# 1 Review

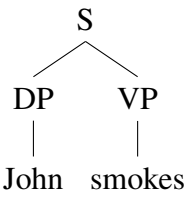
To briefly review the previous lecture, we are constructing a compositional semantic system for the following simple grammar.

(1) a. S → DP VP
b. DP → John | Mary
c. VP → smokes | left

We assume that declarative sentences denote **truth-values**, and proper names denote **individuals**. We also assume that when a DP is made up of a proper name and nothing else, the DP has the same denotation as the proper name, e.g.: for any model $M$, we have the following equivalence.

$$\left[\!\!\left[ \begin{array}{c} \text{DP} \\ | \\ \text{John} \end{array} \right]\!\!\right]^M = [\![\text{John}]\!]^M$$

The **Compositionality Principle** says that the meaning of a complex expression is determined solely by the meanings of its parts and their syntax. Let us apply this to the sentence 'John smokes', which we assume has the syntactic structure in (2):

(2)



The meaning of this sentence is determined by the meaning of [$_{\text{DP}}$ John] and the meaning of [$_{\text{VP}}$ smokes]. You can understand the 'meanings' here to be the denotations, as denotations are a type of meanings (and we want the Compositionality Principle to hold for all kinds of meanings, although we only discuss truth-conditional meanings in this course). So for any model $M$, $[\![(2)]\!]^M$ is determined by the denotations of DP and VP, i.e. $[\![[_{\text{DP}}\ \text{John}]]\!]^M$ and $[\![[_{\text{VP}}\ \text{smokes}]]\!]^M$.

We know what $[\![(2)]\!]^M$ is, namely, a truth-value. It is 1 or 0, depending on what $M$ is. We also know what $[\![[_{\text{DP}}\ \text{John}]]\!]^M$ is, namely some individual. The exact referent is determined by the model. What about $[\![[_{\text{VP}}\ \text{smokes}]]\!]^M$? This is the topic for this lecture.

# 2 The Denotations of Intransitive Verbs

Verbs like 'smokes' that only take one argument are called **intransitive verbs**. Unlike for sentences and proper names, it is not immediately clear what the denotations of intransitive verbs should be. But there is one thing we know about them. According to the compositionality principle, $[\![(2)]\!]^M$ is solely determined by $[\![[_{\text{DP}}\ \text{John}]]\!]^M$ and $[\![[_{\text{VP}}\ \text{smokes}]]\!]^M$ for any model $M$. And we know that $[\![(2)]\!]^M$ is a truth-value (either 0 or 1) and $[\![[_{\text{DP}}\ \text{John}]]\!]^M$ is some individual. So $[\![[_{\text{VP}}\ \text{smokes}]]\!]^M$ is something that can combine with an individual and produce a truth-value!

In order to give a concrete analysis of intransitive verbs, we adopt **Frege's Conjecture**, which states that the only way of combining meanings is **function application**. That is, when you put two meanings together, one of them needs to be a **function** that takes the other meaning as its

**argument**, and the resulting meaning is the value that the function returns for that argument. This is a conjecture and there is no direct empirical evidence that this is correct, but it actually takes us very far, as we will see throughout this course (although we will also see that we might want to have other 'modes of composition' than function application). So let's assume that Frege's Conjecture is correct, and give an analysis to $[\![[_{\text{VP}}\ \text{smokes}]]\!]^M$.

Let us first formulate the compositional rule explicitly as in (3). We call this rule the **Branching Node Rule**.

(3) **Branching Node Rule**

For any model $M$, $\left[\!\!\left[ \begin{array}{c} \text{A} \\ \text{B}\quad\text{C} \end{array} \right]\!\!\right]^M = [\![\text{B}]\!]^M([\![\text{C}]\!]^M)$ or $\left[\!\!\left[ \begin{array}{c} \text{A} \\ \text{B}\quad\text{C} \end{array} \right]\!\!\right]^M = [\![\text{C}]\!]^M([\![\text{B}]\!]^M)$

In words, if B and C are sisters in a tree, the denotation of the subtree $[_{\text{A}}\ \text{B}\ \text{C}]$ is either the denotation of B applied to the denotation of C or the denotation of C applied to the denotation of B, whichever makes sense. Usually, only one of these two possibilities makes sense.

For example, for (2), we know that $[\![[_{\text{DP}}\ \text{John}]]\!]^M$ is not a function but an individual. So $[\![[_{\text{DP}}\ \text{John}]]\!]^M([\![[_{\text{VP}}\ \text{smokes}]]\!]^M)$ doesn't make sense. Therefore, $[\![[_{\text{VP}}\ \text{smokes}]]\!]^M$ must be a function. Furthermore, it must be a function such that $[\![(2)]\!]^M = [\![[_{\text{VP}}\ \text{smokes}]]\!]^M([\![[_{\text{DP}}\ \text{John}]]\!]^M)$.

Now, what exactly is the function $[\![[_{\text{VP}}\ \text{smokes}]]\!]^M$? It should be able to take $[\![[_{\text{DP}}\ \text{John}]]\!]^M$—an individual—as its argument, and it should return $[\![(3)]\!]^M$—a truth-value (0 or 1)—as its output. Since $[\![[_{\text{DP}}\ \text{John}]]\!]^M$ can in principle be any individual in the model, it should be able to combine with any individual in the model and for each of these individuals, it should return a truth-value. So it's a function from individuals to truth-values.

But which truth-value should it return? Recall that we know the truth-condition of the sentence 'John smokes': For any model $M$, the denotation of this sentence should be 1 if the denotation of 'John' in $M$ smokes in $M$, and 0 if not. So we want:

(4) For any model $M$, $[\![[_{\text{DP}}\ \text{John}]]\!]^M([\![[_{\text{VP}}\ \text{smokes}]]\!]^M) = 1$ iff $[\![[_{\text{DP}}\ \text{John}]]\!]^M$ smokes in $M$.

Generalizing this, $[\![[_{\text{VP}}\ \text{smokes}]]\!]^M$ should return 1 when its input is a person who smokes in $M$, and 0 when they do not smoke in $M$.

(5) For any model $M$, $[\![[_{\text{VP}}\ \text{smokes}]]\!]^M$ is the function that takes an individual $x$ in $M$ and returns 1 iff $x$ smokes in $M$.

We take this to be the denotation of $[_{\text{VP}}\ \text{smokes}]$.

As in the case of DPs, we assume that non-branching constituents like '$[_{\text{VP}}$ smokes]' simply inherit the meaning of its sole daughter, so we can also conclude that $[\![[_{\text{VP}}\ \text{smokes}]]\!]^M = [\![\text{smokes}]\!]^M$ for any model $M$. Then, we can state the denotation of the verb as follows.

(6) For any model $M$, $[\![\text{smokes}]\!]^M$ is the function that takes an individual $x$ in $M$ and returns 1 iff $x$ smokes in $M$.

The other intransitive verb in our toy grammar, 'left', can be given essentially the same analysis. That is, its denotation is (7).

(7) For any model $M$, $[\![\text{left}]\!]^M$ is the function that takes an individual $x$ in $M$ and returns 1 iff $x$ left in $M$.

Furthermore, if you want to enrich the grammar by adding more intransitive verbs, e.g. 'yawned',

'ran', etc., it is easy to cook up the denotations for them, using the same recipe.

(8) a. For any model $M$, $[\![\text{yawned}]\!]^M$ is the function that takes an individual $x$ in $M$ and returns 1 iff $x$ yawned in $M$.
b. For any model $M$, $[\![\text{ran}]\!]^M$ is the function that takes an individual $x$ in $M$ and returns 1 iff $x$ ran in $M$.

At this point, let us state the assumption that non-branching constituents simply inherit the denotations of their daughter constituents as a 'compositional rule'. Although there's no real composition going on in such cases, you can see them as a trivial kind of composition. We call this rule the Non-Branching Node Rule.

(9) **Non-Branching Node Rule**
For any model $M$, $\left[\!\!\left[ \begin{array}{c} \text{A} \\ | \\ \text{B} \end{array} \right]\!\!\right]^M = [\![\text{B}]\!]^M$

To sum up, we now have the denotations for all the words and two compositional rules, Branching Node Rule and Non-Branching Node Rule, to derive the denotations of syntactically complex expressions, including sentences. Generally, model-theoretic semantics has these two components, the list of words/morphemes called the **Lexicon** and a set of compositional rules. For our toy grammar, the Lexicon looks like (10).

(10) For any model $M$
a. $[\![\text{John}]\!]^M$ = some individual determined by $M$
b. $[\![\text{Mary}]\!]^M$ = some individual determined by $M$
c. $[\![\text{smokes}]\!]^M$ is the function that takes an individual $x$ in $M$ and returns 1 iff $x$ smokes in $M$
d. $[\![\text{left}]\!]^M$ is the function that takes an individual $x$ in $M$ and returns 1 iff $x$ left in $M$

And the compositional rules are the following two.

(11) **Branching Node Rule**
For any model $M$, $\left[\!\!\left[ \begin{array}{c} \text{A} \\ \diagup\diagdown \\ \text{B}\ \ \text{C} \end{array} \right]\!\!\right]^M = [\![\text{B}]\!]^M([\![\text{C}]\!]^M)$ or $\left[\!\!\left[ \begin{array}{c} \text{A} \\ \diagup\diagdown \\ \text{B}\ \ \text{C} \end{array} \right]\!\!\right]^M = [\![\text{C}]\!]^M([\![\text{B}]\!]^M)$

(12) **Non-Branching Node Rule**
For any model $M$, $\left[\!\!\left[ \begin{array}{c} \text{A} \\ | \\ \text{B} \end{array} \right]\!\!\right]^M = [\![\text{B}]\!]^M$

These twoÂăcompositional rules tell you how to derive the denotations of each grammatical constituent. For example, the Non-Branching Node Rule says,

$$[\![[_{\text{DP}}\ \text{John}\ ]]\!]^M = [\![\text{John}]\!]^M$$

and

$$[\![[_{\text{VP}}\ \text{smokes}\ ]]\!]^M = [\![\text{smokes}]\!]^M$$

And Branching Node Rule says:

$$\begin{aligned} [\![[_{\text{S}}\ [_{\text{DP}}\ \text{John}\ ]\ [_{\text{VP}}\ \text{smokes}\ ]]]\!]^M &= [\![[_{\text{VP}}\ \text{smokes}\ ]]\!]^M([\![[_{\text{DP}}\ \text{John}\ ]]\!]^M) \\ &= [\![\text{smokes}]\!]^M([\![\text{John}]\!]^M) \end{aligned}$$

Similarly, you can compute the meanings of all the other grammatical constituents that our simple syntax produces.

# 3 The Lambda-Notation for Functions

## 3.1 Basics

In formal semantics, it is customary to use the **lambda-notation** for functions, instead of lengthy descriptions in English like 'the function that takes an individual $x$ and returns $1$ iff $x$ smokes in $M$.' Everything we will do in this course could be done without the lambda-notation, but it would be surely too cumbersome to always use plain English to talk about functions. So let us introduce the lambda-notation at this point.

In the lambda-notation, a function looks like:

$$\lambda v\colon\ \phi.\ \alpha$$

- The symbol $\lambda$ (the Greek letter 'lambda') has no meaning. It just says that what follows is a function.
- $v$ is a variable, representing the input of the function. It's a variable because its value varies depending on what the input is. You can pick any variable name here, e.g. $x$, $y$, $X$, etc.
- $\phi$ describes what kind of input the function admits (typically in terms of $v$), meaning it defines the domain of the function. Please keep in mind that $\phi$ needs to be a statement that can be true or false. The idea is that if $\phi$ is true, the input is an appropriate kind of object for the function to operate on, if $\phi$ is false, it is not.
- $\alpha$ describes the output.

It's easier to understand this by looking at concrete examples. Here is one:

(13) $\lambda x\colon\ x$ is a natural number. $x + 5$

This is a function that takes a natural number and adds $5$ to it. You are probably more familiar with the notation $f(x) = x + 5$. What's in (13) is the same function. But the lambda-notation is more convenient than this notation, because it is clear that you are referring to the function itself, rather than the output value of $f$ applied to $x$, which is denoted by $f(x)$. Also, in the lambda-notation, you can explicitly specify the domain (i.e. '$x$ is a natural number'). You can also write (13) as (14). (Recall $\mathbb{N}$ is the set of natural numbers)

(14) $\lambda x\colon\ x \in \mathbb{N}.\ x + 5$

Functions like (14) whose domain restriction is expressed as the variable $x$ being in some set $S$ are often abbreviated to $\lambda x \in S.\ \alpha$. For instance, (14) is often written as $\lambda x \in \mathbb{N}.\ x + 5$. We will adopt this abbreviation in this course, as it is common in the formal semantic literature.

It is important to notice that the choice of the variable $x$ above is not important, precisely because it is a variable. We could have chosen, say, $y$ instead, as in (15), and meant the same thing. So (15) and the above two functions are all the same.

(15) $\lambda y\colon\ y$ is a natural number. $y + 5$

Usually, we use $x, y, z$ as variable names, but in principle they can be any letter. It is, for example, totally legitimate to use a completely arbitrary symbol as in '$\lambda\sqrt{\dagger^{\bullet}_{\Sigma}} \in \mathbb{N}.\ \sqrt{\dagger^{\bullet}_{\Sigma}} + 5$', but if there's no reason to complicate the representation, you'd better use simple symbols for variables.

Functions need not be about mathematical operations. For instance, (16) is a legitimate function (assuming that each dog has a unique name).

(16) $\lambda y$: $y$ is a dog in London. $y$'s name

This is a function that takes any dog in London and returns its name.

Using the lambda-notation, we can re-write the denotations of the intransitive verbs as follows.

(17) For any model $M$,
   a. $[\![\text{smokes}]\!]^M = \lambda x$: $x$ is an individual in $M$. 1 iff $x$ smokes in $M$
   b. $[\![\text{left}]\!]^M = \lambda x$: $x$ is an individual in $M$. 1 iff $x$ left in $M$

If you read published papers in formal semantics, '1 iff' is often omitted, but we keep it here for the sake of explicitness. The set of all individuals in a given model (call the domain of the model) is often written as $D$. Thus, these functions are more compactly written as:

(18) For any model $M$,
   a. $[\![\text{smokes}]\!]^M = \lambda x \in D$. 1 iff $x$ smokes in $M$
   b. $[\![\text{left}]\!]^M = \lambda x \in D$. 1 iff $x$ left in $M$

### 3.2 Functions that return functions

Things get a bit more complicated, when you consider functions that return functions as values. For example, we can define a function $f^+$ that takes a natural number $n$, and returns a function, which in turn take a natural number $m$ and returns the value $n + m$. As you can probably guess, $f^+$ represents the operation of addition. Notice that $f^+(3) \neq f^+(5)$, because $f^+(3)$ is the function that takes a natural number and adds 3 to it, while $f^+(5)$ is the function that takes a natural number and adds 5 to it. In the lambda-notation, $f^+$ can be written as (19). We use '[' and ']' to indicate where the function starts and ends.

(19) $f^+ = [\lambda n \in \mathbb{N}. [\lambda m \in \mathbb{N}. n + m]]$

This function looks like it has two $\lambda$'s, but that's not the right way of understanding it. Recall that in the lambda-notation, each function has three parts. The first two parts are collapsed into $\lambda n \in \mathbb{N}$, which says that it takes any member $n$ of $\mathbb{N}$, which is to say that $n$ is a natural number, as an input. Then the output description specifies what function it turns, namely $[\lambda m \in \mathbb{N}. n + m]$. If $n = 3$, it returns $[\lambda m \in \mathbb{N}. 3 + m]$, if $n = 5$, it returns $[\lambda m \in \mathbb{N}. 5 + m]$. These functions in turn have three parts. They take any member $m$ of $\mathbb{N}$, i.e. any natural number, and returns $3 + m$ and $5 + m$, respectively.

### 3.3 Lambda-conversion

When a function applies to an appropriate argument, you can simplify the representation. This process is called '$\lambda$-conversion' (or $\beta$-reduction, which is perhaps a more mathematically rigorous term). Here is an example of $[\![\text{smokes}]\!]^M$ applied to an individual, John.

$$\begin{aligned}[\![\text{smokes}]\!]^M(\text{John}) &= [\lambda x \in D.\ 1 \text{ iff } x \text{ smokes in } M](\text{John}) \\ &= 1 \text{ iff John smokes in } M\end{aligned}$$

We are following the standard convention that the function comes to the left of the argument and the argument is indicated by '(' and ')'. So things like '(John)$[\lambda x \in D$. 1 iff $x$ smokes in $M]$' and '$(\lambda x \in D$. 1 iff $x$ smokes in $M)$[John]' don't make sense. These expressions are not part of

our metalanguage, and should never be used.

Here are some more examples, illustrating $\lambda$-conversion.

(20) a. $[\lambda x \in \mathbb{N}.\ x + 5](12) = 5 + 12 = 17$

b. $[\lambda x \in \mathbb{N}.\ x + 5]([\lambda y \in \mathbb{N}.\ y^2](3)) = [\lambda x \in \mathbb{N}.\ x + 5](3^2) = [\lambda x \in \mathbb{N}.\ x + 5](9) = 9 + 5 = 14$

Recall that a function can return a function. Consider the following function $f^-$. This function performs subtraction.

(21) $f^- = [\lambda n \in \mathbb{N}.\ [\lambda m \in \mathbb{N}.\ n - m]]$

When one argument, say 3, is given, it returns:

$$f^-(3) = [\lambda n \in \mathbb{N}.\ [\lambda m \in \mathbb{N}.\ n - m]](3) = [\lambda m \in \mathbb{N}.\ 3 - m]$$

Here is an important convention. When two arguments follow a function, the function is applied to the left one (= the one closer to the function) first.

$$f^-(3)(2) = [\lambda n \in \mathbb{N}.\ [\lambda m \in \mathbb{N}.\ n - m]](3)(2) = [\lambda m \in \mathbb{N}.\ 3 - m](2) = 3 - 2 = 1$$

Notice that $f^-(2)(3) = -1$, so $f^-(3)(2) \neq f^-(2)(3)$.

Also remember that square brackets are used to delimit functions. Notice the following inequality:

$$[\lambda n \in \mathbb{N}.\ [\lambda m \in \mathbb{N}.\ n - m]](3)(2) \neq [\lambda n \in \mathbb{N}.\ [\lambda m \in \mathbb{N}.\ n - m](3)](2)$$

The left-hand side is 1, as calculated above. The right-hand side is $-1$, because:

$$[\lambda n \in \mathbb{N}.\ [\lambda m \in \mathbb{N}.\ n - m](3)](2) = [\lambda n \in \mathbb{N}.\ n - 3](2) = 2 - 3 = -1$$

Here, 3 is the argument of the inner function, and 2 is the argument of the whole function. Here we applied the inner function to 3 first, but the order of application does not change the final output.

$$[\lambda n \in \mathbb{N}.\ [\lambda m \in \mathbb{N}.\ n - m](3)](2) = [\lambda m \in \mathbb{N}.\ 2 - m](3) = 2 - 3 = -1$$

## 4 Summary and Computations

To summarise, our semantics consists of the Lexicon (22) (now using the $\lambda$-notation) and compositional rules (23).

(22) For any model $M$

a. $[\![\text{John}]\!]^M$ = some individual determined by $M$

b. $[\![\text{Mary}]\!]^M$ = some individual determined by $M$

c. $[\![\text{smokes}]\!]^M = \lambda x \in D.\ 1$ iff $x$ smokes in $M$

d. $[\![\text{left}]\!]^M = \lambda x \in D.\ 1$ iff $x$ left in $M$

(23) a. **Branching Node Rule**

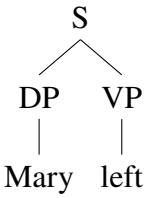
For any model $M$, $\left[\!\!\left[ \begin{array}{c} \text{A} \\ \text{B}\ \ \text{C} \end{array} \right]\!\!\right]^M = [\![\text{B}]\!]^M([\![\text{C}]\!]^M)$ or $\left[\!\!\left[ \begin{array}{c} \text{A} \\ \text{B}\ \ \text{C} \end{array} \right]\!\!\right]^M = [\![\text{C}]\!]^M([\![\text{B}]\!]^M)$

b. **Non-Branching Node Rule**

For any model $M$, $\left[\!\!\left[ \begin{array}{c} \text{A} \\ | \\ \text{B} \end{array} \right]\!\!\right]^M = [\![\text{B}]\!]^M$.

It is important to realize that from these, we can now compute the meanings of any grammatical constituent in the grammar. We can show this concretely by *computing* the meaning. As an example, let us compute the meaning of 'Mary left', which has the structure in (24).

(24)




We compute the meaning of this sentence with respect to a particular model, let's say $M_3$. To simplify, let's assume that in $M_3$, 'Mary' refers to an individual $c$, and also that $c$ left in the situation described by $M_3$. That is, we will use a model $M_3$ such that $[\![\text{Mary}]\!]^{M_3} = c$ and $[\![\text{left}]\!]^{M_3}(c) = 1$. Clearly, in this model, (24) should come out true. And we can verify this by computing the denotation of the sentence as follows.

1. By the Non-Branching Node Rule, $[\![[_{\text{DP}}\ \text{Mary}]]\!]^{M_3} = [\![\text{Mary}]\!]^{M_3}$. Also, since $[\![\text{Mary}]\!]^{M_3} = c$, we have $[\![[_{\text{DP}}\ \text{Mary}]]\!]^{M_3} = c$.
2. Similarly, by the Non-Branching Node Rule, $[\![[_{\text{VP}}\ \text{left}]]\!]^{M_3} = [\![\text{left}]\!]^{M_3}$. Since the Lexicon says that for any model $M$, $[\![\text{left}]\!]^M = \lambda x \in D.\ 1$ iff $x$ left in $M$, $[\![[_{\text{VP}}\ \text{left}]]\!]^{M_3} = \lambda x \in D.\ 1$ iff $x$ left in $M_3$. (Recall that $D$ is the set of all individuals in the model)
3. Finally, according to the Branching Node Rule,

$$[\![(24)]\!]^{M_3} = [\![[_{\text{VP}}\ \text{left}]]\!]^{M_3}([\![[_{\text{DP}}\ \text{Mary}]]\!]^{M_3})$$

From 1. and 2., this is equivalent to:

$$[\lambda x \in D.\ 1 \text{ iff } x \text{ left in } M_3](c)$$

By assumption $c$ is a person who left in $M_3$, so by $\lambda$-conversion, we obtain 1. So the sentence denotes 1 in $M_3$.

We have chosen a particular model $M_3$ here, so we ended up with a particular truth-value. If we didn't know what the model looks like, we can simply end the computation with 1 iff $c$ left in $M_3$, because this statement means that if $c$ left in $M_3$, the denotation of the sentence is 1, if not, it is 0.

In the above computation, we started with the individual words and computed the meanings of more complex expressions by using the compositional rules. Such a computation is called a **bottom-up computation**.

Instead, we can start with the most complex expression and go down the tree towards the terminal nodes, using the compositional rules. That will be a **top-down computation**. Here's

a demonstration for the same sentence and the same model:

$$\begin{aligned}
[\![(24)]\!]^{M_3} &= [\![[_{\text{VP}}\ \text{left}]]\!]^{M_3}([\![[_{\text{DP}}\ \text{Mary}]]\!]^{M_3}) && \text{(BNR)}\\
&= [\![[_{\text{VP}}\ \text{left}]]\!]^{M_3}([\![\text{Mary}]\!]^{M_3}) && \text{(NBNR)}\\
&= [\![[_{\text{VP}}\ \text{left}]]\!]^{M_3}(c) && \text{(by assumption)}\\
&= [\![\text{left}]\!]^{M_3}(c) && \text{(NBNR)}\\
&= [\lambda x \in D.\ 1 \text{ iff } x \text{ left in } M_3](c) && \text{(Lexicon)}\\
&= 1 && (\lambda\text{-conversion})
\end{aligned}$$

In each line, we perform one operation, e.g. the first line decomposes the sentence into DP and VP by the Branching Node Rule, the second line removes the DP-projection by the Non-Branching Node Rule, and so on.

## 5 (Optional) Model Theoretic Semantics

In this optional section, we will explain how model-theoretic semantics works in some more detail.

For the small grammar we have been discussing, a very simple model suffices. In particular, a model $M$ will have two parts, $D_M$ and $V_M$. $D_M$ is called the domain of the model and is the set of individuals in the state of affairs $M$ represents. So $D_M$ is just a set. $V_M$, on the other hand, is meant to tell you what is going on in the model. We can see it as a function of the following kind.

Recall that our grammar generates four simple sentences, e.g. 'John smokes'. In order to know the truth or falsity of this sentence, you need to know (i) who John is and (ii) whether that person smokes. $V_M$ tells you both of these things.

Let us start with (i). If $M$ is a model, then $V_M$ must be a function such that for any proper name $N$, $[\![N]\!]^M = V_M(N)$ such that $V_M(N) \in D_M$. In words, the denotation of a proper name $N$ is $V_M(N)$, which must be some individual in $D_M$. More concretely, let us assume that there are three individuals, $a$, $b$ and $c$. Suppose, furthermore, that there are three models $M_1$, $M_2$ and $M_3$ such that:

(25) a. $[\![\text{John}]\!]^{M_1} = V_{M_1}(\text{John}) = a$
b. $[\![\text{John}]\!]^{M_2} = V_{M_2}(\text{John}) = b$
c. $[\![\text{John}]\!]^{M_3} = V_{M_3}(\text{John}) = c$

This illustrates how the denotation of 'John' varies across situations.

Moving on to (ii), $V_M$ also tells you whether the individual referred to by 'John' smokes in the situation represented by $M$. Specifically, $V_M$ assigns some set to the verb 'smokes'. That is, for any model $M$, we have $V_M(\text{smokes}) \subseteq D_M$. For instance:

(26) a. $V_{M_1}(\text{smokes}) = \{a, b, c\}$
b. $V_{M_2}(\text{smokes}) = \{a, c\}$
c. $V_{M_3}(\text{smokes}) = \{a, b\}$

For example, (26a) means that in $M_1$, $a$, $b$ and $c$ smoke.

In the previous sections we stated the denotation of the verb 'smokes' as follows (where $D$ is actually $D_M$):

(27) $[\![\text{smokes}]\!]^M = \lambda x \in D.\ 1$ iff $x$ smokes in $M$

This is actually a shorthand. What $x$ smokes in $M$ actually means is that $x$ is a member of the set $V_M(\text{smokes})$. That is to say, (27) is a paraphrase of (28).

(28) $\quad [\![\text{smokes}]\!]^M = \lambda x \in D_M.\ 1 \text{ iff } x \in V_M(\text{smokes})$

To summarise, a model $M$ is made up of two things, a set of individuals $D_M$ and a function $V_M$ that assigns values to expressions like proper names and intransitive verbs. Proper names are given individuals as their values, and intransitive verbs are given sets of individuals as their values.