First-Order Logic (continued)

Satisfaction and Models

Definition:
A formula p is satisfied in a structure $\mathcal{M} = \langle \mathcal{A}, \mathcal{I} \rangle$ by an A-valuation $\rho \in [\mathcal{X} \to A]$ (where $\mathcal{A} = \langle A, \dots \rangle$), if $[\![p]\!]^{\mathcal{M}} \rho = \mathit{true}$.
p is satisfiable in a structure $\mathcal{M} = \langle \mathcal{A}, \mathcal{I} \rangle$ if it is satisfied in $\mathcal{M}$ by some $\rho \in [\mathcal{X} \to A]$.
p is valid in a structure $\mathcal{M} = \langle \mathcal{A}, \mathcal{I} \rangle$ if $[\![p]\!]^{\mathcal{M}} \rho = \mathit{true}$ for all A-valuations $\rho \in [\mathcal{X} \to A]$ where $\mathcal{A} = \langle A, \dots \rangle$. Note that if p is a sentence, i.e, has no free variables, then if p is satisfiable in $\mathcal{M}$, then p is valid in $\mathcal{M}$ since it is satisfied by every assignment ρ'.
We say that a sentence p is satisfiable if p is satisfiable in *some* structure $\mathcal{M}$.
Such a structure is called a model of p.
A formula p is said to be logically valid, also called a tautology, if it is valid in every structure $\mathcal{M} = \langle \mathcal{A}, \mathcal{I} \rangle$ and every A-valuation $\rho \in [\mathcal{X} \to A]$ (with respect to $\mathcal{A} = \langle A, \dots \rangle$).

Since in Logic, everything comes with its converse or with its dual, we also have the following notions.

Definition:
A formula p is falsified in a structure $\mathcal{M} = \langle \mathcal{A}, \mathcal{I} \rangle$ by an A-valuation $\rho \in [\mathcal{X} \to A]$ (where $\mathcal{A} = \langle A, \dots \rangle$), if $[\![p]\!]^{\mathcal{M}} \rho = \mathit{false}$.
p is falsifiable (in a structure $\mathcal{M} = \langle \mathcal{A}, \mathcal{I} \rangle$) if it is falsified in $\mathcal{M}$ by some $\rho \in [\mathcal{X} \to A]$. Such a pair of a structure and a valuation is called a counter-example.
p is invalid or unsatisfiable in a structure $\mathcal{M} = \langle \mathcal{A}, \mathcal{I} \rangle$ if $[\![p]\!]^{\mathcal{M}} \rho = \mathit{false}$ for all A-valuations $\rho \in [\mathcal{X} \to A]$ where $\mathcal{A} = \langle A, \dots \rangle$. Such a structure $\mathcal{M}$ is called a counter-model of p. Note that if p is a sentence, then any counter-example is a counter-model, since its truth value in all valuations is the same.
p is called a contradiction if it is unsatisfiable in *every* structure $\mathcal{M} = \langle \mathcal{A}, \mathcal{I} \rangle$ and every A-valuation $\rho \in [\mathcal{X} \to A]$ (with respect to $\mathcal{A} = \langle A, \dots \rangle$).

Models of sentences

We can also flip the discourse around, and ask which structures satisfy a given set of sentences. As noted above, a first-order structure $\mathcal{M} = \langle \mathcal{A}, \mathcal{I} \rangle$ in which a sentence p is satisfied is called a model of p. The notion extends naturally to a set of sentences Γ: $\mathcal{M} = \langle \mathcal{A}, \mathcal{I} \rangle$ is a model of Γ if $\mathcal{M}$ is a model for *each* sentence p in Γ. Note that here we are talking about sentences, i.e., formulas with *no* free variables. Remember that for sentences, the particular assignment of values from the domain of discourse A does not come into play.

What about formulas which in general may have free variables? While our semantic definitions need this generality, when looking for models, we consider all possible assignments of values in the domain of discourse to the free variables. In other

words, we implicitly take the "universal closure" of a formula. Therefore in practice, we will only need to be concerned about sentences when speaking about models.

Let us look as some examples of how we use sentence to define classes of models.

Example: The structure $\mathcal{M} = \langle \mathcal{N}, \mathcal{I} \rangle$ is a model of the sentence $(\forall x)\ 0 \leq x$. Here $\mathcal{N} = \langle \mathbb{N}, 0_{\mathbb{N}} \rangle$ (the natural numbers with zero being the meaning of the symbol 0), and $\mathcal{I}(\leq)$ is the less-or-equal relation on natural numbers.

Example: Any pre-ordered set where binary symbol R in $\prod$ is interpreted as the ordering relation is a model of both sentences:
$(\forall x)\ R(x,x)$ and $(\forall x)(\forall y)(\forall z)\ R(x,y) \wedge R(y,z) \rightarrow R(x,z)$.

The equality predicate symbol = is usually accorded a special role: it is mapped to any congruence relation, i.e., a relation that is reflexive, transitive and symmetric, and for every function and relation permits the substitution by related (i.e., equal) terms.

Sentences using equality are used to define certain classes of structures.

Example: Consider the language $\mathcal{L}(\Sigma, \Pi, \mathcal{X})$, where $\sum = \{ +^{(2)}, 0^{(0)} \}$ and $\prod = \{ = \}$. A monoid is *any* structure with equality that satisfies the sentences $(\forall x)(\forall y)(\forall z)\ (x + y) + z = x + (y + z)$, $(\forall x)\ x + 0 = x$ and $(\forall x)\ 0 + x = x$.

We next consider the notion of logical equivalence, just as we did in the case of propositions. We recall the semantics of first-order formulas.

Definition: A formula q is a logical consequence of formula p, written $p \Rightarrow q$, if for every structure $\mathcal{M} = \langle \mathcal{A}, \mathcal{I} \rangle$ and every A-valuation $\rho \in [\mathcal{X} \rightarrow A]$, whenever $[\![p]\!]^{\mathcal{M}} \rho = true$ then $[\![q]\!]^{\mathcal{M}} \rho = true$ too. For sentences p, q, this means that if $\mathcal{M}$ is a model for p, *then* $\mathcal{M}$ is also a model for q.

The notion of logical consequence generalises to the case where we have a set of formulas Γ: A formula q is a logical consequence of set of formulas Γ, written $\Gamma \models q$, if for every structure $\mathcal{M} = \langle \mathcal{A}, \mathcal{I} \rangle$ and every A-valuation $\rho \in [\mathcal{X} \rightarrow A]$, whenever for each formula p in Γ we have $[\![p]\!]^{\mathcal{M}} \rho = true$, then $[\![q]\!]^{\mathcal{M}} \rho = true$ too. That is any way to make satisfy each and every formula in Γ is also a way of satisfying q.

Exercise: It is a good idea for you to spend some time looking at the various definitions to understand the difference between the four distinct concepts:

- (Syntax) $p \rightarrow q$, a piece of syntax, where $\rightarrow$ is a connective (i.e., implemented in OCaml as a constructor in a data type for abstract syntax)
- (Boolean function): *implication,* a binary function on the mathematical booleans, i.e., in the function space $\mathbb{B} \times \mathbb{B} \rightarrow \mathbb{B}$ (implemented in OCaml as a function of type `bool * bool -> bool` or `bool -> bool -> bool)`. You may wish to write down its truth table using the values `true` and `false`. Do not use `T` and `F`, which are pieces of syntax (constructors).
- (Logical consequence) $p \Rightarrow q$, a logical concept, where every way (i.e., structure & valuation) of satisfying formula p is a way of satisfying q.

- (Meta-logical) Words such as "if" (or in the reverse direction "only if") used in all the definitions.

Further, try to refrain from writing "$\Rightarrow$" between two expressions you are trying to show are equal when you really should use the mathematical equality symbol "=". It is tempting to write the symbol "$\Rightarrow$" in a mathematical proof when proceeding from one statement to another. This I can understand, though this is "implies" at the meta-logical level. Once you master this distinction of meanings (using colours or using different symbols), we can come back to understanding some fundamental theorems of logic, and how these different concepts are intimately connected.

Note: If the formula $p \to q$ is a tautology then formula q is a logical consequence of formula p, i.e., $p \Rightarrow q$.

You may also complain that we have used the symbol $\to$ both as a logical connective as well as the symbol to denote *function spaces*. This overloading is a pun, which shows itself in a deep connection between logic on the one hand and functional programming on the other. This is called the so-called "Curry-Howard Correspondence", expressed as the slogan: "Propositions as Types, Programs as Proofs".

When two formulas p and q are logical consequences of each other, they are said to be logically equivalent.

Definition: Formulas p and q are said to be logically equivalent, written $p \Leftrightarrow q$, if for every structure $\mathscr{M} = \langle \mathscr{A}, \mathscr{I} \rangle$ and every A-valuation $\rho \in [\mathscr{X} \to A]$ $[\![p]\!]^{\mathscr{M}} \rho = [\![q]\!]^{\mathscr{M}} \rho$. For sentences p, q, this means that *if* $\mathscr{M}$ is a model for p, *then* $\mathscr{M}$ is also a model for q *and conversely too*.

Note very carefully that we have used the notion of mathematical equality in the semantic concept of the boolean values. The notation $p \Leftrightarrow q$ does not mean that p and q are (syntactically) identical. Nor does it mean that p is defined equal to q.

Exercise: Show that if $p \Leftrightarrow p$ for all formulas p.
Exercise: Show that if $p \Leftrightarrow q$ then $q \Leftrightarrow p$, for all formulas p, q.
Exercise: Show that if $p \Leftrightarrow q$ and $q \Leftrightarrow r$ then $p \Leftrightarrow r$, for all formulas p, q, r.

It will require a separate result to show that if two formulas are logically equivalent, then one can replace another in any context within another formula.

Exercise: Try to state this result.

The so-called laws of first-order logic are all instances of logical equivalences.

Exercise: From first principles, i.e., the definition of $[\![p]\!]^{\mathscr{M}} \rho$, show that all the equality laws you have seen for propositional logic define logical equivalences for first-order logic formulas as well.

We now present some logical equivalences specific to first-order logic. As can be expected, these are related to the quantifiers. We can systematically cover all cases by considering how to "swap" connectives at consecutive levels in a formula tree.

The first two are the generalisations of the De Morgan Law
$\neg((\forall x)\ p) \ \Leftrightarrow\ (\exists x)\ (\neg p)$
$\neg((\exists x)\ p) \ \Leftrightarrow\ (\forall x)\ (\neg p)$

Since quantifiers are binders, and the nearest binder governs bound variables, we can dispense with outer quantifiers of the same variable.
$(\mathcal{Q}x)(\forall x)\ p) \ \Leftrightarrow\ (\forall x)\ p$
$(\mathcal{Q}x)(\exists x)\ p) \ \Leftrightarrow\ (\exists x)\ p$

These two rules are in fact special cases of the conditional rule
$(\mathcal{Q}x)\ p \ \Leftrightarrow\ p$ if $x \notin fv(p)$

Similar quantifiers of *different variables* can be swapped — their order is not important.
$(\forall x)(\forall y)\ p \ \Leftrightarrow\ (\forall y)(\forall x)\ p$
$(\exists x)(\exists y)\ p \ \Leftrightarrow\ (\exists y)(\exists x)\ p$

Finally, conjunction commutes with universal quantification, and disjunction with existential quantification.
$((\forall x)\ p)) \wedge ((\forall x)\ q)) \ \Leftrightarrow\ (\forall x)\ (p \wedge q)$
$((\exists x)\ p)) \vee ((\exists x)\ q)) \ \Leftrightarrow\ (\exists x)\ (p \vee q)$

The De Morgan Laws, the commutation of quantification of different variables and the last pair of laws above all indicate that universal quantification behaves as a generalised conjunction while existential quantification as a generalised disjunction.

What about pulling up quantifiers with respect to the other connectives?
These can be done via the following conditional equivalences. Note that in the right hand sides, we have expanded the scope of the quantification. This is safe to do since we are ensuring that no free occurrences of the variable x in the

$((\forall x)\ p)) \wedge q \ \Leftrightarrow\ (\forall x)\ (p \wedge q)$ if $x \notin fv(q)$
$((\exists x)\ p)) \wedge q \ \Leftrightarrow\ (\exists x)\ (p \wedge q)$ if $x \notin fv(q)$

$((\forall x)\ p)) \vee q \ \Leftrightarrow\ (\forall x)\ (p \vee q)$ if $x \notin fv(q)$
$((\exists x)\ p)) \vee q \ \Leftrightarrow\ (\exists x)\ (p \vee q)$ if $x \notin fv(q)$

The above 4 rules have their counterparts when the quantifier appears in the second sub-formula, since conjunction and disjunction are commutative.

$p \wedge ((\forall x)\ q) \ \Leftrightarrow\ (\forall x)\ (p \wedge q)$ if $x \notin fv(p)$

$p \wedge ((\exists x)\ q) \ \Leftrightarrow\ (\exists x)\ (p \wedge q)$ if $x \notin fv(p)$

$p \vee ((\forall x)\ q) \ \Leftrightarrow\ (\forall x)\ (p \vee q)$ if $x \notin fv(p)$
$p \vee ((\exists x)\ q) \ \Leftrightarrow\ (\exists x)\ (p \vee q)$ if $x \notin fv(p)$

Let us now look at the connective $\rightarrow$ and how it behave in such rules.

As expected, if the quantification appears in the second sub-formula, we see the same pattern as above.

$p \rightarrow ((\forall x)\ q) \ \Leftrightarrow\ (\forall x)\ (p \rightarrow q)$ if $x \notin fv(p)$
$p \rightarrow ((\exists x)\ q) \ \Leftrightarrow\ (\exists x)\ (p \rightarrow q)$ if $x \notin fv(p)$

However, when the quantification appears in the first sub-formula, we observe that the quantifier gets dualised. This is because the first sub-formula is in a negative position (and so De Morgan's Laws dualise the quantifier).

$((\forall x)\ p)) \rightarrow q \ \Leftrightarrow\ (\exists x)\ (p \rightarrow q)$ if $x \notin fv(q)$
$((\exists x)\ p)) \rightarrow q \ \Leftrightarrow\ (\forall x)\ (p \rightarrow q)$ if $x \notin fv(q)$

The above rules allow us to state the following result.

Lemma (Prenex Normal Form) Every formula p can be transformed into a logically equivalent formula p' which is in prenex normal form.

The proof uses the above rules, in left to right order, where the quantifier is brought above the other quantifier.

Note that if two different rules apply, in general, we could apply either. So the prenex normal form may not be unique. However, if we choose to follow the rules in any fixed order, then the transformation is deterministic.

Exercise: Give an example of a formula (preferably a sentence) which has two different prenex normal forms.

Exercise: Write a program `prenex: form -> form` to convert a given formula to an equivalent formula in prenex normal form.

What about the swapping of $(\forall x)$ vis-a-vis $(\exists y)$ — for different variables x, y? This is not an equivalence. We can show from first principles that
$(\exists y)(\forall x)\ p \Rightarrow (\forall x)(\exists y)\ p$
However, the reverse direction may not hold.

Exercise: Give a counter-example to show that $(\forall x)(\exists y)\ p \Rightarrow (\exists y)(\forall x)\ p$ does <u>not</u> hold.