Tarski's Idea Illustrated in a Little Language

Document for Linguistics 426 / Philosophy 426

Fall, 2016

Instructor: Richmond Thomason

Version of: September 26, 2016

1. The metalanguage

The metalanguage is English—but English supplemented with variables, and regimented in the ways that mathematicians regiment their language. In particular, since we will be talking about language, we will need variables ranging over expressions. Following the usual practice, we use Greek letters for this. Example: "If ϕ is a formula, so is $\neg\phi$" means that if any expression is a formula, so is the result of appending $\neg$ to it.

2. Syntax of the object language

The object language $\mathcal{L}$ is a version of FOL ("First order logic") designed to talk about a single relation.

Here's the inductive definition of the formulas of $\mathcal{L}$.

Basis: A *term* of $\mathcal{L}$ is any member of the set $\{a, b, x_1, x_2, \ldots\}$.
a and b are *constants*; $x_1, x_2, \ldots$ are *variables*.
R is a (2-place) *predicate letter*.
If α and β are terms, then $R(\alpha, \beta)$ is an (atomic) formula.

Induction: (1) If ϕ is a formula, so is $\neg\phi$;
(2) If ϕ and ψ are formulas, so is $(\phi \rightarrow \psi)$.
(3) If ϕ is a formula and x is a variable, so is $\forall x\phi$;

3. Building a syntactic structure

The above induction not only define the formulas, it shows how to construct them, and it exhibits their structure. The following picture shows how the formula $\forall x_1(R(x_1, a) \rightarrow R(a, x_1))$ can be built up, mechanically following the rules in the definition.

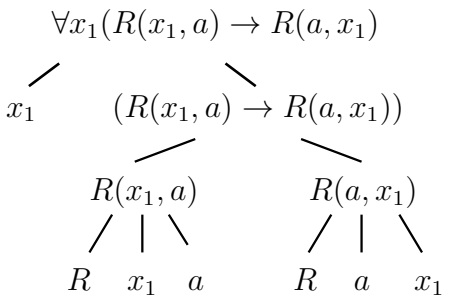


Figure 1

If you look at it from the bottom up, Figure 1 shows how pieces are assembled according to the rules to build up formulas. For instance, in the lower left corner, we have three items.

$R \quad x_1 \quad a$

The basis clause of the inductive definition tells us that x_1 and a are terms, and that $R(x_1, a)$ and $R(a, x_1)$ are (atomic) formulas. Clause (2) of the definition assembles $R(x_1, a)$ and $R(a, x_1)$ into the formula $(R(x_1, a) \rightarrow R(a, x_1))$, and clause (3) assembles the variable x and $R((x_1, a) \rightarrow R(a, x_1))$ into the formula $\forall x_1(R(x_1, a) \rightarrow R(a, x_1))$.

If you look at it from the top down, Figure 1 shows the *constituent structure* of the formula at the top, by showing how it breaks down into components, and, in turn, how these components break down.

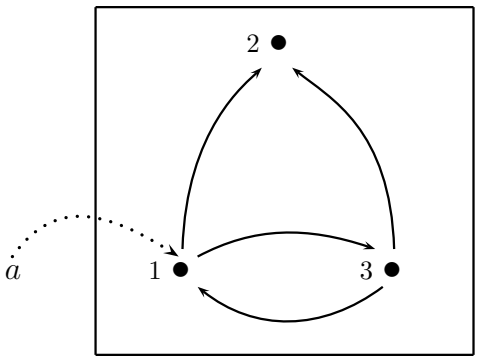
4. Tarski's semantic rules

The central notion of Tarski's semantics is *satisfaction in a model*. This is a generalization of the notion of truth to formulas that have free variables. He defines satisfaction by an induction on syntactic complexity. This means that it's defined first for atomic formulas, and then the satisfaction conditions for complex formulas are characterized in terms of the satisfaction conditions for its syntactic components. In terms of Figure 1, we work up the tree attaching satisfaction conditions to expressions, until we work out the condition for the formula at the top, $\forall x_1(R(x_1, a) \rightarrow R(a, x_1)$.

Tarski interprets forumulas in *relational structures*, which you can think of as little worlds. Since the language $\mathcal{L}$ has a very limited vocabulary—it can only talk about a single two-place relation—we will work with relational structures that only involve one such relation.

There is a simple way to picture such a structure: represent its objects by points, and picture the relation with arrows connecting points. Nothing is left to guesswork; if there is no arrow from one point to another, or to itself, then the relation doesn't hold.

Here is our first structure.



Structure 1

The objects of this structure are 1, 2, and 3. You can think of them as numbers, but it doesn't matter what three things they in fact are.

We'll illustrate how Tarski's approach to semantics works for formulas containing no terms other than a and x_1. The dotted line from a to 1 is a word-world association, showing that that the reference in Structure 1 of the constant a is 1.

First we need to define the *reference* of a term in a structure, relative to an assignment of a value to the variable x_1. If x_1 is assigned the value n (this value may be 1, 2, or 3),

then a refers to 1 and x_1 refers to n. This makes sense—the reference of a constant stays the same, and the reference of a variable varies.

Tarski defines *whether n satisfies a formula* by an induction on the formula's syntactic complexity. The induction begins with atomic formulas, and says that $R(\alpha, \beta)$ is satisfied by Structure 1 if the reference of α is related in the structure to the reference of β. For instance, $R(x_1, a)$ is satisfied when x_1 is assigned 3 (because of the arrow from 3 to 1 in the diagram), and is not satisfied when x_1 is assigned 2 (because there is no arrow from 2 to 1 in the diagram).

The induction continues with clauses for the three types of complex formulas: negations (with $\neg$), conditionals (with $\rightarrow$) and universal quantifications (with $\forall$).

An assignment of n to to the variables x_1 satisfies:
(1) $\neg\phi$ if the assignment doesn't satisfy ϕ;
(2) $(\phi \rightarrow \psi)$ if either the assignment doesn't satisfy ϕ or it satisfies ψ;
(3) $\forall x_1 \phi$ if every assignment of a value 1, 2, or 3 to x_1 satisfies ϕ.

So, for instance, the assignment of 2 to x_1 satisfies $\neg R(x_1, a)$ because it doesn't satisfy $R(x_1, a)$, because there's no arrow from 2 to 1.

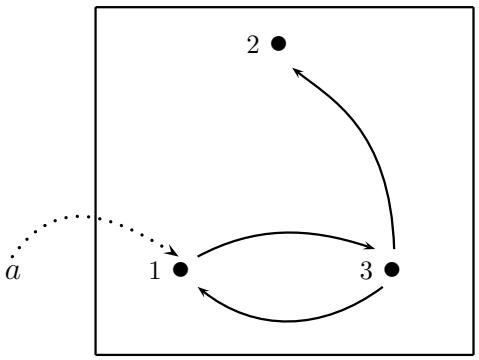
Now, we'll use the rules to work upwards through Figure 1, to determine whether the assigment of 3 to x_1 satisfies the formula at the top, $(R(x_1, a) \rightarrow R(a, x_1))$. We start with the atomic formulas $R(x_1, a)$ and $R(a, x_1)$. The assignment satisfies $R(x_1, a)$ (because there's an arrow from 3 to 1) and it satisfies $R(a, x_1)$ (because of the arrow from 1 to 3).

We now move up the diagram to $(R(x_1, a) \rightarrow R(a, x_1))$. The assignment satisfies this formula because it satisfies $R(a, x_1)$.

But when we come to $\forall x_1(R(x_1, a) \rightarrow R(a, x_1))$ we have to consider *all* assignments of values to x_1. This formula is not satisfied—by the assignment of 3 to x_2 or in fact by any assignment—because if we assign 2 to x_1, $R(x_1, a)$ is satisfied and $R(a, x_1)$ is not satisfied. This means that the conditional $(R(x_1, a) \rightarrow R(a, x_1))$ is *not* satisfied.

Since there is an assignment that doesn't satisfy $(R(x_1, a) \rightarrow R(a, x_1))$, the universal formula $\forall x_1(R(x_1, a) \rightarrow R(a, x_1))$ is not satisfied.

By changing the structure, of course we change the satisfaction conditions for formulas. Removing the link from 1 to 2 in Structure 1 produces the following structure. See if you can work out why now $\forall x_1(R(x_1, a) \rightarrow R(a, x_1))$ is satisfied (by any assignment of a value to x_1.)



Structure 2

5. Type assignments and semantic interpretation

6. Lambda expressions and the logic of types