- Interpret predicates and constants of the basic alphabet $\mathcal{S}$ through relations and distinguished elements of the structure:

$$\mathcal{S} = \{On(\cdot,\cdot), LeftOf(\cdot,\cdot), Color(\cdot,\cdot), Block(\cdot), a, b, c, d, e, azul, rojo, ..., piso\}$$

  - $a \overset{\mathfrak{B}}{\mapsto} A$, $b \overset{\mathfrak{B}}{\mapsto} B$, $\ldots$, $verde \overset{\mathfrak{B}}{\mapsto} green$, $piso \overset{\mathfrak{B}}{\mapsto} floor, \ldots$

    Constants interpreted as distinguished elements of structure $\mathfrak{B}$

  - $Block \overset{\mathfrak{B}}{\mapsto} Block^{\mathcal{B}}$, $Color \overset{\mathfrak{B}}{\mapsto} Color^{\mathcal{B}}$, $\ldots$ $\quad (= \overset{\mathfrak{B}}{\mapsto} =^{\mathcal{B}})$

    Unary predicates interpreted as unary relations on the domain; two-ary (binary) predicates as binary relations on the domain, etc.

- $\mathfrak{B}$ is an interpretation structure for (or compatible with) language $L(\mathcal{S})$

- Notice that the names in the symbolic language do not have to coincide with the names in the structure (but see page 18)
(showing this was the reason for choosing the "strange" alphabet $\mathcal{S}$)

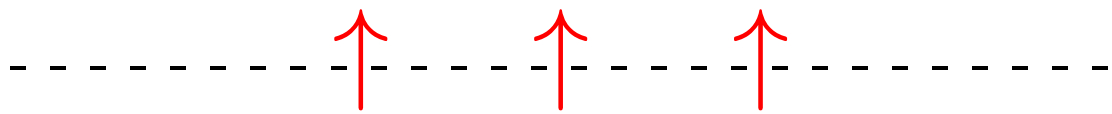
- We have now define when a formula is true in a compatible interpretation structure
- In general:

Meta-level, structural, interpretation level

$$\mathfrak{B} = \langle \mathcal{B}, Block^{\mathcal{B}}, On^{\mathcal{B}}, Color^{\mathcal{B}}, LeftOf^{\mathcal{B}}, A, B, C, D, E, green, \ldots \rangle$$

"there are $e_1, e_2 \in \mathcal{B}$ such that $(e_1, e_2) \in On^{\mathcal{B}}$ and $(e_2, red) \in Color^{\mathcal{B}}$"

Statement in the meta-language of usual Math



$$\exists x \exists y (On(x, y) \wedge Color(y, rojo))$$

Statement in the object language

Symbolic, formal, object level

$$\mathcal{S} = \{On(\cdot,\cdot), LeftOf(\cdot,\cdot), Color(\cdot,\cdot), Block(\cdot), a, b, c, d, e, azul, rojo, ..., piso\}$$

- This sentence should be true once interpreted above

  $\mathfrak{B}$ should make $\exists x \exists y (On(x, y) \wedge Color(y, rojo))$ true

  Denoted: $\mathfrak{B} \models \exists x \exists y (On(x, y) \wedge Color(y, rojo))$

- We can give a general definition of truth of formulas under interpretation structures
- We will restrict ourselves to a particular class of interpretation structures

  They are simple and naturally appear in RDBs and their extensions
- We will consider "syntactic" structures whose elements belong to the object language (a.k.a. Herbrand structures)

  So as in RDBs: the data items and predicates appear both in the schema and in the RDB itself as an interpretation structure

  So, we know this and have used this without being aware of it

Example: Given the language $L(\mathcal{S})$ above, consider the structure

$$\mathcal{D} = \langle Dom, Block, On, Color, LeftOf, a, b, \ldots \rangle$$

- Domain Dom contains the set of constants of $L(\mathcal{S})$:

  $$Dom = \{a, b, c, d, e, azul, rojo, ..., piso, \ldots\}$$

  The constants of $\mathcal{S}$ are then interpreted by themselves

- Predicates are interpreted by relations (represented as tables):

Block	
	a
	b
	c
	d
	e

On		
	c	*b*
	b	*a*
	a	*piso*
	d	*piso*
	e	*piso*

Color		
	a	*rojo*
	b	*verde*
	c	*amarillo*
	d	*purpura*
	e	*azul*

LeftOf		
	a	*d*
	d	*e*

  That is, $Block^{\mathcal{D}} = \{a, b, c, d, e\}, \ldots,\ LeftOf^{\mathcal{D}} = \{(a, d), (d, e)\}$

- A "syntactic" structure, at the level of the symbolic language

  Perfectly O.K. as set-theoretic structure

- The DB becomes an interpretation structure for the language determined by its schema

  A relational model of our blocks world

- Usually Dom is implicit, and this structure (the DB) is represented as:

$$\mathcal{D} = \{Block(a), Block(b), Block(c), Block(d), Block(e), On(c, b), On(b, a), On(a, piso), On(d, piso), On(e, piso), Color(a, rojo), Color(b, verde), Color(c, amarillo), Color(d, purpura), Color(e, azul), LeftOf(a, d), LeftOf(d, e)\} \quad (*)$$

- It contains all the "true" atomic sentences of the language, and only them (still to be formalized)
- We want to define when a formula $\varphi \in L(\mathcal{S})$ is true in $\mathcal{D}$
- But what about formulas with free variables, e.g. $\exists x On(x, y)$?

  Its truth (or not) should depend on the value y takes in Dom
- We need, in addition to $\mathcal{D}$, an assignment

$$\theta : \{u, v, w, x, y, z, \ldots\} \to Dom$$

  A function that maps variables to domain elements
- With $\theta_1 : y \mapsto a$ the formula should be true, but not with $\theta_2 : y \mapsto c$

- We will give an inductive definition of "a formula φ being true in structure $\mathcal{D}$ under assignment θ"

  To be denoted with: $\mathcal{D} \models \varphi[\theta]$

- We mean "induction on the structure of a formula"

  Following the syntactic construction formula rules on page 8

- $\varphi[\theta]$ denotes formula φ with its free variables replaced by the domain elements according to θ

  E.g. $\exists x On(x, y)[\theta_1]$ becomes $\exists x On(x, a)$

- We expect $\mathcal{D} \models \exists x On(x, y)[\theta_1]$ to hold

- Since we do not care about the values assigned to variables that are not free in φ, we usually write $\mathcal{D} \models \exists x On(x, y)[a]$
- Let's proceed ...

- Definition of $\mathcal{D} \models \varphi[\theta]$ by induction on legal formulas φ

We assume the structure is represented as in (*) on page 19

1. φ is atomic formula, but not an equality

$$\mathcal{D} \models \varphi[\theta] :\Longleftrightarrow \varphi[\theta] \in \mathcal{D}$$

Definition of LHS in terms of RHS, no need for ":", just to emphasize

Also: "$\Longleftrightarrow$" is the usual mathematical symbol of the meta-language

Examples:

$\mathcal{D} \models On(b, a)$ (no need for θ, formula is "ground", no variables)

$\mathcal{D} \not\models On(d, b)$ (meaning "it is not the case that $\mathcal{D} \models On(d, b)$")

$\mathcal{D} \models On(x, a)[b]$

$\mathcal{D} \not\models On(x, a)[c]$

Remark: This case captures the "closed-world assumption" (CWA) of RDBs:

A positive fact is true if an only it is explicitly represented in the DB, in a table

2. For t_1, t_2 in *Dom* or variables (i.e. terms):

   $\mathcal{D} \models (t_1 = t_2)[\theta] \iff t_1[\theta]$ and $t_2[\theta]$ are (syntactically) identical

   Examples:

   $\mathcal{D} \models a = a$ $\quad\quad$ $\mathcal{D} \not\models a = b$

   $\mathcal{D} \models (x = a)[a]$ $\quad\quad$ $\mathcal{D} \not\models (x = b)[c]$

   Remark: This captures the "unique-names assumption" (UNA) of RDBs:

   Two different names (constants) in a RDB denote two different objects; e.g. `john` and `peter`, being syntactically different, are treated as different by the DB

   That is, "`john` = `peter`" is never true in a RDB

3. If φ is a (legal) formula:

$$\mathcal{D} \models \neg\varphi[\theta] \quad :\Longleftrightarrow \quad \mathcal{D} \not\models \varphi[\theta]$$

Examples:

$\mathcal{D} \models \neg On(d, b)$, because $\mathcal{D} \not\models On(d, b)$

(equivalently due to CWA: $On(d, b) \notin \mathcal{D}$)

$\mathcal{D} \models \neg a = b$ (denoted $\mathcal{D} \models a \neq b$), because $\mathcal{D} \not\models a = b$

4. If φ, ψ are formulas:

$\mathcal{D} \models (\varphi \wedge \psi)[\theta] \quad :\Longleftrightarrow \quad \mathcal{D} \models \varphi[\theta]$ and $\mathcal{D} \models \psi[\theta]$

$\mathcal{D} \models (\varphi \vee \psi)[\theta] \quad :\Longleftrightarrow \quad \mathcal{D} \models \varphi[\theta]$ or $\mathcal{D} \models \psi[\theta]$

$\mathcal{D} \models \underbrace{(\varphi \rightarrow \psi)}_{\text{object language}}[\theta] \quad :\Longleftrightarrow$ If $\mathcal{D} \models \varphi[\theta]$, then $\mathcal{D} \models \psi[\theta]$

(equivalently $\underbrace{\mathcal{D} \models \varphi[\theta] \implies \mathcal{D} \models \psi[\theta]}_{\text{meta-language}}$)

- Here we are assigning their intended meaning to propositional connectives (the same as in propositional logic)

- Notice how we recursively pass from a possible complex formula on the LHS to simpler subformulas on the RHS

5. If φ is formula, and x a variable:

   $$\mathcal{D} \models \forall x\varphi[\theta] \iff \mathcal{D} \models \varphi[\theta\frac{x}{s}], \text{ for every } s \in Dom$$

   Here, $\theta\frac{x}{s}$ is the assignment that coincides with θ on every variable, possibly except for x that is assigned the value s

   Example: $\mathcal{D} \models \forall x \neg On(piso, x)$ (θ irrelevant; no free variables on LHS)

   In fact: $\mathcal{D} \models \neg On(piso, x)[s]$, for every $s \in Dom$

6. If φ is formula, and x a variable:

   $$\mathcal{D} \models \exists x\varphi[\theta] \iff \mathcal{D} \models \varphi[\theta\frac{x}{s}], \text{ for some } s \in Dom$$

   Example: $\mathcal{D} \models \exists x(Block(x) \wedge \neg On(x, piso))$

   In fact: $\mathcal{D} \models (Block(x) \wedge \neg On(x, piso))[b]$, with $b \in Dom$

- Here we are assigning meaning (semantics) to quantifiers

- The definition of truth is compositional in the sense that the truth of a formula is determined by the truth of its subformulas

  Something we ~~see already~~ clearly see in propositional logic, through the use of truth tables

- The inductive nature of this definition enables a recursive procedure to evaluate the truth of a formula in a database

- This compositional evaluation of formulas (queries in RDBs) is at the basis of SQL

  The standard query language for RDBs

Example: *"There is a block that has no other block on top"*

- $\exists x(Block(x) \wedge \forall y(Block(y) \rightarrow \neg On(y,x)))$
- $\mathcal{D} \models \exists x(Block(x) \wedge \forall y(Block(y) \rightarrow \neg On(y,x)))$?

1. Is there an $s \in Dom$, such that:
   $\mathcal{D} \models (Block(x) \wedge \forall y(Block(y) \rightarrow \neg On(y,x)))[s]$?
2. Is there a $s \in Dom$, such that:

$\mathcal{D} \models Block(x)[s]$ and $\mathcal{D} \models \forall y(Block(y) \rightarrow \neg On(y,x)))[s]$?

3. So, in $\mathcal{D}$: Is there a $s \in Dom$, such that $Block(s)$ and for every $t \in Dom$: $Block(t) \in \mathcal{D} \Rightarrow On(t,s) \notin \mathcal{D}$?

   Clearly this true in $\mathcal{D}$ for $s = c$:
   $Block(c) \in \mathcal{D}$ and, for every $t \in Dom$: $Block(t) \in \mathcal{D} \Rightarrow On(t,c) \notin \mathcal{D}$

   (also for values d and e for s)

- So, the answer is *Yes!*
- This is something a RDB actually does for query and IC evaluation!

Exercise: Are the following sentences true or false in the structure (database) $\mathcal{D}$ that models the blocks world?

Decide intuitively, but also using the inductive definition of truth

1. $\exists x(Block(x) \wedge Colour(x, azul))$
2. $\forall x \neg \exists y\, On(y, x)$
3. $\neg \forall x \neg \exists y\, On(y, x)$
4. $\exists x \exists y \neg On(y, x)$
5. $\forall x(\exists y\, On(x, y) \rightarrow Block(x))$
6. $\exists y(Block(y) \wedge \neg \exists z(Block(z) \wedge LeftOf(y, z))$
7. $\forall x \forall y(Colour(x, z) \wedge Colour(y, z) \rightarrow x = y)$