Syntax and Semantic Systems

.... Notes 3IA3

Additional definitions and notes

$\mathfrak{B} \subset \mathfrak{F}$ is inconsistent if for any $F \in \mathfrak{F}$: $\mathfrak{B}$ |- F
In other case, $\mathfrak{B}$ is consistent.
$\mathfrak{B}$ is decidable if it is consistent and for each F
either $\mathfrak{B}$ |- F or $\mathfrak{B}$ |- $\neg$F

Instead of $\mathfrak{B} \cup \{G\}$ |- F we write $\mathfrak{B}$, G |- F
Instead of {G1, ..., Gn} |- F we write G1, ...,Gn |- F

Propositional Calculus

…………… Notes 31A3

Def. Boolean function of implication -> and that of negation $\bar{\ }$

x	y	x->y
0	0	1
0	1	1
1	0	0
1	1	1

x	$\bar{x}$
0	1
1	0

Def. We say that $t\in\mathfrak{I}$ interprets implication, negation if

$$t(x \Rightarrow y) = t(x) \rightarrow t(y) , \qquad t(\neg x) = \overline{t(x)}$$

Def. Semantic system of ProC $[\mathfrak{I}, \mathfrak{I}_{ProC}]$:

$\mathfrak{I}_{ProC} = \{ t \mid t: \mathfrak{I} \rightarrow \{0,1\}$ & t interprets implication and negation$\}$

Def. Interpretation $t \in \mathfrak{I}_{ProC}$ is a model of formula $F\in\mathfrak{I}$ if $t(F)=1$

Def. Formula F is (semantically) satisfiable if it has a model in $\mathfrak{I}_{ProC}$

Lemma. (Reflexivity of implication) |- F => F

Proof.

|- F => ((F=>F) => F)	A1
|- (F => ((F=>F) => F) => ((F => (F=>F)) => (F=>F))	A2
|- (F => (F=>F)) => (F=>F)	MP
|- F => (F=>F)	A1
|- F => F	MP

Semantically:

t(F)	t(F)->t(F)
0	1
1	1

i.e., |= F

Lemma. Deduction. Let $\mathfrak{B} \subset \mathfrak{F}$, F, G $\in \mathfrak{F}$. Then

$\mathfrak{B}$, F |- G iff $\mathfrak{B}$ |- F => G

Remark:

F1, F2, ..., Fn |- G iff |- F1 => (F2 => ... (Fn => G)...)

Proof. 1) Let $\mathfrak{B}$ |- F=>G . Then we extend the proof by F , and MP will yield $\mathfrak{B}$, F |- G

2) Let $\mathfrak{B}$, F |- G . Therefore, there is a proof C1,...,Cn = G from premises $\mathfrak{B}$ and F . We proof by induction that $\mathfrak{B}$ |- F => Ci , i=1,...,n

.... MFF, ProC. p.10

Additional propositional functors

Def. Disjunction	F V G	is defined as	¬F => G
Conjunction	F & G	is defined as	¬ (F => ¬G)
Equivalence	F <=>G	is defined as	(F=>G) & (G=>F)

Remark. Interpretations of the semantic system interpret the above functors:

t(F&G) = t(F) . t(G) ,   t(F V G) = t(F) + t(G) ,
t(F<=>G) = t(F) ≡ t(G)

where . is boolean product, + boolean sum, ≡ boolean equivalence:

x	y	x.y	x+y	x≡y
0	0	0	0	1
0	1	0	1	0
1	0	0	1	0
1	1	1	1	1

Priority of the functors:	¬	(∀x)	(∃x)
	&		
	V		
	=>	<=>	

Lemma 13.

1) |- (F=>G) => ((G=>H) => (F=>H))
|- (F=>G) & (G=>H) => (F=>H)   transitivity of implication
2) |- ¬F => (F=>G)   1st paradox of implication
3) |- F => ¬¬F   double negation
|- ¬¬F => F
|- F <=> ¬¬F
5) |- (F=>G) => (¬G => ¬F)   transposition
|- (F=>G) <=> (¬G => ¬F)
6) |- F => (¬G => ¬(F=>G))

7) $\vdash (\neg F \Rightarrow F) \Rightarrow F$ Clavius law
$\vdash (F \Rightarrow \neg F) \Rightarrow \neg F$

8) $\vdash (F \Rightarrow (G \,\&\, \neg G)) \Rightarrow \neg F$
$\vdash (F \Rightarrow G) \,\&\, (F \Rightarrow \neg G)) \Rightarrow \neg F$ proof by contradiction(see later)

Proof.

………………………… MFF, ProC, p11-12

Lemma. (Proof by two contradict cases). If 𝔅, F |- G and 𝔅, ¬F |- G then 𝔅 |- G

Proof.

𝔅 |- F=>G	assumptions
𝔅 |- ¬F=>G	
𝔅 |- ¬G => ¬F	lemma 13, 5), MP
𝔅 |- (¬G => ¬F) => ((¬F => G) => (¬G => G))	lemma 13, 1)
𝔅 |- ¬G => G	MP
𝔅 |- G	lemma 13, 7)

Lemma. (Proof by contradiction). If 𝔅, F |- G and 𝔅, F |- ¬G , then 𝔅 |- ¬F

Proof. Following assumptions and

|- G => (¬G => (G & ¬G))	lemma 19, 3)
𝔅, F |- G & ¬G	assumptions, MP
𝔅 |- F => (G & ¬G)	deduction lemma
𝔅 |- ¬F	lemma 13, 8)

Lemma 19.

1) |- (F & G) => F   implication for conjunction
2) |- (F & G) => G
3) |- F => (G => F&G)   or: F, G |- F&G
4) |- F&G => G&F
|- F&G <=> G&F   commutative law
5) |- F => (F V G)   implication for disjunction
6) |- G => (F V G)   or: F |- F V G , G |- F V G
7) |- (F=>H) => ((G=>H) => (F V G => H))
or: F=>H, G=>H |- F V G => H
or-elimination
8) |- F V G => G V F
|- F V G <=> G V F   commutative law
9) |- (F <=> G) => (F => G)   from equivalence to implication
10) |- (F <=> G) => (G => F)
11) |- (F=>G) => ((G=>F) => (F<=>G)
12) |- (F<=>G) => (G<=>F)
|- (F<=>G) <=> (G<=>F)   commutative law
13) |- (F<=>G) => (¬F <=> ¬G)
|- (F<=>G) <=> (¬F <=> ¬G)
14) |- F V ¬F   true and false
|- ¬(F & ¬F)
16) |- ¬(F & G) <=> ¬F V ¬G   de Morgan laws
|- ¬(F V G) <=> ¬F & ¬G
17) |- F & (G V H) <=> (F&G V F&H)   distributive laws
|- F V (G&H) <=> ((F V G) & (F V H))
23) |- (F => G) <=> (¬F V G)
|- ¬(F=>G) <=> F & ¬G
|- ¬(F&G) <=> (F => ¬G)
24) |- F <=> F   reflexivity of equivalence
|- (F<=>G) <=> ((F<=>H)<=>(H<=>G))   transitivity of equiv.
25) |- (F => (G => H)) <=> (F&G => H)   compound implication

Commutative laws:

|- F&G <=> G&F

|- F V G <=> G V F

|- (F<=>G) <=> (G<=>F)

Associative laws:

|- ((F&G) & H) <=> (F & (G&H))

|- ((F V G) V H) <=> (F V (G V H))

|- ((F<=>G) <=> H) <=> (F <=> (G<=>H))

Proof.

.... MFF, ProC, p.21-22

Def. Let q1,...,qn be prop. variables, G1,...,Gn formulas, then $Sb^{q1,\dots,qn}_{G1,\dots,Gn}$ (F) is defined as follows:

a) $Sb^{q1,\dots,qn}_{G1,\dots,Gn}$ (p) = Gi if for some i=1,...,n p=qi
p otherwise

b) $Sb^{q1,\dots,qn}_{G1,\dots,Gn}$ (¬F) = ¬$Sb^{q1,\dots,qn}_{G1,\dots,Gn}$ (F)

c) $Sb^{q1,\dots,qn}_{G1,\dots,Gn}$ (F=>G) = $Sb^{q1,\dots,qn}_{G1,\dots,Gn}$ (F) => $Sb^{q1,\dots,qn}_{G1,\dots,Gn}$ (G)

Theorem. Let q1,...,qn be prop. variables, G1,...,Gn formulas, then for any F :
if |- F then |- $Sb^{q1,\dots,qn}_{G1,\dots,Gn}$ (F)

Remark. If $\mathcal{B}$ |- F then $Sb^{q1,\dots,qn}_{G1,\dots,Gn}$ ($\mathcal{B}$) |- $Sb^{q1,\dots,qn}_{G1,\dots,Gn}$ (F)
But not: |- F => $Sb^{q1,\dots,qn}_{G1,\dots,Gn}$ (F)

Theorem. (Extensionality). Let q1,...,qn be prop. variables, G1,...,Gn, G1',...,Gn' formulas. Then for each F

G1<=>G1', ..., Gn<=>Gn' |- $Sb^{q1,\dots,qn}_{G1,\dots,Gn}$ (F) <=> $Sb^{q1,\dots,qn}_{G1',\dots,Gn'}$ (F)

Proof.

.... MFF, ProC, p.24

Theorem. Combined system of ProC is complete, i.e., for each ℬ, F

ℬ |- F iff ℬ |= F

Proof (at least from left to right).

a) Proof that |= A1 (F => (G=>F)

t(F)	t(G)	t(G) -> t(F)	t(F) -> (t(G)->t(F))
0	0	1	1
0	1	0	1
1	0	1	1
1	1	1	1

Similarly |= A2, |= A3

b) Proof F, F=>G |= G, i.e., proof that for every t :

if t(F)=1 and t(F=>G) =1 then also t(G) = 1

t(F)	t(G)	t(F=>G) = t(F) -> t(G)	
0	0	1	
0	1	1	
1	0	0	
1	1	1	<<< model for F, f=>G is also model for G

Example. Prove syntactically/ semantically:

|- (F => (G => H)) <=> (F&G => H)
|= (F => (G => H)) <=> (F&G => H)

…………………… ProC, A

Example. Prove syntactically/ semantically

|- p=>r, r=>s |- p&r => q&s
|- p=>r, r=>s |= p&r => q&s

…………………… ProC, B

Well-defined formula

Hierarchical tree

.... Book, p.33-34

Literal

Def. Literal is either prop. variable (atom) p or the negation of prop. variable (atom) $\neg p$.

Formula F is in conjunctive normal form (ConjNF) if it is conjunction of disjunction of literals, i.e, corresponding grammar (rewrite) rules:

F ::= D | D & F
D ::= L | L V D
L ::= <prop.variable> | ¬ <prop.variable>

Formula F is in disjunctive normal form (DisjNF) if it is disjunction of conjunctions of literals.

Lemma 1.43. A disjunction of literals $L1 \vee \dots \vee Ln$ is semantically valid (tautology) iff there are $1 \le i,j \le n$ such that Li is $\neg Lj$

Proof.

1) If Li equals $\neg Lj$, then all the interpretations are models for the above disjunction. For example $p \vee q \vee \neg p$

2) Assume that no literal Lk has a matching negation in the above disjunction. Then for each k , $1 \le k \le n$, we assign $t(Lk)=0$ if Lk is a prop. variable, or $t(Lk)=1$ if Lk is a negation of prop. variable. E.g., $\neg q \vee p \vee r$ can be made false by $t(p)=0$

From truth-value tables to formulas

.... ProC, C

Conjunctive Normal Form

First, we have to do a preprocessing to get rid of implications by replacing F=>G by ¬F V G (procedure IMPL).

Then, we have to process negations by using double negation and de Morgan laws (procedure NEG).

Now, we can call the fundamental procedure CNF that converts the formula F to its equivalent ConjNF.

```
function IMPL(F)
   case
      F is literal:     return F
      F is G=>H:        return ¬IMPL(G) V IMPL(H)
      F is G&H:         return IMPL(G) & IMPL(H)
      F is G V H:       return IMPL(G) V IMPL(H)
   endcase
endfunction
```

```
function NEG(F)
   case
      F is literal:     return F
      F is ¬¬H          return NEG(H)
      F is G&H          return NEG(G) & NEG(H)
      F is G V H:       return NEG(G) V NEG(H)
      F is ¬(G&H):      return NEG(¬G) V NEG(¬H)
      F is ¬(G V H):    return NEG(¬G) & NEG(¬H)
   endcase
endfunction
```

```
function CNF(F)
   case
      F is literal:     return F
      F is G&H:         return CNF(F) & Cnf(H)
      F is F V H:       return DISTR(CNF(G), CNF(H))
   endcase
endfunction
```

DISTR(G,H): both G, H are in ConjNF. However, if G is a conjunction, say G = G1 & G2 then we must use the distributive law:

G V H = (G1 V H) & (G2 V H)

Similarly for H :

```
function DISTR(G,H)
   case
      G is G1&G2:   return DISTR(G1,H) & DISTR(G2,H)
      H is H1&H2:   return DISTR(G,H1) & DISTR(G,H2)
      else          return G V H                    //no conjunction
   endcase
endfunction
```

Conclusion: Any formula F is converted to ConjNF by calling

CNF(NEF(IMPL(F)))

<u>Example</u>. Convert ¬p&q => p & (r=>q) to ConjNF

.... ProC, D

Predicate Calculus

……………… notes 3AI3

Comments.

□	a quantifier
⧄	opposite quantifier
Pred	set of predicates of the language $\mathscr{L}$
Fn	set of functors
Const	set of constants
Var	set of variables

The set Ss(F) of all the substructures of F is the smallest set containing F and with every element, it contains all its substructures.

t1 is a subterm of t : $t1 \in Ss(t)$. Notation: $t1 \lhd t$
Similarly: F1 is a subformula of F : $F1 \in Ss(F)$. Notation: $F1 \lhd F$

Set of the variables of the term t : $Var(t) = Var \cap Ss(t)$

Def. The set of all variables/ free variables/ bound variables of the formula F : notation: Var(F) / Fv(F) / Bv(F)

a) Let F be atomic formula P(t1,...,tn) then

$$Var(F) = \bigcup_{i=1}^{n} Var(t_i), \quad Fv(F) = Bv(F) = Var(F)$$

b) Let $F = \neg F1$ then

$$Var(\neg F1) = Var(F1), \quad Fv(\neg F1) = Fv(F1), \quad Bv(\neg F1) = Bv(F1)$$

Similarly for F = F1 => F2

c) Let $F = (\forall x)\, F1$ then

$$Var((\forall x)F1) = Var(F1)$$
$$Fv((\forall x)F1) = Fv(F1) \setminus \{x\}$$
$$Bv((\forall x)F) = \begin{cases} Bv(F1) & \text{if } x \notin Fv(F1) \\ Bv(F1) \cup \{x\} & \text{if } x \in Fv(F1) \end{cases}$$

Lemma. $Var(F) = Fv(F) \cup Bv(F)$

Definitions.

Formula F is closed if $Fv(F) = \emptyset$

Formula F is open if there is no x, G so that $(\forall x)G \lhd F$.

Formula has clean variables if $Fv(F) \cap Bv(F) = \emptyset$ and there is no x, G so that $(\forall x)G \lhd F$.

For $(\forall x)F, (\exists x)F$ we say that F minus any substructure of $(\forall x)G$, $(\exists x)G$ is the scope of $(\forall x), (\exists x)$.

.... MFF, PreC, p. 28

Note. Variable x is free in F if it is a leaf in the hierarchical tree of F and there is no path upwards from x to a node $(\forall x), (\exists x)$.
Variable $x \in Var(F)$ is bound if it is in the scope of some $(\forall x), (\exists x)$.

.... TextBook, p.104

Def. Substitution of term t0 for free occurrences of variable x into the term t / formula F : Notion $Sb^x_{t0}(t)$ / $Sb^x_{t0}(F)$

a) $Sb^x_{t0}(x) = t0$
   $Sb^x_{t0}(y) = y$ if $y \neq x$
   $Sb^x_{t0}(c) = c$ if c is a constant
b) $Sb^x_{t0}(f(t1,...,tn)) = f(Sb^x_{t0}(t1), ..., Sb^x_{t0}(tn))$
c) similarly for $P(t1,...,tn)$
d) $Sb^x_{t0}(\neg F) = \neg Sb^x_{t0}(F)$, similarly for $F1 \Rightarrow F2$
e) $Sb^x_{t0}((\forall x)F) = (\forall x)F$
   $Sb^x_{t0}((\forall y)F) = (\forall y)\, Sb^x_{t0}(F)$ if $y \neq x$

Note. Hence, t0 is substituted for the free variable x in F

Def. Term t0 is substitutable for variable x into formula F if there is no $(\forall z)G \lhd F$ so that $z \in Var(t0)$ and $x \in Fv((\forall z)G)$. Notation: Subble(t0,x,F)
Note. Hence, no free variable x in F occurs in the scope of $(\forall z)$, $(\exists z)$ for any variable z in t0 .

.... TextBook, p. 105, fig. 2.3; p.106-7, fig. 2.4

Lemma 32.
0) $F \vdash Sb^{x}_{t0}(F)$ if Subble(t0,x,F)
1) $\vdash Sb^{x}_{t0}(F) \Rightarrow (\exists x)F$ if Subble(t0,x,F)
$\vdash F \Rightarrow (\exists x)F$
2) $\vdash (\forall x)F \Rightarrow (\exists x)F$
3) $\vdash (\forall x)(F \Rightarrow G) \Rightarrow ((\forall x)F \Rightarrow (\forall x)G)$
4) $\vdash (\forall x)(F \Rightarrow G) \Rightarrow ((\exists x)F \Rightarrow (\exists x)G)$
5) $\vdash (\square x)F \Leftrightarrow F$ if $x \notin Fv(F)$
6) $\vdash (\forall x)(F \Leftrightarrow G) \Rightarrow ((\forall x)F \Leftrightarrow (\forall x)G)$
10) $\vdash (\forall x)(F \& G) \Leftrightarrow ((\forall x)F \& (\forall x)G)$
11) $\vdash (\exists x)(F \vee G) \Leftrightarrow ((\exists x)F \vee (\exists x)G)$
12) $\vdash F$ iff $\vdash Sb^{x}_{c}(F)$ where c is a constant not occurring in F
Proof.

.... MFF, PreC, p. 36, 37, 36b, 36c

Lemma 33. Let Subble(t0,x,F) then:
if $Sb^{x}_{t0}(F) \vdash G$ then $(\exists x)F \vdash G$

Lemma 33a.
$\vdash (\square x)(\square y)F \Leftrightarrow (\square y)(\square x)F$

Example. Prove using L. 33 (different way than L. 32, 4)):
$(\forall x)(F \Rightarrow G),\ (\exists x)F \vdash (\exists x)G$
Proof.
$(\forall x)(F \Rightarrow G),\ Sb^{x}_{t0}(F) \vdash Sb^{x}_{t0}(F) \Rightarrow Sb^{x}_{t0}(G)$ L.32,3); A5

(∀x)(F=>G), Sb^{x}_{t0}(F) |- Sb^{x}_{t0}(G)   MP
(∀x)(F=>G), Sb^{x}_{t0}(F) |- (∃x)G   L.32,1)
(∀x)(F=>G), (∃x)F |- (∃x)G

Example. Prove using L.33 (here Fv(F)={x}, Fv(G)={y}):
(∃x)F, (∀x)(∀y)(F=>G) |- (∀y)Q

Proof.

(∀x)(∀y)(F=>G), Sb^{x}_{t0}(F) |- (∀y) ((∀x)F => (∀x)G)   L.33a; L.32,3)
(∀x)(∀y)(F=>G), Sb^{x}_{t0}(F) |- (∀y) (Sb^{x}_{t0}(F) => Sb^{x}_{t0}(G))   A5
(∀x)(∀y)(F=>G), Sb^{x}_{t0}(F) |- (Sb^{x}_{t0}(F) => (∀y)Sb^{x}_{t0}(G))   L.32,3)
(∀x)(∀y)(F=>G), Sb^{x}_{t0}(F) |- (∀y)Sb^{x}_{t0}(G)   MP
(∀x)(∀y)(F=>G), Sb^{x}_{t0}(F) |- (∀y)G   L.32,12)
(∀x)(∀y)(F=>G), (∃x)(F) |- (∀y)G   L.33

Lemma 35. Let x ∉ Fv(G) . Then
1) (□x)(G=>F) <=> (G => (□x)F)
2) (□x)(F=>G) <=> ((⍁x)F => G)
3) (□x)(G V F) <=> (G V (□x)F)
(□x)(G & F) <=> (G & (□x)F)

Proof.

.... MFF, PreC, p. 36b

Lemma 36a. De Morgan laws
|- ¬(∀x)F <=> (∃x)¬F |- ¬(∃x)F <=> (∀x)¬F

Def. Variable y is strongly substitutable into F if y∉Var(F) and (∀y)G ⊀ F . Notation: SSubble(y, ., F)

Lemma 36b. (Renaming bound variables) Let SSubble(y,.,F) then for any variable x :
1) |- (∀x)F <=> (∀y) Sb^{x}_{y}(F)
2) |- (∃x)F <=> (∃y) Sb^{x}_{y}(F)

3) If F is closed then |- F <=> $Sb^{x}_{y}(F)$
Proof.

………. MFF, PreC, p. 40

Lemma 38. |- (∃y)(∀x)F => (∀x)(∃y)F
Note. Not the other way!

Semantics for PreC

In the syntax system: $\mathcal{B}$ |- F if *there is* a proof, i.e., a sequence F1,...,Fn (see its definition). However, $\mathcal{B}$ $\nvdash$ F if there is *no* proof. We'd have to consider every 'candidate' proof and show it is not one. Thus, syntax system is a 'positive' characterization of the logic: we need to find just one sequence (proof).

Semantics works in the opposite way. To show that $\mathcal{B}$ |≠ F is simple: find an interpretation that is model for $\mathcal{B}$ but not for F . (Interpretation is a model of a formula if it returns 1 .)
To show $\mathcal{B}$ |= F is a harder problem (see ProC): indicate that all the interpretations that are models for $\mathcal{B}$ are also models for F .
Thus, semantic system is a 'negative' characterization of the logic.

PreC: We will find that there are infinitely many interpretations.
To encounter (∃x)F we try to find some instance (concrete value) of x so that F 'holds' (is 1 for an interpretation) for that particular instance of x . Otherwise, if there is no value that satisfies F then 0 is returned.
Similarly, evaluating (∀x)F means to show that F evaluates to 1 for all possible values (instances) of x . Otherwise, if there is some value of x that does not satisfy F then 0 is returned.

Def. (Relational) structure $\mathfrak{M}$ for the language $\mathcal{L}$ is the 4-tuple

$[\, M, Pred^{\mathfrak{M}}, Fn^{\mathfrak{M}}, Const^{\mathfrak{M}} \,]$

where M is a universum of the structure $\mathfrak{M}$ (set of concrete values)

$Pred^{\mathfrak{M}}$ is a set that for each $P \in Pred$ of arity n contains a concrete relation $P^{\mathfrak{M}} \subseteq M^n$ of n-tuples over M

$Fn^{\mathfrak{M}}$ is a set that for each $f \in Fn$ of arity n contains a concrete function of the form $f^{\mathfrak{M}} : M^n \text{->} M$

$Const^{\mathfrak{M}}$ that for each constant $c \in Const$ contains a concrete element $c^{\mathfrak{M}} \in M$.

Example. Let $Pred=\{F, R\}$, $arity(F)=1$, $arity(R)=2$, $Const=\{i\}$.
Let $M=\{a,b,c\}$, $F^{\mathfrak{M}}=\{b,c\}$, $R^{\mathfrak{M}}=\{(a,a), (a,b), (a,c), (b,c), (c,c)\}$, $i^{\mathfrak{M}}=a$.
Check some formulas for this structure

.... TextBook, p. 125

Homework: TextBook, example 2.16 (p.125-6)

Def. Let $\mathfrak{M}$ be a structure. Evaluation e of variables is the mapping

$e : Var \text{->} M$

Def. Let $\mathfrak{M}$ be a structure, e an evaluation. Value $|t|^{\mathfrak{M}}[e]$ of the term t in $\mathfrak{M}$ with respect to e is defined by induction as follows:

a) $x \in Var$: $\quad |x|^{\mathfrak{M}}[e] = e(x)$

b) $c \in Const$: $\quad |c|^{\mathfrak{M}}[e] = c^{\mathfrak{M}}$

c) $f \in Fn$, $arity(f)=n$: $\quad |f(t1,\ldots,tn)|^{\mathfrak{M}}[e] = f^{\mathfrak{M}}(|t1|^{\mathfrak{M}}[e],\ldots,|tn|^{\mathfrak{M}}[e])$

Def. Let $\mathfrak{M}$ be a structure, e an evaluation. Value $|F|^{\mathfrak{M}}[e]$ of the formula F in $\mathfrak{M}$ with respect to e is defined as follows:

a) $P \in Pred$, $arity(P)=n$:

$$\begin{array}{rll} |P(t1,\ldots,tn)|^{\mathfrak{M}}[e] = 1 & \text{if the n-tuple } (|t1|^{\mathfrak{M}}[e], \ldots, |tn|^{\mathfrak{M}}[e]) & \in P^{\mathfrak{M}} \\ = 0 & & \notin \end{array}$$

b) $F = \neg F1$:

$|\neg F1|^{\mathfrak{M}}[e] = \overline{|F1|^{\mathfrak{M}}[e]}$

Similarly for F1=>F2

c) F = (∀x)F1:

$|(\forall x)F1|^{\mathfrak{M}}[e] = 1$ if $(\forall m \in M)\ |F1|^{\mathfrak{M}}[e^{x/m}] = 1$
where the evaluation $e^{x/m}$ arises from e by replacing the pair (x,e(x)) by (x,m)
0 otherwise

Note. $|(\exists x)F1|^{\mathfrak{M}}[e] = 1$ iff $(\exists m \in M)\ |F1|^{\mathfrak{M}}[e^{x/m}] = 1$

Def. Formula F is (semantically) valid in $\mathfrak{M}$ with respect to e ($\mathfrak{M} \models F[e]$) if

$|F|^{\mathfrak{M}}[e] = 1$

Def. 1) Formula F is satisfiable if there is some structure $\mathfrak{M}$ and some evaluation e so that $\mathfrak{M} \models F[e]$.

2) Formula F is (semantically) valid in the structure $\mathfrak{M}$ ($\mathfrak{M}$ is a model of F) ($\mathfrak{M} \models F$) if for each evaluation e :

$\mathfrak{M} \models F[e]$

3) Formula F is (semantically) valid ($\models F$) if it is valid for all the structures of the language $\mathfrak{L}$.

4) We say that $\mathfrak{M}$ is a model of a set $\mathfrak{P}$ of formulas if it is a model of each formula in $\mathfrak{P}$.

4) Let $\mathfrak{P}$ be a set of premises, G a formula, then G is (semantically) valid from the premises $\mathfrak{P}$ if each model $\mathfrak{M}$ of $\mathfrak{P}$ if also the model of G ($\mathfrak{P} \models G$).

Note. The symbol $\models$ is overloaded: it is used for $\mathfrak{M} \models F$ as well as for $\mathfrak{P} \models F$.

Theorem. PreC is complete, i.e., for each $\mathfrak{B}$, F

$$\mathfrak{B} \vdash F \qquad \text{iff} \qquad \mathfrak{B} \models F$$

Undecidability of PreC

Decision problem. A solution to a decision problem is an algorithm (e.g., a program in C) that takes *any* problem instance as its input and *always* terminates, producing a *correct* 'yes' or 'no' output.

The decision problem of provability in PreC is undecidable, i.e., there is no algorithms which, given any formula F, decides whether $\vdash F$ or $\nvdash F$.

Gödel's Theorem.

2nd-order Logic

PreC is 1st-order logic; it processes quantifiers above variables: $(\forall x)$, $(\exists x)$.
2nd-order logic: it also processes quantifiers above predicates: $(\forall P)$, $(\exists P)$.

Resolution Principle

……… 31A3 notes