Greek letters will *only* be expressions in our metalanguage (that designate strings/ expressions of our object language).

1. alpha	= "α"	Alpha	= "Α"
2. beta	= "β"	Beta	= "Β"
3. gamma	= "γ"	Gamma	= "Γ"
4. delta	= "δ"	Delta	= "Δ"
5. epsilon	= "ε"	Epsilon	= "Ε"
6. zeta	= "ζ"	Zeta	= "Ζ"
7. eta	= "η"	Eta	= "Η"
8. theta	= "θ"	Theta	= "Θ"
9. iota	= "ι"	Iota	= "Ι"
10. kappa	= "κ"	Kappa	= "Κ"
11. lambda	= "λ"	Lambda	= "Λ"
12. mu	= "μ"	Mu	= "Μ"
13. nu	= "ν"	Nu	= "Ν"
14. xi	= "ξ"	Xi	= "Ξ"
15. omicron	= "ο"	Omicron	= "Ο"
16. pi	= "π"	Pi	= "Π"
17. rho	= "ρ ϱ"	Rho	= "Ρ"
18. sigma	= "σ ς"	Sigma	= "Σ"
19. tau	= "τ"	Tau	= "Τ"
20. upsilon	= "υ"	Upsilon	= "Υ"
21. phi	= "φ"	Phi	= "Φ"
22. chi	= "χ"	Chi	= "Χ"
23. psi	= "ψ"	Psi	= "Ψ"
24. omega	= "ω"	Omega	= "Ω"

Our conventions:

* using capital Greek letters as variables in the metalanguage ("metavariables") designating **sets of** object language **formulas**

* using small Greek letters like φ (phi), ψ (psi), χ (chi) as metavariables designating **single** object language **formulas**

* using small Greek letters like α, β as metavariables designating **term expressions** of the object language

* using small Greek letters like ξ (xi), ζ (zeta) as metavariables designating **variables** in the object language (x, x', x'', x''', etc.)

Also only part of the metalanguage: spelled-out words like "`and`", "`or`", "`iff`" (the object language will use symbols)

When the signature of the object language is meant to represent numbers, I'll *try* to stop using `0` as a term constant in the object language, instead I'll use `Z`. I'll use `S` as a functor expression in the object language, so the idea will be that these object language expressions are *meant to* be standing for these objects (numbers) in a model's domain:

```
   Z --> 0
  SZ --> 1
 SSZ --> 2
SSSZ --> 3
```
etc.

When n is a *metalanguage* variable for a number (≥ 0), I'll sometimes write $\mathtt{S}^n\mathtt{Z}$ to stand for the object language term expression consisting of n copies of `S` followed by a `Z`. This is a piece of metalanguage notation for designating that object language string. The n and the superscripts aren't grammatical inside the object language.

When we write just bare `Z`, `SZ`, and so on, these are our metalanguage names for those same object language strings.

In some ways it'd be helpful if we also distinguished between the use of symbols like `+`, `=`, `<` inside the object language and inside the metalanguage. Sometimes I'll write out the words "`plus`" and so on; that will always be meant as part of the metalanguage. But inevitably sometimes we'll also use the symbols in the metalanguage. If it's important to be very explicit, I might write something like this:

$\mathtt{SnZ} \frown \mathtt{"+"} \frown \mathtt{SZ}$

but if/when I write:

$\mathtt{S}^n\mathtt{Z} + \mathtt{SZ}$

you should understand it the same way. It doesn't make sense for the `+` in the second line to be the metalanguage plus symbol, since the expressions before it and after it are expressions designating *strings in the object language*, not expressions designating numbers. That whole metalanguage expression picks out a complex object language term expression.

If, on the other hand, I write $n + 1$, then I am using `+` to mean addition, and this metalanguage expression designates a number.

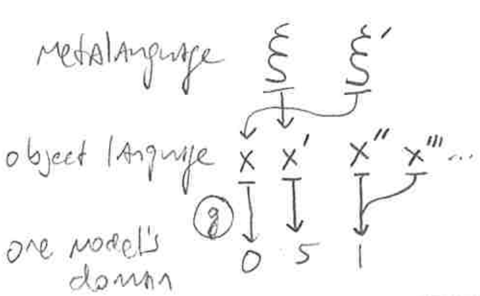
Moby Dick

The first letter of Moby Dick is "C". (using)

The first letter of <u>Moby Dick</u> is "M". (mentioning)

Assignments $g(\cdot)$ and A model's $\pi(\cdot)$ take $\left\{ \begin{array}{l} \text{metalanguage expressions designating} \\ \text{object language strings} \end{array} \right\}$ as arguments,

deliver objects from the model's domain (or sets/functions over them) as results



what is $g(\xi)$?

what is $g(\xi^{\frown}{}''{}'')$?

$\Gamma \models \phi$
entails
has as <u>logical consequence</u>

$\Gamma, \Delta, \psi \models \phi$

} ϕ is true on every model (+assignment) where all the premises are

$\models \phi$
is <u>logically true</u>
valid formula/sentence

} ϕ is true on every model (+ass)

[Some logical truths are called <u>Tautologies</u>

$0=0 \vee 0\neq 0$ } yes

$0=0$
$\exists x(x=0)$ } no

$0\neq 1$
$0<1$ } not even logical truths]

when ϕ is true on/ satisfied by some model (+ass)

$=$

ϕ is <u>satisfiable</u>

set Γ is satisfiable when there's <u>some</u> model making <u>all</u> of its elements true

when ϕ is true on/ satisfied by no model (+ass)

$=$

ϕ is <u>unsatisfiable</u>
logically false

set Γ is unsatisfiable when there's no model making all of its elements true

can write as: $\Gamma \models \bot$

[Some logical falsehoods are called <u>contradictions</u>

$0=0 \wedge 0\neq 0$ } yes

$0\neq 0$ } no]

ϕ is unsatisfiable iff $\sim\phi$ is true on every model (+ass)

$\Gamma \cup \{\phi\}$ is unsat iff $\Gamma \models \sim\phi$

(what if the fault is inside Γ?
$P, \sim P \models$ anything)

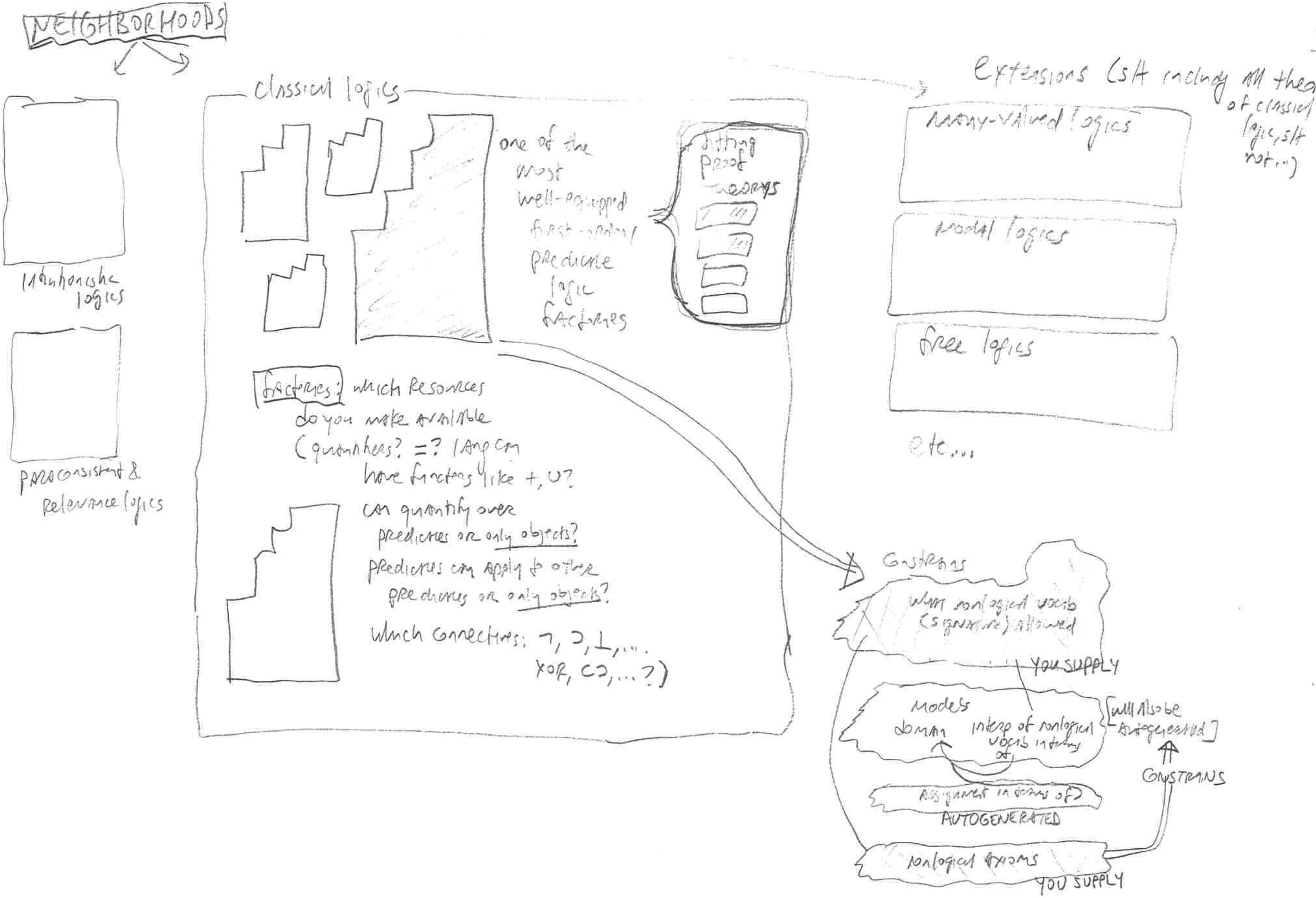
$\phi \mathrel{=\!\!|\!\models} \psi$ says ϕ, ψ are <u>logically equivalent</u>

meaning for every model+ass (M,g) $[\![\phi]\!]_{Mg} = [\![\psi]\!]_{Mg}$

equivalent to

$\phi \models \psi$ and $\phi \mathrel{=\!|} \psi$

(Goldrei uses $\equiv$; others use other notation)


NEIGHBORHOODS
Intuitionistic logics
paraconsistent & relevance logics
classical logics
one of the most well-equipped first-order predicate logic factories
sitting proof theorys
factories: which resources do you make available
(quantifiers? =? language have functions like +, ∪?
can quantify over predicates or only objects?
predicates can apply to other predicates or only objects?
which connectives: ¬, ⊃, ⊥, ...
XOR, ⊂⊃, ...?)
extensions (s/t includes all theorems of classical logic, s/t not...)
many-valued logics
modal logics
free logics
etc...
Constrains
what nonlogical vocab (signature) allowed
YOU SUPPLY
Models
domain
interp of nonlogical vocab in terms of
[will also be autogenerated]
CONSTRAINS
Assignment (in terms of)
AUTOGENERATED
nonlogical axioms
YOU SUPPLY