Simple Types / Non-dependent types

type theory $\cong$ computational view of types

Categorical judgments

Harper jumps right in; not useful for teaching outwards!

sets → not a useful intuition

A type

$a : A$

$a_1 \equiv a_2 : A$ $\quad$ a_1 & a_2 are definitionally equal as elements of type A → rules of calculation

hypothetical judgment

abbr Γ

$$\underbrace{x_1 : A_1, \cdots x_n : A_n}_{\text{variables}} \vdash \underbrace{a(x_1, \cdots x_n)}_{\text{open term}} : A$$

one idea: a mapping: $a : A_1, \cdots A_n \to A$

the term is paramaterized by $x_1 \cdots x_n$

general theory of variables

↳ structural properties ← by experience, it doesn't work if this is not presented earlier

identity variable * reflexivity

$$\frac{}{\Gamma, x : A, \Gamma' \vdash x : A}$$

"I'm willing checks that Frank and Steve have to cash"

computation substitution * transitivity

$$\frac{\Gamma, x : A, \Gamma' \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma, \Gamma' \vdash [a/x]b : B}$$

α-equiv / capture avoiding substitution

"It's like HS algebra" ← variable

A generalized HS mathematics

weakening

$$\frac{\Gamma \vdash b : B}{\Gamma, x : A \vdash b : B} \, \omega$$

possibility of vacuous dependence

contraction

$$\frac{\Gamma, x : A, y : A, \Gamma' \vdash b : B}{\Gamma, z : A, \Gamma' [z, z/x, y] b : B}$$

exchange $$\frac{\Gamma, x:A, y:B, \Gamma' \vdash c:C}{\Gamma, y:B, x:A, \Gamma' \vdash c:C}$$ ⟵ a little more complicated in dep theory

⟶ end structural properties ⟵

"what are the rules for variables? how do mappings behave"

Def'al equality

1) Equivalence relation
2) Congruence (see rules: "you can replace equals w/ equals)
3) $\boxed{\text{Functionality}}$ maps respect def'l equality

$$\frac{\Gamma, x:A \vdash b:B \qquad \Gamma \vdash a_1 \equiv a_2 : A}{\Gamma \vdash [a_1/x]b \equiv [a_2/x]b : B}$$

equality is delicate & important

(what I say and don't say)

Defining types

build

1) Formation: how to construct a type
towards T 2) Intro
away T 3) Elim
what can I do w/ it
Gentzen
4) Congruence principles
5) Computation rules

} comes from proof theory

(?) 6) Unicity / Universality characterize the type

it all comes down to that

comes from category theory

Limits	Colimits
Negative types	**Positive types**

Intuitively: either the introduction is "primary" or the elimination is primary (elim-oriented)

Focusing

Unit

$$\frac{}{1\ \text{type}}\ 1F$$

unit

You could have called it w/alus (complaining about bubble PLS which call this void)

$$\frac{}{\Gamma \vdash * : 1}\ 1I$$

$\langle\rangle$

(no elim)

(no comp)

(no cong)

DEGENERATE

Q: why is this elim oriented?

A: hang on.

$$\begin{tikzcd} & TT \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\ PT \arrow[rr, leftrightarrow] & & CT \end{tikzcd}$$

If you know something, then it has to show up on all three corners; if it's only on one corner, it's bullshit. It's just an esoteric doodad.

Product

$$\frac{A\ \text{type} \qquad B\ \text{type}}{A \times B\ \text{type}}\ \times F$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : A}{\Gamma \vdash \langle a, b \rangle : A \times B}\ \times I$$

or $\Gamma; x : A, y : B \vdash \langle x, y \rangle : A \times B$

will not rewrite congruence rules ↳ you'll get it

$$\frac{\Gamma \vdash c : A \times B}{\Gamma \vdash \text{fst } c : A}\ \times E_1 \qquad \frac{\Gamma \vdash c : A \times B}{\Gamma \vdash \text{snd } c : B}\ \times E_2$$

$$\frac{\Gamma \vdash a_1 \equiv a_2 : A \qquad \Gamma \vdash b_1 \equiv b_2 : B}{\Gamma \vdash \langle a_1, b_1 \rangle \equiv \langle a_2, b_2 \rangle : A \times B}$$

$$\frac{\Gamma \vdash c_1 \equiv c_2 : A \times B}{\Gamma \vdash \text{fst } c_1 \equiv \text{fst } c_2 : A \times B}$$

etc

$$\underbrace{\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B}{\Gamma \vdash \mathrm{fst}\,\langle a, b\rangle \equiv a : A}\ \times C_1 \qquad\qquad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B}{\Gamma \vdash \mathrm{snd}\,\langle a, b\rangle \equiv b : B}}_{\text{commutation conditions}}$$

$$\begin{tikzcd}
 & \Gamma \arrow[ddl, bend right, "a"'] \arrow[d, dotted, "! \ \langle a,b\rangle"] \arrow[ddr, bend left, "b"] & \\
 & A \times B \arrow[dl, "\mathrm{fst}"'] \arrow[dr, "\mathrm{snd}"] & \\
A & & B
\end{tikzcd}$$

uniqueness

$$\frac{\Gamma \vdash c : A \times B}{\Gamma \vdash \langle \mathrm{fst}\ c, \mathrm{snd}\ c\rangle \equiv c : A \times B}\ \times U$$

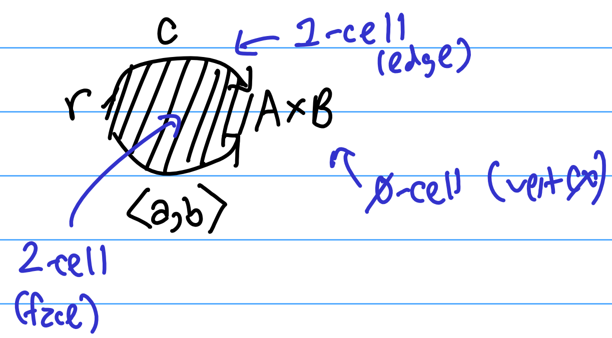
(revisit this later)

is this the right unicity principle?

Should be a weaker condition then definitional equivalence

Turned into: "homotopy canonicity"

have a 2-cell (map between maps) which relates the two.



Function Space

"the premiere example"

$$\frac{A\ \text{type} \qquad B\ \text{type}}{A \to B\ \ \text{type}} \to F$$

$$\frac{\Gamma, x{:}A \vdash b : B}{\Gamma \vdash \lambda x.\, b : A \to B} \to I$$

$$\frac{\Gamma \vdash b : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash b(a) : B} \to E$$

$$\frac{\Gamma, x{:}A \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma \vdash (\lambda x.\, b)(a) \equiv [a/x]b : B} \to C\ (\beta)$$

$$\frac{\Gamma \vdash a : A \to B}{\Gamma \vdash \lambda x.\, b_1 \equiv \lambda x.\, b_2 : A \to B} \to U(\xi)$$

$X_i \to$ not function extensionality

unicity

$$\frac{\Gamma \vdash a : A \to B}{\Gamma \vdash a \equiv \lambda x.\, a(x) : A \to B} \ (\eta)$$

$$\frac{\Gamma, x{:}A \vdash a_1(x) \equiv a_2(x) : B}{\Gamma \vdash a_1 \equiv a_2 : A \to B} \ (\text{ext})$$

Warning:

$\lambda x{:}\mathbb{N}.\ x + \emptyset \not\equiv \lambda x{:}\mathbb{N}.\ \emptyset + x$

b/c $x{:}\mathbb{N} \vdash x + \emptyset \not\equiv \emptyset + x : \mathbb{N}$

cf HS algo: definitional equality is simplification, but some equations require proof

to show this requires proof

Positive types

positive = elimination
negative = introduction

Empty

$$\frac{}{0 \text{ type}} \; 0F$$

void

(no-I)

$$\frac{\Gamma \vdash a : 0}{\Gamma \vdash \mathrm{abort}(a) : A} \; 0E$$

Same for products
{lazy}

fst
snd

function's a black box
I just need to know how to apply it (λ is a native's FFI is built in).

eg) $x : 0 \vdash \mathrm{abort}(x) : A$
$x : \mathbb{N} \to 0 \vdash \mathrm{abort}(x(3)) : A$

idea: abort makes conditional branches the same type, even when one end is contradictory

Sum

$$\frac{A \text{ type} \quad B \text{ type}}{A+B \text{ type}} \; +F$$

giant blind spot, the missing sum type
$1B must be —Hoare
null ptr analysis ooga-booga

eg) $2 := 1+1$

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \mathrm{inl}(a) : A+B} \; +I_1 \qquad \frac{\Gamma \vdash b : B}{\Gamma \vdash \mathrm{inr}(b) : A+B} \; +I_2$$

you can't ~~just~~ green in the theory: case is an induction principle

$$\frac{\Gamma \vdash c : A+B \quad \Gamma, x:A \vdash d : C \quad \Gamma, y:B \vdash e : C}{\Gamma \vdash \mathrm{case}(c; x.d; y.e) : C} \; +E$$

$$\mathrm{case}(\mathrm{inl}(a); x.d; y.e) \equiv [a/x]d : C$$
$$\mathrm{case}(\mathrm{inr}(b); x.d; y.e) \equiv [b/x]e : C \qquad +C$$

unicity?

why no diagram for functions?

$$\begin{tikzcd}
A \arrow[dr, "\mathrm{inl}"] \arrow[ddr, bend right, "d"'] & \Gamma \arrow[d, "c"] & B \arrow[dl, "\mathrm{inr}"'] \arrow[ddl, bend left, "e"] \\
& A+B \arrow[d, dashed, "{[d,e]}", "1"'] & \\
& C &
\end{tikzcd}$$

In HoTT, path structure of +/- types is different

ex) express unicity of elimination rule

Props as Types

STT ~ IPL

$$1 \longleftrightarrow \top$$

$$A \times B \longleftrightarrow A \wedge B$$

validity is justified by proof theory

$$\vdots$$

$$A \text{ type} \longleftrightarrow A \text{ prop}$$

$$M : A \ (\exists M) \longleftrightarrow A \text{ true}$$

idea: $x : a \vdash b : B \longleftrightarrow \boxed{A \leq B}$

$(\exists b)$

preorder on propositions!!!

HW) 1) show $\leq$ is a preorder (RT)

2) $\top$ is greatest $\quad A \leq \top$

$A \wedge B$ is glb $\quad A \wedge B \leq A \quad A \wedge B \leq B$

$$\frac{C \leq A \qquad C \leq B}{C \leq A \wedge B}$$

complemented distributive lattice

↑

boolean algebra

⇑

classical logic

3) $\bot$ is least $\quad A \leq A \vee B \quad B \leq A \vee B$

$A \vee B$ is lub

$$\frac{A \leq C \qquad B \leq C}{A \vee B \leq C}$$

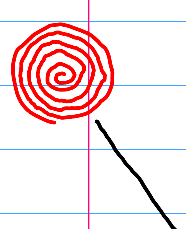
4) $\supset$ is exponential $\quad (A \supset B) \wedge A \leq B$

$$\frac{C \wedge A \leq B}{C \leq A \supset B}$$

we have a heyting (pre-) algebra

(no antisymmetry w/o univalence)

5) distributivity $\quad$ ($\wedge$ dist over $\vee$)



"squeezing balloon": it just pops out somewhere else, but it's the same thing

Negation

$\neg A \equiv A \to 0$ i.e. $A \supset \bot$

easy: $A \leq \neg\neg A$

question: $\neg\neg A \leq A$?

answering this is a lot of work, but we do NOT have this in general

aim of proof theory

"I don't not like asparagus"

irrefutability $\neq$ assent

trivial $A \vee \neg A \leq \top$

not $\top \leq A \vee \neg A$ in general

but if you know something about A

~~Define $\bar{A}$ (complement of A) to be smallest B s.t. $A \wedge B \leq \bot$~~

~~don't have this in Heyting algebras~~

~~note that $A \wedge \neg A \leq \bot$. $\boxed{\text{Not necessarily maximal!}}$~~

~~If we have $\bar{A}$, then $\neg A \leq \bar{A}$~~

~~Boolean Algebra: complemented HA~~

~~every A has a complemented $\bar{A}$~~

The point is, $\neg A$ is NOT a complement

(there is no closed world assumption)

Ex) Show that $\neg\neg(A \vee \neg A)$ for general A

intuitionistic logic is consistent with classical logic the power of homotopy type theory

is the ability to make finer distinctions

Correction

$\neg A$ is by $\underline{\text{dfn}}$ the **largest** B inconsistent w/ A

1) $A \wedge \neg A \leq \bot$
2) if $A \wedge B \leq \bot$ then $B \leq \neg A$

$\bar{A}$ is by $\underline{\text{dfn}}$ the **smallest** B that complements A

1) $\top \leq A \vee \bar{A}$
2) if $\top \leq A \vee B$ then $\bar{A} \leq B$

so $\neg A \leq \bar{A}$ but we do not in general have $\bar{A} \leq \neg A$

if so, then we have BOOLEAN ALGEBRA in which

$\top \leq A \vee \neg A = A \vee \bar{A}$

$\bar{\bar{A}} \leq A$ (and $A \leq \bar{\bar{A}}$)

"real PL research"

Key idea: Type-indexed <u>family</u> of types

eg) $x : \mathbb{N} \vdash \mathrm{Vec}(x)$ type $\qquad \{\mathrm{Vec}(x)\}_{x:\mathbb{N}}$

type of vectors of length x

so if I have $a : \mathbb{N}$ then $\mathrm{Vec}\, a$ is a type

$a_1 \equiv a_2 : \mathbb{N} \qquad \mathrm{Vec}(a_1) \equiv \mathrm{Vec}(a_2)$

definitional equality of types

eg) $x, y : A \vdash \mathrm{Id}_A(x,y)$ type

type of identifications of x, y

proofs of equivalence

cells/paths (HoTT)

eg) $x, y : A,\ p, q : \mathrm{Id}_A(x,y) \vdash \mathrm{Id}_{\mathrm{Id}_A(x,y)}(p,q)$

$\rightsquigarrow$ higher dimensional type theory

eventually: a type is a <u>weak ∞-groupoid</u>

$\Gamma \vdash A$ type	$\Gamma \vdash A_1 \equiv A_2$
$\Gamma \vdash a : A$	$\Gamma \vdash a_1 \equiv a_2 : A$

Γ ctx $\qquad \Gamma' \vdash \gamma : \Gamma$

$x_1 : A_1, \ldots, x_n : A_n \vdash A$ type

$\{\{A(x_1, \ldots x_n)\}_{x_n \in A_n}\}$

$\{\ldots\}_{a_1 \in A_1}$

no longer just a mapping from Γ to A

(so you need <u>fibrations</u>)

Structural Properties

$$\frac{}{\Gamma, x:A, \Gamma' \vdash x:A}\ V/R$$

can be $b:B$ or B type

$$\frac{\Gamma, x:A, \Gamma' \vdash J \qquad \Gamma \vdash a:A}{\Gamma[a/x]\Gamma' \vdash [a/x]J}\ S/T$$

sequential dependencies

$$\frac{\Gamma, x:A, \Gamma' \vdash B\ \text{type} \qquad \Gamma \vdash a_1 \equiv a_2 : A}{\Gamma[a_{1\text{ or }2}/x]\Gamma' \vdash [a_1/x]B \equiv [a_2/x]B}$$

$$\frac{\Gamma, x:A, \Gamma' \vdash b:B \qquad \Gamma \vdash a_1 \equiv a_2 : A}{\Gamma[a_{1\text{ or }2}/x]\Gamma' \vdash [a_1/x]b \equiv [a_2/x]b : [a_{1\text{ or }2}/x]B}$$

functionality

CENTRAL

$$\frac{\overset{a_1 \equiv a_2 : A}{\Gamma \vdash a:A} \qquad \Gamma \vdash A \equiv B}{\Gamma \vdash a:B}$$

$$\frac{\Gamma \vdash a:A \qquad \Gamma \equiv \Gamma'}{\Gamma' \vdash a:A}$$

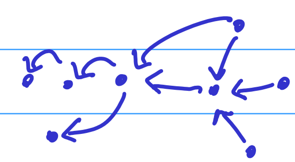
$a_1 \equiv a_2 : A$

A type

$A_1 \equiv A_2$

respect

Idea: Γ is a DAG



can reorder non-deps

$$\frac{\Gamma \vdash \overset{B\ \text{type}}{J} \qquad \Gamma \vdash A\ \text{type}}{\Gamma, x:A \vdash \underset{B\ \text{type}}{J}}$$

weakening

no x!!

$$\frac{\Gamma, x:A, y:B, \Gamma' \vdash J \qquad \Gamma \vdash B\ \text{type}}{\Gamma, y:B, x:A, \Gamma' \vdash J}$$

exchange

$$\frac{\Gamma, x:A, y:A, \Gamma' \vdash J}{\Gamma, z:A, [z,z/x,y]\Gamma' \vdash [z,z/x,y]J}$$

contraction

Natural numbers (simple)

$$\frac{}{\mathbb{N}\ \text{type}} \qquad \frac{}{\Gamma \vdash 0 : \mathbb{N}} \qquad \frac{\Gamma \vdash a : \mathbb{N}}{\Gamma \vdash \text{succ}(a) : \mathbb{N}}$$

base, inductive set

$$\frac{\Gamma \vdash a : \mathbb{N} \qquad \Gamma \vdash b : D \qquad \Gamma, x : \mathbb{N}, y : D \vdash c : D}{\Gamma \vdash \text{rec}[b; x, y . c](a) : D}$$

$$\text{plus} = \lambda x y .\ \underline{\text{rec}}\,[x ; \underbrace{u}_{\text{pred}}, \underbrace{v}_{x+u} .\ \text{succ}(v)]\,(y)$$

$$\text{rec}[b; x, y . c](0) \equiv b : D$$

$$\text{rec}[b; x, y . c](\text{succ}(a)) \equiv [a, \underbrace{\underline{\text{rec}}[b; x, y . c](a)}_{\text{recursive call}} / x, y]\, c : D$$

Check
$a + 0 \equiv a$
$a + \text{succ}\, b \equiv \text{succ}\,(a+b)$
$0 + a \not\equiv a$
$\text{succ}\, a + b \not\equiv \text{succ}\,(a+b)$ — Source of a lot of aggravation
for all m, n, $\bar{m} + \bar{n} \equiv \bar{n} + \bar{m}$
$x : \mathbb{N}, y : \mathbb{N} \vdash x + y \not\equiv y + x : \mathbb{N}$
$\text{Vec}\,(0 + x) \not\equiv \text{Vec}\ x$
$\text{Vec}\ x \equiv \text{Vec}\,(x + 0)$

definitional equality ~ calculational
propositional equality ~ proof

notational device

<u>Universes</u> : cumulative hierarchy of universes (simple types)

informal

1) $\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \dots$

2) $\mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \mathcal{U}_2 \subseteq \dots$

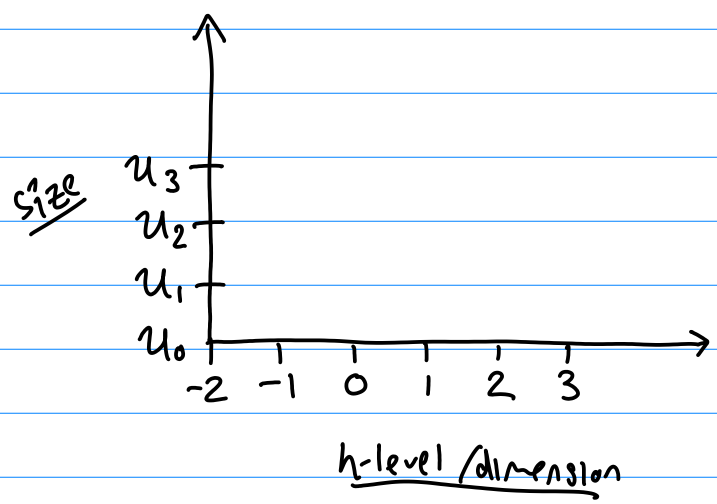
needed to avoid paradox $\mathcal{U} : \mathcal{U}$

replace A type by $A \in \mathcal{U}$ i.e. $A \in \mathcal{U}_i$ for some i

$$\frac{i \geq 0}{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}} \; (\mathcal{U}_i\text{-F}) \; (\mathcal{U}_i\text{-I}) \qquad \frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash A : \mathcal{U}_{i+1}} \; (\mathcal{U}_{i+1}\text{-I})$$

Idea: 1) elements of universes are types

2) every type is the element of some universe



Simple types → Dependent types

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash A \to B : \mathcal{U}_i}$$

system solves constraints (e.g. Coq)

$\dots$

$$\frac{\Gamma \vdash A : \mathcal{U} \qquad \Gamma \vdash B : \mathcal{U}}{\Gamma \vdash A + B : \mathcal{U}}$$

it's like an inaccessible cardinal, where all the powersets are contained in it

$$\frac{\Gamma, x : A \vdash a(x) : [\mathrm{inl}(x)/z]D \qquad \Gamma, y : B \vdash b(y) : [\mathrm{inr}(x)/z]D}{\Gamma \vdash \underline{\mathrm{case}}[x.a;\, y.b](c) : [c/z]D}$$

the join of the case does $\underline{\text{not}}$ have to be constant!

Define $2 := 1 + 1$

$tt := \mathrm{inl}(*)$

$ff := \mathrm{inr}(*)$

$\mathrm{if}(a; b; c) := \mathrm{case}[\_.b; \_.c](a)$

not a special form, the binder is just degenerate

Notice:

$$\frac{a : 2 \qquad z : 2 \vdash D : \mathcal{U} \qquad b : [tt/z]D \qquad c : [tt/z]D}{\mathrm{if}(a; b; c) : \underline{\underline{[a/z]D}}}$$

$x : \mathrm{Vec}(10)$

$y : \mathrm{Vec}(20)$

$z : 2$

$$\vdash \mathrm{if}(z; x; y) : \underbrace{\mathrm{Vec}(\mathrm{if}(z; 10; 20))}_{\text{family of types}}$$

beginning of expressiveness of type theory

by analogy: a "runtime value" in a "type"

↪ but it's not proper to speak of it that way

be careful!

as long as β-only, no problems

<u>unicity</u> causes problems

exercise: ind. principle for booleans

Remark: In the setting I'm describing, propositions (things we state in math) are types. You're taught that propositions are booleans; in geometry class, you learned truth tables; the rules of boolean logic, and then you went to Euclid writing chart form proofs, where lines were justified with principles. But I never really did understood what the truth tables had to do with the charts. It's not so clear; the thing they wanted to teach you was the proof objects, but there was this crazy story that there were only two propositions, true and false. I want you to keep in mind: booleans are not propositions, and the if I'm writing is not implication. All conventional programming languages screw this up; a program that yields a boolean when it's run is a predicate, when it is not. A myriad of messes in PL stem from this misunderstanding.

$$\dfrac{\begin{array}{ll}\Gamma \vdash a : \mathbb{N} & \Gamma, z:\mathbb{N} \vdash D : \mathcal{U} \\ \Gamma \vdash b : [0/z]D & \Gamma, x:\mathbb{N}, y:[x/z]D \vdash c : [\mathrm{succ}(x)/z]D\end{array}}{\Gamma \vdash \underbrace{\mathrm{rec}[b; x,y.c]}_{\text{recursion}}(a) : [a/z]D}$$

recursion { principle of mathematical induction

↳ computational content thereof

We need languages which are suitable for human discourse but are also executable. That is what type theory is promising us. The math is the code. What the hell are DSLs about? What is a domain? The domain is all of math and science. I don't see how you draw boundaries around domains. You want a language where you can express mathematics, and that should be executable. Mathematics is the language of science. If we have this, then we have a complete unification, and reasoning and programming are the same. The whole point of doing type theory is that grand unified theory.

So a smart person would just go home at this point. But the program verification folks would then write this imperative program with messages and objects and mutable state, and then prove it. But the math is the program. Why not just run the spec? It's like when you dialed phone numbers by moving your finger as long as the number you wanted to dial. It's crazy!