check sentences to supply names, we allow the star sentence to introduce **one** new name letter to the tree, to break up the logjam.¹

¹ This move is legitimate in light of our semantics: every model must have at least one object, and every object must have a name. So a true universal or false existential is bound to have at least one instance.

Here, for example, we allow “ $\exists x Gx$ ” to introduce an instance by the False Existential rule. (But then: no further new name letter can be introduced by ‘star’ sentences on this tree path.)

$\checkmark \sim \exists x Gx$		$\sim \forall x Gx \checkmark$
		$\exists x Gx *$
$\forall x Gx$		Ga

The True Universal rule then obligatorily applies “ $\forall x Gx$ ” to “a”.

$\checkmark \sim \exists x Gx$		$\sim \forall x Gx \checkmark$
		$\exists x Gx *$
$* \forall x Gx$		Ga
Ga		

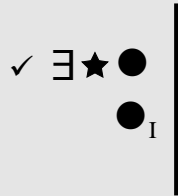
With “Ga” on both sides of the path, the tree closes – correctly judging the argument valid.

$\checkmark \sim \exists x Gx$		$\sim \forall x Gx \checkmark$
		$\exists x Gx *$
$* \forall x Gx$		Ga
Ga		
		\times

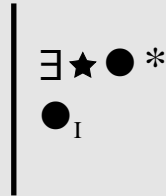
With this last-resort, ‘logjam-breaking’ policy in place, our account of the star rules is complete.

Summary: Quantifier Truth Tree Rules

True Existential

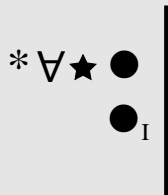


False Existential

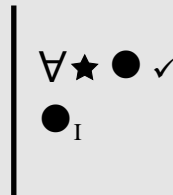


(Restriction: \bullet_I must use a **new** name letter)

True Universal



False Universal



(Restriction: \bullet_I must use a **new** name letter)

- Star rules apply to a path **after** all check rules have been made.
- A star sentence makes instances to all and only name letters **already** on its tree path.

(‘*Unjamming*’ *Exception*: if there are **no** name letters on the tree path, nor any check rules on that path to introduce name letters, a star rule can introduce **one** new name letter.)