Pf: - Sps S is unsatisfiable
- then for any A,
$S \models A$ and $S \models \neg A$
(trivially: if $v \models S$ then $v \models A, \neg A$
... because there are no such v!)

- by completeness theorem
$S \vdash A$ and $S \vdash \neg A$
- then proofs are finite
- hence for some finite $S_0 \subseteq S$ have
$S_0 \vdash A$ and $S_0 \vdash \neg A$
- hence
$S_0 \models A$ and $S_0 \models \neg A$ (by soundness)
- but since no $v \models A, \neg A$
must have S_0 unsatisfiable!

First-order logic

- first-order logic is an enrichment of PL in which:
  - prop'l variables p_i are replaced by familiar mathematical propositions like "$x < y$" or "$f(z) = 25$"
  - also add quantifiers to the mix!
    - $\exists$ "there exists"
    - $\forall$ "for all"

- before defining FOL, need to define relations, functions, and structures

---

Relations and functions

Def'n Sps A is a non-empty set and $n \in \{1, 2, \dots\}$.

An **n-ary relation** on A is a subset $R \subseteq A^n$, where $A^n = \{(a_1, \dots, a_n) : a_i \in A\}$ is the set of n-tuples of A

↪ we call n the **arity** of R

Ex (i) Sps $A = \mathbb{N} = \{0, 1, 2, \dots\}$. Define:
$R = \{n \in \mathbb{N} : n \text{ is prime}\}$; then $R \subseteq \mathbb{N}$ is a 1-ary ("unary") rel'n on $\mathbb{N}$.
$S = \{(m, n) \in \mathbb{N}^2 : m < n\}$; then $S \subseteq \mathbb{N}^2$ is a 2-ary ("binary") rel'n on $\mathbb{N}$.

(ii) Sps $A = \mathbb{R}$. Define:
$T = \{(x, y, z) \in \mathbb{R}^3 : y \text{ is between } x, z$ (i.e. $x < y < z$ or $z < y < x$)$\}$; then $T \subseteq \mathbb{R}^3$ is a 3-ary ("ternary") rel'n on $\mathbb{R}$.

(iii) Sps $G = \langle V, E \rangle$ is a graph w/ vertex set V and edge set E.
Define $F = \{(x, y) \in V^2 :$ there is a

path $x = x_0, x_1, \ldots, x_n = y$ from x to y]; then $F \subseteq V^2$ is a binary rel'n on G

Notation: We think of an n-ary rel'n $R \subseteq A^n$ as being a "property" of n-tuples. We also write $R(a_1, \ldots, a_n)$ to mean $(a_1, \ldots, a_n) \in R$.

- e.g. for the unary R on $\mathbb{N}$ above $R(n)$ means $n \in R$ means "n is prime"
- for binary rel'ns specifically, we sometimes write aRb for $R(a,b)$
- e.g. the rel'n S in (i) is the familiar $<$ rel'n on $\mathbb{N}$
- will write $a S b$ or $a < b$ instead of $S(a,b)$ or $(a,b) \in S$

Note: On any set A we have the binary rel'n $=$ of equality, which is special (more later!)

Def'n An n-ary function on A is a function $f: A^n \to A$

i.e. a rule for assigning to each $(a_1, \ldots, a_n) \in A^n$ an output $f(a_1, \ldots, a_n) \in A$

(we call n the arity of f

Ex (i) Sps $A = \mathbb{N}$. Define:
$f(n) = n^2$ (unary function)
$g(m,n) = m \cdot n$ (binary function)

(ii) Sps $A = \mathbb{R}$. Define:
$$h(a,b,c) = \begin{cases} 1 & \text{if } a<b<c \\ 0 & \text{o/w} \end{cases}$$
(ternary function)

Convention: - also allow "0-ary" functions
- Since $A^0 = \{\emptyset\}$ (i.e. only 0-ary tuple is $\emptyset$), 0-ary functions are just rules $f(\emptyset) = c$, where $c \in A$
- think of f as just being the constant c (unique output of f)

Convention: - for binary functions f may write afb for $f(a,b)$
- e.g. $a+b$ instead of $+(a,b)$

Structures

Def'n a **structure** consists of a nonempty set A (the **universe**) along w/ a collection of relations and functions on A.

- We write $\mathcal{A} = \langle A; f, g, h, \ldots; R, S, T, \ldots \rangle$ to mean $\mathcal{A}$ is a structure w/ universe A, functions $f, g, h, \ldots$, and rel'ns $R, S, T, \ldots$

- Note: universe A can be finite or infinite (not empty)
- can have finitely or infinitely many functions (or none); same w/ rel'ns

- whereas wffs in PL are logical combos of basic propositions (the variables p_i) wffs in FOL will be assertions about structures (more in a moment!)

Def'n The signature of a structure $\mathcal{A} = \langle A; F, g, h, \ldots; R, S, T, \ldots \rangle$ is
$\langle \text{arity}(F), \text{arity}(g), \text{arity}(h), \ldots; \text{arity}(R), \text{arity}(S), \ldots \rangle$

Ex (i) The "structure of arithmetic" is
$\mathcal{N} = \langle N; 0, S, +, \cdot, < \rangle$
where 0 is the constant 0
S is the unary successor function:
$S(n) = n+1$
$+$ and $\cdot$ are the usual (binary) sum and product on N
$<$ is the usual (binary) order on N
- the arity of $\mathcal{N}$ is $\langle 0, 1, 2, 2; 2 \rangle$

(ii) A group is a set G equipped w/ an identity el't e and binary group operation $\circ$.

i.e. a structure of the form

$$\mathcal{G} = \langle G; e; \circ \rangle$$

arity of $\mathcal{G}$ is $\langle 0, 2; \rangle$ (blank after 2 indicates: no rel'ns on G)

(iii) A graph is a nonempty vertex set V equipped w/ a binary edge rel'n $E \subseteq V^2$, i.e. a structure of the form

$$\langle V; ; E \rangle$$

(no functions)

arity is $\langle 0; 2 \rangle$

First-order languages

- goal: develop a logic (FOL) in which we can formally write assertions about structures.

Ex: (i) Consider the structure

$$\mathcal{N} = \langle \mathbb{N}; 0, S, +, \cdot; < \rangle$$

How could we write "the square of an odd number is odd" in this structure?

- we introduce following symbols:

$x, y, z, \ldots$	variables
$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$	connectives
$(,)$	parentheses
$\exists, \forall$	quantifiers: "exists", "for all"
$=$	equality symbol
$0, S, +, \cdot, <$	functions and relns from $\mathbb{N}$

- our assertion above is equiv to:
  "For all natural numbers x, if x is odd then $x \cdot x$ is odd"
- since we don't have a reln in $\mathbb{N}$ expressing "x is odd", can rewrite this as "there exists a natural number y s.t. $x = 2y+1$"
- don't have the constants 2 and 1 in $\mathbb{N}$, but can write them as $S(S(0))$ and $S(0)$, respectively.
- replacing "For all" w/ $\forall$ and "there is" w/ $\exists$ can rewrite our assertion:

$$\forall x \left(\exists y \left(x = S(S(0)) \cdot y + S(0) \right) \Rightarrow \exists z \left(x \cdot x = S(S(0)) \cdot z + S(0) \right) \right)$$

will be a wff in FOL!

- the thus now written using only symbols above, we often abbreviate to increase readability, writing

2 for $S(S(0))$, 1 for $S(0)$,
x^2 for $x \cdot x$, etc.
$\forall x \, (\exists y (2y+1 = x) \Rightarrow \exists z (2z+1 = x^2))$

(ii) Sps $G = \langle G, e, \circ \rangle$ is a group.
We can write "every element of G has a two-sided inverse" by
$\forall x \, \exists y \, (x \cdot y = e \wedge y \cdot x = e)$

Syntax of FOL:

- We formalize ideas above

Defn a first-order language L consists of: (i) logical symbols:

$x_1, x_2, \ldots$	variables
$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$	connectives
$\exists, \forall$	quantifiers
$(,)$	parentheses
$=$	equality

(ii) non-logical symbols consisting of: relation symbols $R_1, R_2, \ldots$
w/ associated arities $n_1, n_2, \ldots$
function symbols $f_1, f_2, \ldots$
w/ associated arities $m_1, m_2, \ldots$

- note: there are symbols (not actual functions and relns), but those

symbols do have arities (these will be the arities of the functions and rel'ns they refer to)

- we call 0-ary function symbols constant symbols and denote them w/ letters like c, d, ...
- Since the logical symbols are ind. in every language, L is determined by its nonlogical symbols
  so often indicate this by writing $L = \{F_1, F_2, \ldots ; R_1, R_2, \ldots\}$

Ex (i) Sps c is a constant symbol, f, g binary function symbols, R a ternary rel'n symbol
then $L = \{c, f, g; R\}$ is a first-order language.

(ii) The language of arithmetic is
$L_{ar} = \{0, S, +, \cdot, <\}$
where 0 is a constant symbol
S unary function symbol
+, · binary function symbols
< binary rel'n symbol

Note: For now these are only symbols ... later will "interpret" them as actual constants, functions, rel'ns.

Terms

- next want to define the wffs of FOL
- were complicated than PL wffs
- first need some "building blocks" for wffs called terms
- intuitively: terms refer to elements of the structure (... that can be computed by its functions)

Def'n Sps $L = \{F_1, F_2, \ldots ; R_1, R_2, \ldots\}$ is a language. The **L-terms** are defined inductively:

(i) every var. x_i is a term

(ii) if F is n-ary function symbol and $t_1, \ldots, t_n$ terms then $Ft_1t_2 \ldots t_n$ is a term

(note: if $n=0$, i.e. F is a constant, then F itself is a term)

- here we are using "Polish notation" writing $Ft_1 \ldots t_n$ instead of $F(t_1, \ldots, t_n)$

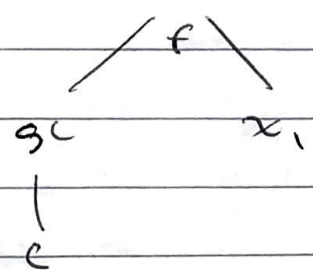
Ex: Sps $L = \{c, g, f\}$ (c: constant, g: unary, f: binary (number of symbols))

then: x_1, c are terms

so: gc is a term

$fgcx_1$ is a term

can parse: $f\,g\,c\,x_1$



<u>Fact</u>: have unique readability for terms
i.e. every term can be read uniquely
as x_i (variable), c (constant), or
$f t_1 t_2 \ldots t_n$ for f n-ary function symbol
$t_1, \ldots, t_n$ terms

<u>Convention</u>: often write $f t_1 \ldots t_n$
more traditionally as $f(t_1, \ldots, t_n)$
↳ term above would be: $f(g(c), x_1)$

<u>Also</u>: for familiar binary function symbols
(e.g. $+$) write $(t_1 \, f \, t_2)$ instead of
$f(t_1, t_2)$

- often use $x, y, z, \ldots$ for variables
(technically all should be among $x_1, x_2, \ldots$)

<u>Ex</u>: in $L_{ar} = \{0, S, +, \cdot, <\}$ following
are terms:
$((x+y)\cdot z)$ (really: $\cdot + x\,y\,z$)
$((x \cdot x) \cdot x + S(0))$ (think: $x^3 + 1$)

Notation - if variables in a term t are among $x_1, \ldots, x_n$, will write $t(x_1, \ldots, x_n)$
- if moreover $u_1, \ldots, u_n$ are terms we write $t[x_1/u_1, \ldots, x_n/u_n]$
for expression obtained by replacing each instance of x_i w/ u_i
↳ Fact: is always a term too!

Ex: if $t(x_1, x_2) = f(g(c), x_1)$ is our term from above (x_2 does not appear - ok!) and $u_1 = g(g(x_5))$ $u_2 = c$ then
$t(x_1/u_1, x_2/u_2) = f(g(c), g(g(x_5)))$

Well-formed formulas in FOL

- wffs in FOL are built out of simple wffs called atomic formulas

Def'n an **atomic formula** is a string of the form
$$R t_1 t_2 \ldots t_n$$
where R is an n-ary rel'n symbol and $t_1, \ldots, t_n$ are terms.

Ex: Sps $L = [c, f, g, R]$ (c: constant, g: unary, f: binary, R: binary rel'n)
then $c, x_1, x_2, g(c), f(g(c), x_1)$ are terms (written formally: gc, $fgcx_1$)

Sc: $Rfgcx_1x_2$ and $=x_1c$ are atomic formulas

- usually write $R(t_1,\ldots,t_n)$ for atomic fmlas (for readability)
- for binary rel'ns S, write $(t_1 S t_2)$ instead of $S(t_1,t_2)$

- Sc e.g. fmlas above can be rewritten:
  $R(f(g(c),x_1),x_2)$
  $(x_1 = c)$

- general wffs built from atomic fmlas just like wffs in PL are built from propositional variables
  ... but now we have quantifiers too!

Def'n well-formed formulas (wffs) are defined recursively:
(i) atomic fmlas are wffs
(ii) if A, B are wffs, so are $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \Rightarrow B)$, $(A \Leftrightarrow B)$
(iii) if A is a wff and x is a variable, $\exists x A$ and $\forall x A$ are wffs

Ex: Sps $L = \{<\}$ consists of a single binary rel'n symbol $<$

then: x, y are terms (variables)
so: $(x<y)$, $(y<x)$ are wffs (atomic fmlas)
so: $((x<y) \vee (y<x))$ is a wff
so: $\exists y\,((x<y) \vee (y<x))$ " "
so: $\forall x \exists y\,((x<y) \vee (y<x))$ " "

can parse: $\forall x \exists y\,((x<y) \vee (y<x))$

$(\forall x$

$\exists y\,((x<y) \vee (y<x))$

$(\exists y$

$((x<y) \vee (y<x))$

$\vee$

$(x<y)$ $(y<x)$

$<$ $<$

x y y x

say: β is a subformula of a wff A if β appears in parse tree of A

- e.g. $((x<y) \vee (y<x))$ is a subfmla of $\forall x \exists y\,((x<y) \vee (y<x))$

(ii) if $L_{ar} = \{0, S, +, \cdot, <\}$ then:

$\forall x (\exists y (x = 2 \cdot y + 1) \Rightarrow \exists z (x \cdot x = 2 \cdot z + 1))$

abbrev. $S(S(0))$ $S(0)$

is a wff.