

# Logic 2: Modal Logic

## Lecture 17

---

Wolfgang Schwarz

March 20, 2023

University of Edinburgh

# Modal predicate logic

---

We have added modal operators ( $\Box, \Diamond, O, P, O(\cdot/\cdot), K_i, F, G, \neg, \Box\rightarrow, \dots$ ) to the language of **propositional logic**.

Now we will expand the base language to that of **first-order predicate logic**.

- $\Box Fa$
- $\Diamond \forall x(Fx \rightarrow \Box Gx)$
- $\forall x(Fx \Box\rightarrow K F Gx)$

## Predicate logic: language

---

**Atomic sentences** of  $\mathcal{L}_P$  consist of a predicate followed by a suitable number of terms (names or variables):

- $Fa$
- $Gx$
- $Hxy$
- $Jaxy$

- Bob is sitting.
- $Sb$  ( $b$ : Bob,  $S$ : — is sitting)
- Bob is talking to Carol.
- $Tbc$  ( $b$ : Bob,  $c$ : Carol,  $T$ : — is talking to —)
- Bob is in Rome.
- $Ibr$  ( $r$ : Rome,  $I$ : — is in —)
- Bob is Carol's father.
- $Fbc$  ( $F$ : — is the father of —)
- Bob met Carol's father in Rome.
- $Mbcr$  ( $M$ : — met the father of — in —)

From atomic sentences, we can construct complex sentences with the help of the truth-functional connectives.

- $\neg Sb$
- $(Sb \wedge Tbc)$
- $(Sb \vee Tbc)$
- $((Sb \rightarrow Tbc) \leftrightarrow Fbc)$

We can also construct complex sentences by adding a quantifier in front of a simpler sentence.

A **quantifier** consists of the symbol  $\forall$  or  $\exists$  followed by a variable.

- $\forall x, \forall y, \forall z, \dots$
- $\exists x, \exists y, \exists z, \dots$

So we can say  $\forall xSb, \forall xSx, \exists xSx, \exists x(Sx \wedge Ixr)$ , etc.

Roughly,

$\forall x$  means 'everything/everyone is such that';

$\exists x$  means 'something/someone is such that'.



- Everyone is sitting.
- Everyone is such that they are sitting.
- $\forall xSx$  ( $S$ : — is sitting)
  
- Bob is talking to someone.
- Someone is such that Bob is talking to them.
- $\exists xTbx$  ( $T$ : — is talking to —)

- Everyone is talking to someone.
- Everyone is such that someone is such that they are talking to them.
- Everyone <sub>$x$</sub>  is such that someone <sub>$y$</sub>  is such that they <sub>$x$</sub>  are talking to them <sub>$y$</sub> .
- $\forall x \exists y Txy$  ( $T$ : — is talking to —)
- Everyone is talking to everyone.
- Everyone <sub>$x$</sub>  is such that everyone <sub>$y$</sub>  is such that they <sub>$x$</sub>  are talking to them <sub>$y$</sub> .
- $\forall x \forall y Txy$  ( $T$ : — is talking to —)

Variables  $x, y, z \dots$  function like pronouns ('it', 'they').

Variables are **logical expressions**.

When translating from English, you cannot give a meaning to a variable.

**Wrong:**

- Every tiger is sleeping.
- $\forall xSx$  ( $x$ : tiger,  $S$ : — is sleeping)

- Every tiger is sleeping.
- Everything is such that if it is a tiger then it is sleeping.
- $\forall x(Tx \rightarrow Sx)$  ( $T$ : — is a tiger,  $S$ : — is sleeping)
  
- Some tiger is sleeping.
- Something is such that it is a tiger and it is sleeping.
- $\exists x(Tx \wedge Sx)$
  
- A car drove by.
- Something is such that it is a car and it drove by.
- $\exists x(Cx \wedge Dx)$  ( $C$ : — is a car,  $D$ : — drove by)

### Jargon:

In  $\forall x(Fx \wedge Gy) \rightarrow Gx$ ,

- $\forall x$  **binds**  $x$ ,
- the first two occurrences of  $x$  are **bound**,
- the third is **free**,
- $y$  only has a **free** occurrence.

# Identity

---

## Identity

It is often useful to have a special predicate for identity.

We write  $a = b$  instead of  $= ab$ , and  $a \neq b$  instead of  $\neg a = b$ .

'=' is a logical symbol. It always means '— is (numerically) identical to —'.

## Two classical laws of identity

1. Everything is identical to itself:  $a = a$ .
2. “Leibniz’ Law”: If  $a = b$  then whatever is true of  $a$  is also true of  $b$ .



Leibniz's Law as an inference rule:

$$b = c$$
$$A$$

---

$$A[b//a]$$
$$a = b$$
$$Fa$$

---

$$Fb$$

Leibniz's Law as an inference rule:

$$a = b$$
$$\Phi$$

---

$$\Phi[b//a]$$
$$a = b$$
$$Fa \wedge Rac$$

---

$$Fa \wedge Rbc$$

Leibniz's Law as an inference rule:

$$a = b$$

$$\Phi$$

---

$$\Phi[b//a]$$

$$a = b$$

$$\Box(a = a)$$

---

$$\Box(a = b)$$

Identity is useful not just to express claims about identity.

We can also use it to translate statements involving definite descriptions.

- The Russian president is trustworthy.
- There is a trustworthy Russian president and there is no more than one Russian president.
- $\exists x(Px \wedge Tx \wedge \forall y(Py \rightarrow y=x))$

- The Russian president might have been trustworthy.
- $\Diamond \exists x (Px \wedge \forall y (Py \rightarrow y=x) \wedge Tx)$
- $\exists x (Px \wedge \forall y (Py \rightarrow y=x) \wedge \Diamond Tx)$

Another thing we can (arguably) express with the identity predicate is existence.

- Bob exists.
- Something is such that it is identical to Bob.
- $\exists x(x = b)$ .

A problematic proof:

1.  $b = b$  (Self-Identity)
2.  $\exists x(x = b)$  (Existential Generalisation)
3.  $\Box \exists x(x = b)$  (Necessitation)

## Trees for first-order predicate logic

---

## Trees for first-order predicate logic

Target:  $\forall x \neg Fx \rightarrow \neg \exists x (Fx \wedge Gx)$

1.  $\neg(\forall x \neg Fx \rightarrow \neg \exists x (Fx \wedge Gx))$  (Ass.)



## Trees for first-order predicate logic

Target:  $\forall x \neg Fx \rightarrow \neg \exists x(Fx \wedge Gx)$

1.  $\neg(\forall x \neg Fx \rightarrow \neg \exists x(Fx \wedge Gx))$  (Ass.)
2.  $\forall x \neg Fx$  (1)
3.  $\neg \neg \exists x(Fx \wedge Gx)$  (1)

## Trees for first-order predicate logic

Target:  $\forall x \neg Fx \rightarrow \neg \exists x(Fx \wedge Gx)$

1.  $\neg(\forall x \neg Fx \rightarrow \neg \exists x(Fx \wedge Gx))$  (Ass.)
2.  $\forall x \neg Fx$  (1)
3.  $\neg \neg \exists x(Fx \wedge Gx)$  (1)
4.  $\exists x(Fx \wedge Gx)$  (3)

## Trees for first-order predicate logic

Target:  $\forall x \neg Fx \rightarrow \neg \exists x(Fx \wedge Gx)$

1.  $\neg(\forall x \neg Fx \rightarrow \neg \exists x(Fx \wedge Gx))$  (Ass.)
2.  $\forall x \neg Fx$  (1)
3.  $\neg \neg \exists x(Fx \wedge Gx)$  (1)
4.  $\exists x(Fx \wedge Gx)$  (3)
5.  $Fa \wedge Ga$  (4)

## Trees for first-order predicate logic

Target:  $\forall x \neg Fx \rightarrow \neg \exists x (Fx \wedge Gx)$

1.  $\neg(\forall x \neg Fx \rightarrow \neg \exists x (Fx \wedge Gx))$  (Ass.)
2.  $\forall x \neg Fx$  (1)
3.  $\neg \neg \exists x (Fx \wedge Gx)$  (1)
4.  $\exists x (Fx \wedge Gx)$  (3)
5.  $Fa \wedge Ga$  (4)
6.  $Fa$  (5)
7.  $Ga$  (5)

## Trees for first-order predicate logic

Target:  $\forall x \neg Fx \rightarrow \neg \exists x(Fx \wedge Gx)$

1.  $\neg(\forall x \neg Fx \rightarrow \neg \exists x(Fx \wedge Gx))$  (Ass.)
2.  $\forall x \neg Fx$  (1)
3.  $\neg \neg \exists x(Fx \wedge Gx)$  (1)
4.  $\exists x(Fx \wedge Gx)$  (3)
5.  $Fa \wedge Ga$  (4)
6.  $Fa$  (5)
7.  $Ga$  (5)
8.  $\neg Fa$  (2)  
x

# Trees for first-order predicate logic

 $\forall xA$  $A[c/x]$ 

old or first

 $\exists xA$  $A[c/x]$ 

new

 $\neg\forall xA$  $\neg A[c/x]$ 

new

 $\neg\exists xA$  $\neg A[c/x]$ 

old or first

Self-Identity:

 $c = c$ 

old

Leibniz' Law:

 $b = c$  $A$  $A[c//b]$

## Trees for first-order predicate logic

Target:  $\forall x\forall y((Rxy \wedge x=y) \rightarrow Rxx)$

1.  $\neg\forall x\forall y((Rxy \wedge x=y) \rightarrow Rxx)$  (Ass.)

## Trees for first-order predicate logic

Target:  $\forall x\forall y((Rxy \wedge x=y) \rightarrow Rxx)$

1.  $\neg\forall x\forall y((Rxy \wedge x=y) \rightarrow Rxx)$  (Ass.)

2.  $\neg\forall y((Ray \wedge a=y) \rightarrow Raa)$  (1)



## Trees for first-order predicate logic

Target:  $\forall x\forall y((Rxy \wedge x=y) \rightarrow Rxx)$

1.  $\neg\forall x\forall y((Rxy \wedge x=y) \rightarrow Rxx)$  (Ass.)
2.  $\neg\forall y((Ray \wedge a=y) \rightarrow Raa)$  (1)
3.  $\neg((Rab \wedge a=b) \rightarrow Raa)$  (2)

## Trees for first-order predicate logic

Target:  $\forall x\forall y((Rxy \wedge x=y) \rightarrow Rxx)$

1.  $\neg\forall x\forall y((Rxy \wedge x=y) \rightarrow Rxx)$  (Ass.)
2.  $\neg\forall y((Ray \wedge a=y) \rightarrow Raa)$  (1)
3.  $\neg((Rab \wedge a=b) \rightarrow Raa)$  (2)
4.  $Rab \wedge a=b$  (3)
5.  $\neg Raa$  (3)

## Trees for first-order predicate logic

Target:  $\forall x\forall y((Rxy \wedge x=y) \rightarrow Rxx)$

1.  $\neg\forall x\forall y((Rxy \wedge x=y) \rightarrow Rxx)$  (Ass.)
2.  $\neg\forall y((Ray \wedge a=y) \rightarrow Raa)$  (1)
3.  $\neg((Rab \wedge a=b) \rightarrow Raa)$  (2)
4.  $Rab \wedge a=b$  (3)
5.  $\neg Raa$  (3)
6.  $Rab$  (4)
7.  $a=b$  (4)

## Trees for first-order predicate logic

Target:  $\forall x\forall y((Rxy \wedge x=y) \rightarrow Rxx)$

1.  $\neg\forall x\forall y((Rxy \wedge x=y) \rightarrow Rxx)$  (Ass.)
2.  $\neg\forall y((Ray \wedge a=y) \rightarrow Raa)$  (1)
3.  $\neg((Rab \wedge a=b) \rightarrow Raa)$  (2)
4.  $Rab \wedge a=b$  (3)
5.  $\neg Raa$  (3)
6.  $Rab$  (4)
7.  $a=b$  (4)
8.  $Raa$  (6,7,LL)  
x