Crash Course on Classical Theory of Computation

Eric Bond Fall 2017

Outline

- Introduction to Logic.
- Brief History of Logic
- Logic's Influence on the Construction of the Machine.
- Formal Language Theory & Automata Theory
- Lambda Calculus
- Intuitionistic Logic & Natural Deduction
- Curry Howard Isomorphism.
- Bleeding Edge: Homotopy Type Theory.

Computation

- Definition: A computer is an artifact that performs computation.
  - But what is computation?

Logic

- A Language?
  - Syntax
  - Semantics

Syntax of Propositional Logic

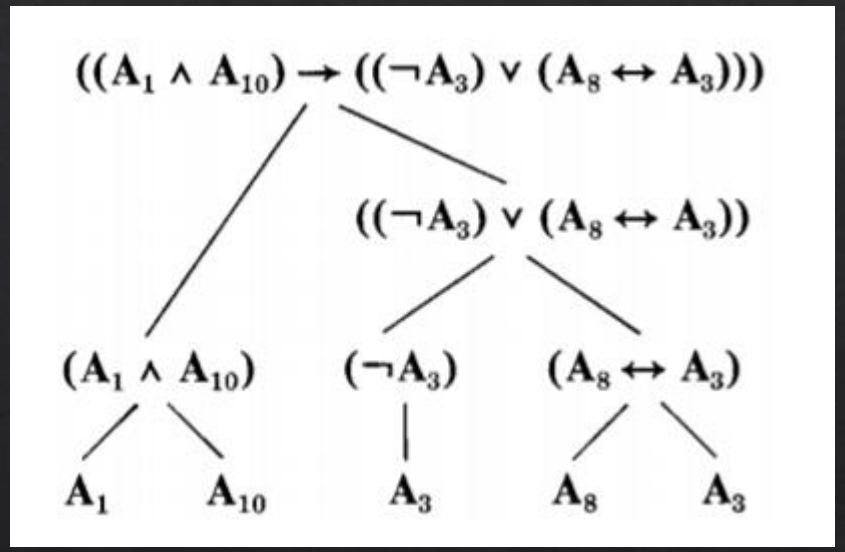
Symbol	Verbose name	Remarks
(	left parenthesis	punctuation
)	right parenthesis	punctuation
$\neg$	negation symbol	English: not
$\wedge$	conjunction symbol	English: and
$\vee$	disjunction symbol	English: or (inclusive)
$\rightarrow$	conditional symbol	English: if __, then __
$\leftrightarrow$	biconditional symbol	English: if and only if
$\mathbf{A}_1$	first sentence symbol	
$\mathbf{A}_2$	second sentence symbol	
...		
$\mathbf{A}_n$	nth sentence symbol	
...		

$$\mathcal{E}_{\neg}(\alpha) = (\neg\alpha)$$
$$\mathcal{E}_{\wedge}(\alpha, \beta) = (\alpha \wedge \beta),$$
$$\mathcal{E}_{\vee}(\alpha, \beta) = (\alpha \vee \beta),$$
$$\mathcal{E}_{\rightarrow}(\alpha, \beta) = (\alpha \rightarrow \beta),$$
$$\mathcal{E}_{\leftrightarrow}(\alpha, \beta) = (\alpha \leftrightarrow \beta).$$

(a) Every sentence symbol is a wff.
(b) If α and β are wffs, then so are $(\neg\alpha)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$.
(c) No expression is a wff unless it is compelled to be one by (a) and (b).

Examples

$((\rightarrow A_3.$



Induction Principle

- For well formed formulas
  - If S is a set of well formed formulas containing all the sentence symbols and closed under all five formula-building operations, then S is the set of *all* wffs.
- In general
  - Assume that C is the set generated from B by the functions in F.
  - If S is a subset of C that includes B and is closed under the functions of F, then S = C.
- "Inductive Structures"
  - Groups generated by an element
  - Span of a basis set
  - Data structures (binary tree, lists, abstract data types…)
  - Programming language syntax

Extension of Syntax (First Order Logic)

- Logical Symbols
  - Parenthesis
  - Sentential connective symbols: ->, V, &, <->
  - Variables: v1, v2, v3, … vn
  - Equality symbol (optional): =
- Parameters
  - Quantifier symbols: ∀, ∃
  - Predicate symbols: ex) P(v1, v2) or P(v1,v2,v3)
  - Constant symbols:
  - Function symbols: ex) f(v1)

A FOL Syntax for Set Theory

- Has..
  - Equality
  - A two place Predicate symbol: $\in$
  - A constant symbol: $\emptyset$
  - Variables are sets
  - Quantifiers
- Ex) Translation process for the pair-set Axiom

$\forall v_1 \forall v_2$ [There is a set whose members are exactly v_1 and v_2]

$\forall v_1 \forall v_2 \exists v_3$ [The members of v_3 are exactly v_1 and v_2]

$\forall v_1 \forall v_2 \exists v_3 \forall v_4 (v_4 \in v_3 \leftrightarrow v_4 = v_1 \vee v_4 = v_2)$

A FOL Syntax for Elementary Number Theory

- Has
  - Equality
  - 1 two-place predicate symbol: $<$
  - Constant symbol: $\mathbf{0}$
  - 1 One-place function symbol: $\mathbf{S}$ (successor)
  - 3 Two-place function symbols:
    - $+$ (addition)
    - $*$ (multiplication)
    - E (exponent)

$$= + \mathbf{SS0\,SS0\,SSSS0}.$$

$\forall v_1$ [If v_1 is nonzero, then v_1 is the successor of some number.]

$$\forall v_1\ (v_1 \neq \mathbf{0} \rightarrow \exists v_2\ v_1 = \mathbf{S}v_2).$$

$$\forall v_1\ ((\neg = v_1 \mathbf{0}) \rightarrow (\neg \forall v_2 (\neg = v_1 \mathbf{S} v_2))).$$

Elementary Semantics (2-value Logic)

- Let S be the set of all sentence symbols.
  - $v : \mathcal{S} \to \{F, T\}$
- Let $\bar{S}$ be the set of wff that can be built from S and the 5 rules.
  - $\overline{v} : \overline{\mathcal{S}} \to \{F, T\}$
- Definition of $\bar{v}$:
  - For any $A \in \mathcal{S}$, $\overline{v}(A) = v(A)$.

1. $\overline{v}((\neg \alpha)) = \begin{cases} T & \text{if } \overline{v}(\alpha) = F, \\ F & \text{otherwise.} \end{cases}$

2. $\overline{v}((\alpha \wedge \beta)) = \begin{cases} T & \text{if } \overline{v}(\alpha) = T \text{ and } \overline{v}(\beta) = T, \\ F & \text{otherwise.} \end{cases}$

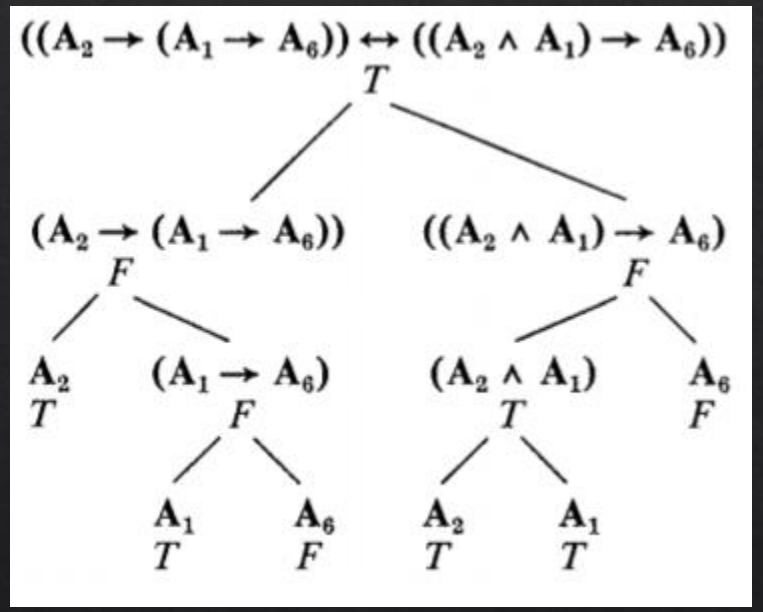
3. $\overline{v}((\alpha \vee \beta)) = \begin{cases} T & \text{if } \overline{v}(\alpha) = T \text{ or } \overline{v}(\beta) = T \text{ (or both)}, \\ F & \text{otherwise.} \end{cases}$

4. $\overline{v}((\alpha \to \beta)) = \begin{cases} F & \text{if } \overline{v}(\alpha) = T \text{ and } \overline{v}(\beta) = F, \\ T & \text{otherwise.} \end{cases}$

5. $\overline{v}((\alpha \leftrightarrow \beta)) = \begin{cases} T & \text{if } \overline{v}(\alpha) = \overline{v}(\beta), \\ F & \text{otherwise.} \end{cases}$

Example

- v & $\bar{v}$ define an ‘interpretation’ of a well formed formula in propositional logic.



Recursion

Recursion Theorem Assume that the subset C of U is freely generated from B by f and g, where

$$f : U \times U \to U,$$
$$g : U \to U.$$

Further assume that V is a set and F, G, and h functions such that

$$h : B \to V,$$
$$F : V \times V \to V,$$
$$G : V \to V.$$

Then there is a unique function

$$\overline{h} : C \to V$$

such that

(i) For x in B, $\overline{h}(x) = h(x)$;
(ii) For x, y in C,

$$\overline{h}(f(x, y)) = F(\overline{h}(x), \overline{h}(y)),$$
$$\overline{h}(g(x)) = G(\overline{h}(x)).$$

Model Theoretic Semantics

Formally, a *structure* $\mathfrak{A}$ for our given first-order language is a function whose domain is the set of parameters and such that[1]

1. $\mathfrak{A}$ assigns to the quantifier symbol $\forall$ a nonempty set $|\mathfrak{A}|$ called the *universe* (or *domain*) of $\mathfrak{A}$.
2. $\mathfrak{A}$ assigns to each n-place predicate symbol P an n-ary relation $P^{\mathfrak{A}} \subseteq |\mathfrak{A}|^n$; i.e., $P^{\mathfrak{A}}$ is a set of n-tuples of members of the universe.
3. $\mathfrak{A}$ assigns to each constant symbol c a member $c^{\mathfrak{A}}$ of the universe $|\mathfrak{A}|$.
4. $\mathfrak{A}$ assigns to each n-place function symbol f an n-ary operation $f^{\mathfrak{A}}$ on $|\mathfrak{A}|$; i.e., $f^{\mathfrak{A}} : |\mathfrak{A}|^n \to |\mathfrak{A}|$.

$$\exists x \forall y \neg y \in x$$

Structure Swapping

And with a different structure….

Example. Consider the language for set theory, whose only parameter (other than $\forall$) is $\in$. Take the structure $\mathfrak{A}$ with

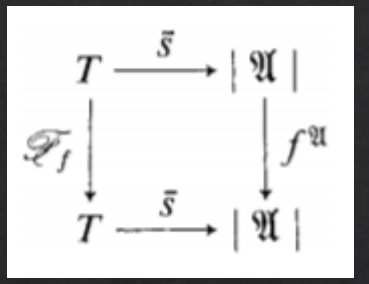
$|\mathfrak{A}|$ = the set of natural numbers,
$\in^{\mathfrak{A}}$ = the set of pairs $\langle m, n \rangle$ such that $m < n$.

$$\exists x \, \forall y \, \neg \, y \in x$$

Homomorphic Structures and Alternative Logics

- Wff of FOL are Inductively generated as well..
- Definable Structures?
  - Groups
  - Vector Spaces
  - Fields
  - Graphs
- Many valued logic
  - Possibility Theory vs Probability Theory
  - Set Theory vs Fuzzy Set Theory

$$\bar{s} : T \to |\mathfrak{A}|,$$



HOMOMORPHISM THEOREM Let h be a homomorphism of $\mathfrak{A}$ into $\mathfrak{B}$, and let s map the set of variables into $|\mathfrak{A}|$.

(a) For any term t, we have $h(\bar{s}(t)) = \overline{h \circ s}(t)$, where $\bar{s}(t)$ is computed in $\mathfrak{A}$ and $\overline{h \circ s}(t)$ is computed in $\mathfrak{B}$.

(b) For any quantifier-free formula α not containing the equality symbol,

$$\models_{\mathfrak{A}} \alpha[s] \quad \text{iff} \quad \models_{\mathfrak{B}} \alpha[h \circ s].$$

(c) If h is one-to-one (i.e., is an isomorphism of $\mathfrak{A}$ into $\mathfrak{B}$), then in part (b) we may delete the restriction "not containing the equality symbol."

(d) If h is a homomorphism of $\mathfrak{A}$ *onto* $\mathfrak{B}$, then in (b) we may delete the restriction "quantifier-free."

A Brief History of Logic, Rise of Axiomatic Mathematics

- Aristotle: Syllogisms, ‘Categories’
- Gottfried Leibniz (1700s):
  - Universal Characteristic
  - Does a universal and formal language able to express all Mathematical, Scientific, and Metaphysical concepts exist? What would its calculus be?
- Gottlob Frege(mid 1800s):
  - *A Formal Language for Pure Thought Modeled on that of Arithmetic*
- George Boole(mid 1800s):
  - *An Investigation into the Laws of Thought*
- The goals of Hilbert’s Program(early 1900s)
  - A formulation of all mathematics; in other words all mathematical statements should be written in a precise formal language, and manipulated according to well defined rules.
  - Completeness: a proof that all true mathematical statements can be proved in the formalism.
  - Consistency: a proof that no contradiction can be obtained in the formalism of mathematics. This consistency proof should preferably use only "finitistic" reasoning about finite mathematical objects.
  - Conservation: a proof that any result about "real objects" obtained using reasoning about "ideal objects" (such as uncountable sets) can be proved without using ideal objects.
  - Decidability: there should be an algorithm for deciding the truth or falsity of any mathematical statement.

Euclid's Postulates

1. A straight line segment can be drawn joining any two points.

2. Any straight line segment can be extended indefinitely in a straight line.

3. Given any straight line segment, a circle can be drawn having the segment as radius and one endpoint as center.

4. All right angles are congruent.

5. If two lines are drawn which intersect a third in such a way that the sum of the inner angles on one side is less than two right angles, then the two lines inevitably must intersect each other on that side if extended far enough. This postulate is equivalent to what is known as the parallel postulate.

Euclid's fifth postulate cannot be proven as a theorem, although this was attempted by many people. Euclid himself used only the first four postulates ("absolute geometry") for the first 28 propositions of the Elements, but was forced to invoke the parallel postulate on the 29th. In 1823, Janos Bolyai and Nicolai Lobachevsky independently realized that entirely self-consistent "non-Euclidean geometries" could be created in which the parallel postulate *did not hold*. (Gauss had also discovered but suppressed the existence of non-Euclidean geometries.)

$*54{\cdot}43.\quad \vdash :. \alpha, \beta \in 1 . \supset : \alpha \cap \beta = \Lambda . \equiv . \alpha \cup \beta \in 2$

Dem.

$\vdash . *54{\cdot}26 . \supset \vdash :. \alpha = \iota' x . \beta = \iota' y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y .$

$[*51{\cdot}231] \qquad \equiv . \iota' x \cap \iota' y = \Lambda .$

$[*13{\cdot}12] \qquad \equiv . \alpha \cap \beta = \Lambda \qquad (1)$

$\vdash . (1) . *11{\cdot}11{\cdot}35 . \supset$

$\vdash :. (\exists x, y) . \alpha = \iota' x . \beta = \iota' y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda \qquad (2)$

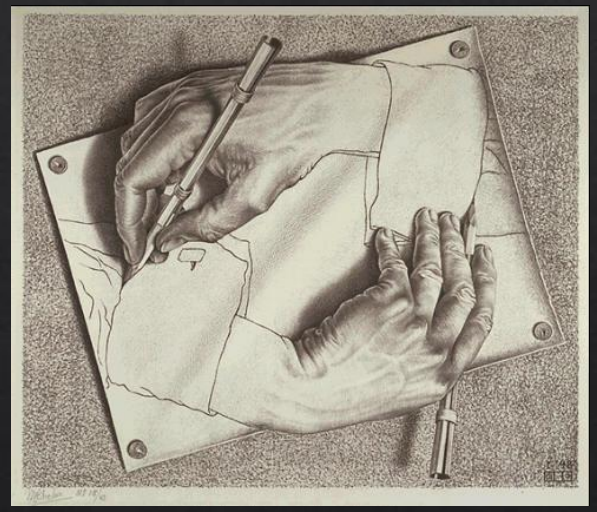
$\vdash . (2) . *11{\cdot}54 . *52{\cdot}1 . \supset \vdash . \text{Prop}$

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

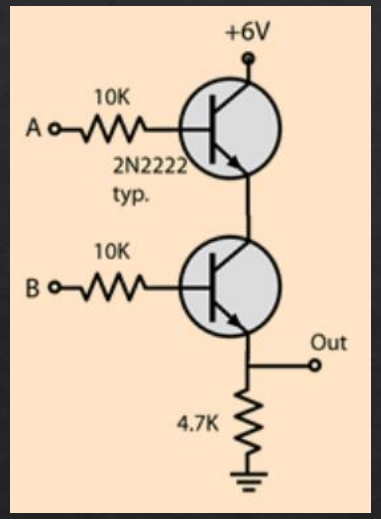
Principia Mathematica

Continued..

- Is Mathematics pure formalism?
  - “No” – Godel’s Incompleteness Theorems (1931)
    - Any formal system strong enough to express arithmetic will be ‘incomplete’. I.e. there are theorems that are unprovable within your formal system.
    - “This sentence is false”
- ***Entscheidungsproblem*** (German for decision problem) - Hilbert (1928)
  - The problem asks for an algorithm that takes as input a statement of a first-order logic (possibly with a finite number of axioms beyond the usual axioms of first-order logic) and answers "Yes" or "No" according to whether the statement is *universally valid*, i.e., valid in every structure satisfying the axioms
  - “No” – Church-Turing Thesis(1936)
    - Alan Turing via Turing Machine model
    - Alonzo Church via Lambda Calculus
      - Yielded 2 equally powerful models of computation.
      - Firm definition of algorithm and computation.

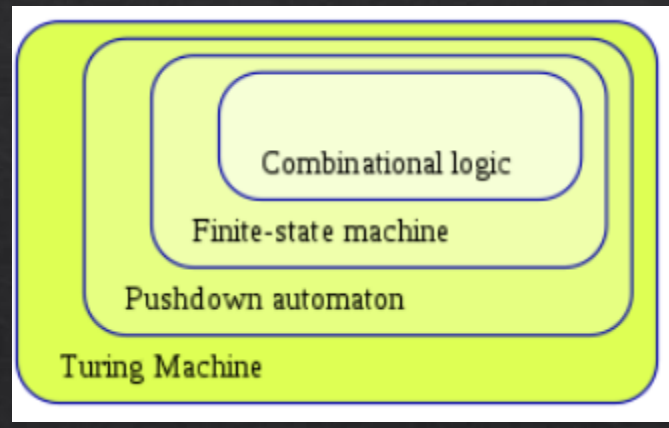


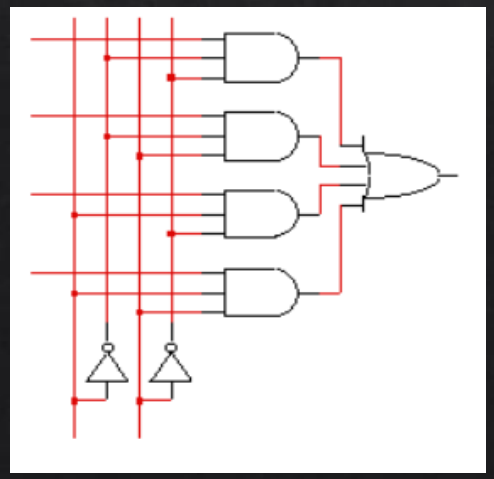
Logic's Influence on Architecture.

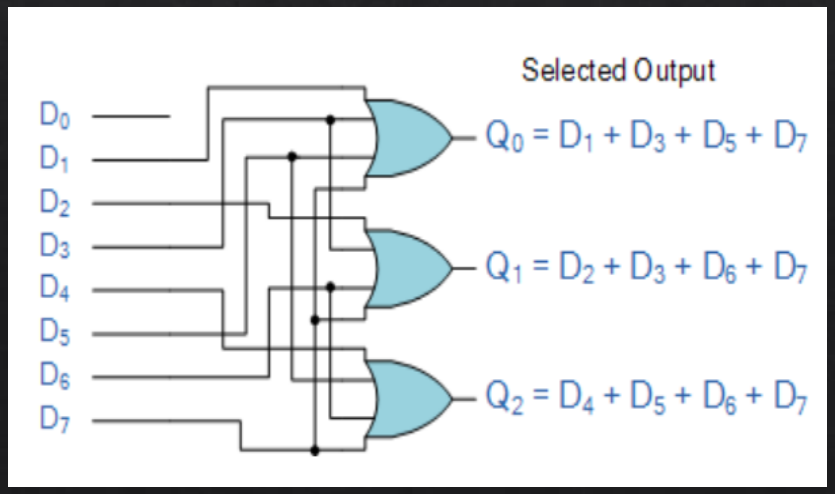


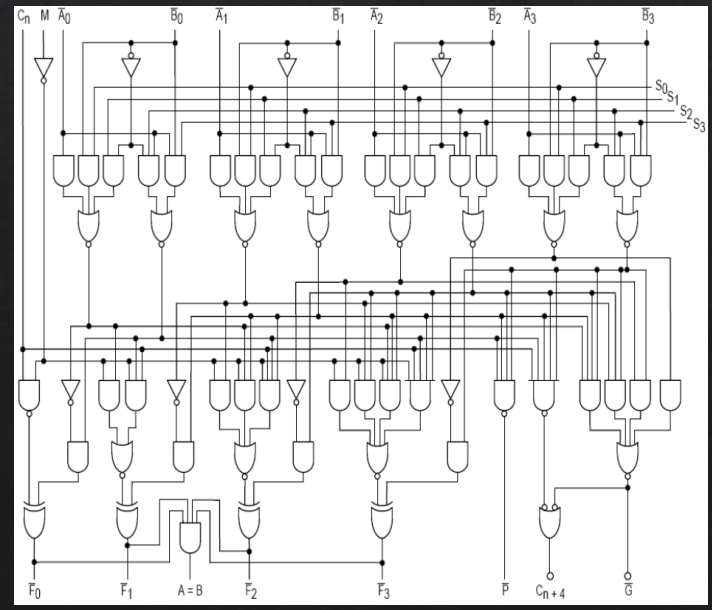
Logic Gates

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
Symbol							
Truth Table	A X 0 1 1 0	B A X 0 0 0 0 1 0 1 0 0 1 1 1	B A X 0 0 1 0 1 1 1 0 1 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 1	B A X 0 0 1 0 1 0 1 0 0 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 0	B A X 0 0 1 0 1 0 1 0 0 1 1 1





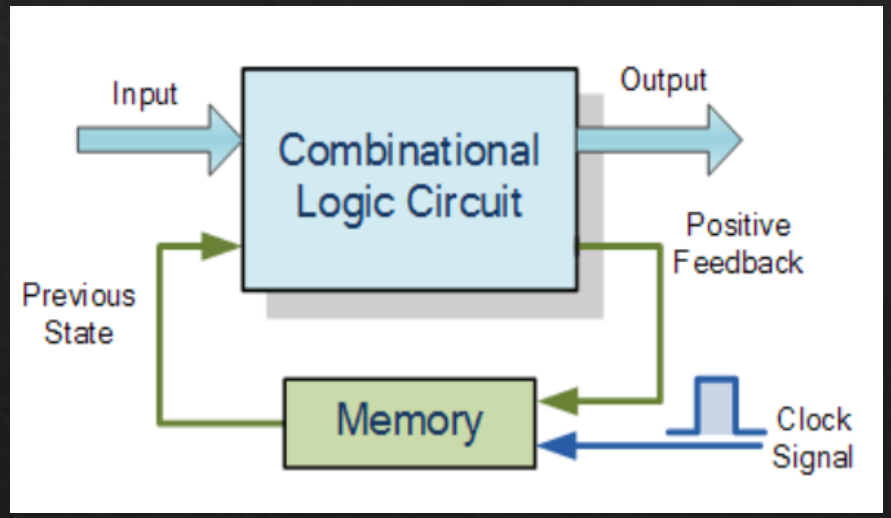


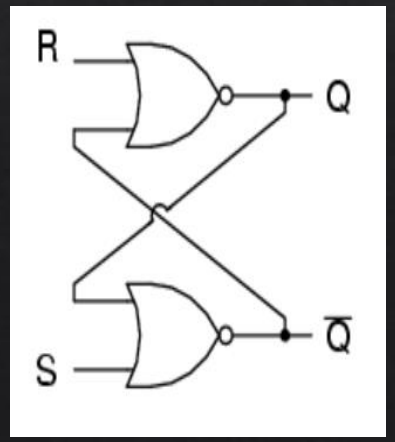


Time Dependent vs Independent Logic

- Combinatorial logic circuits are pure Boolean functions.
  - Independent of time,.
  - Output only as a function of input

- Time dependent circuits use input and previous state.





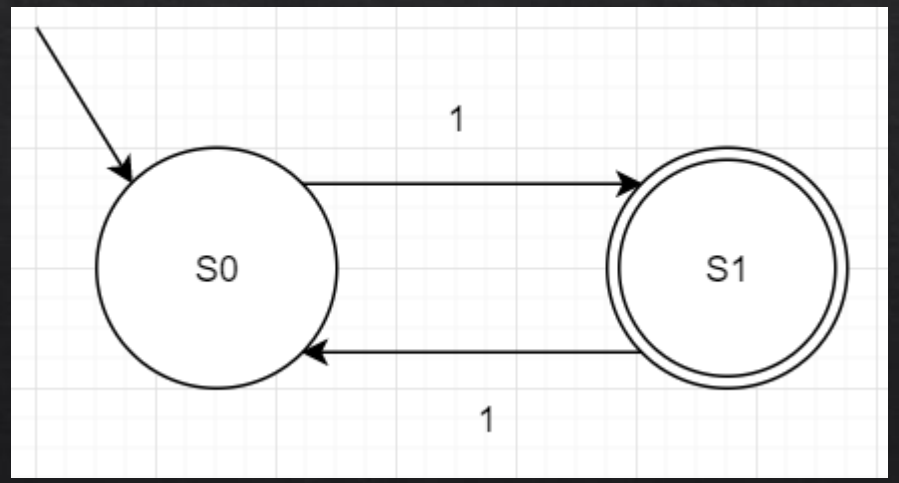
Formal Language Theory

- Set of Symbols
- Set of Rules
- Inductively generate ‘words’ (valid sequences / wff)
- Set of ‘words’ is a ‘language’
- Are there languages for music? English? DNA?

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Example

- Language: Odd natural numbers in a unary number system.
  - Symbols = {1}
  - Rules = { f(x) = x11 }



Deterministic Finite Automaton

A *finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

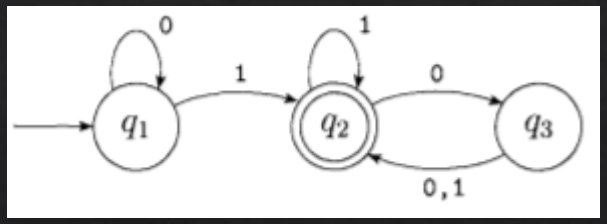
1. Q is a finite set called the *states*,
2. Σ is a finite set called the *alphabet*,
3. $\delta: Q \times \Sigma \longrightarrow Q$ is the *transition function*,[1]
4. $q_0 \in Q$ is the *start state*, and
5. $F \subseteq Q$ is the *set of accept states*.[2]

$Q = \{q_1, q_2, q_3\}$,
$\Sigma = \{0,1\}$,
δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

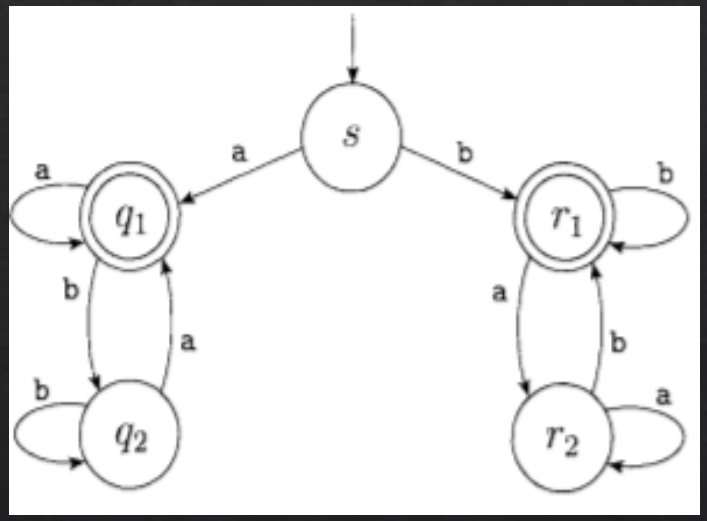
q_1 is the start state, and
$F = \{q_2\}$.

$A = \{w \mid w$ contains at least one 1 and an even number of 0s follow the last 1$\}$.



Another Example

- What does this one do?



A Definition of Computation

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and let $w = w_1 w_2 \cdots w_n$ be a string where each w_i is a member of the alphabet Σ. Then M ***accepts*** w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \ldots, n-1$, and
3. $r_n \in F$.

We say that M ***recognizes language*** A if $A = \{w | M \text{ accepts } w\}$.

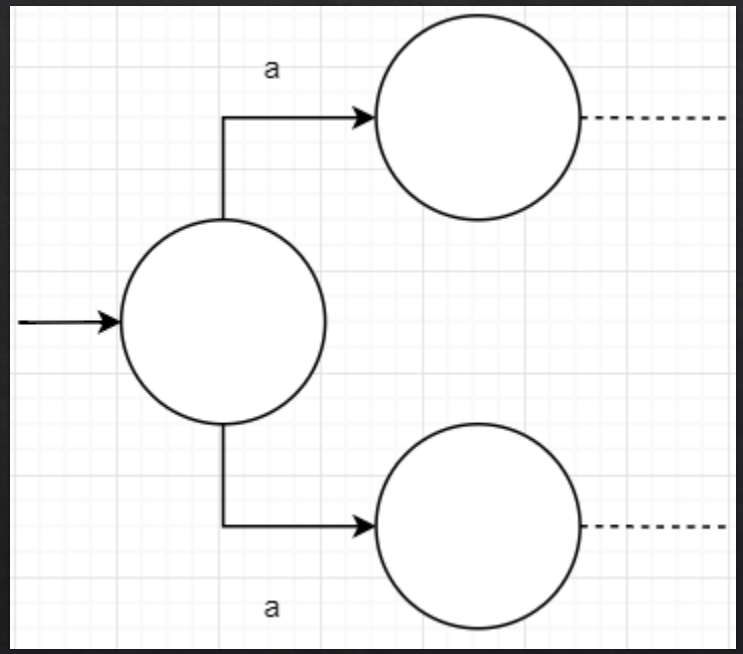
A language is called a ***regular language*** if some finite automaton recognizes it.

Operations on Regular Languages

Let A and B be languages. We define the regular operations ***union***, ***concatenation***, and ***star*** as follows.

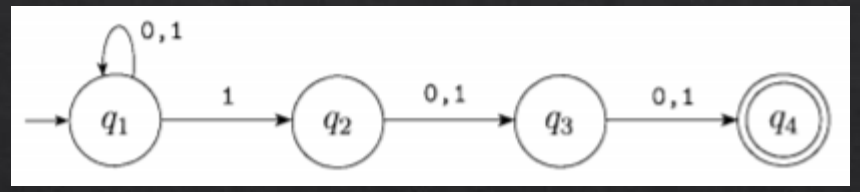
- **Union**: $A \cup B = \{x | \, x \in A \text{ or } x \in B\}$.
- **Concatenation**: $A \circ B = \{xy | \, x \in A \text{ and } y \in B\}$.
- **Star**: $A^* = \{x_1 x_2 \ldots x_k | \, k \geq 0 \text{ and each } x_i \in A\}$.

More than one choice

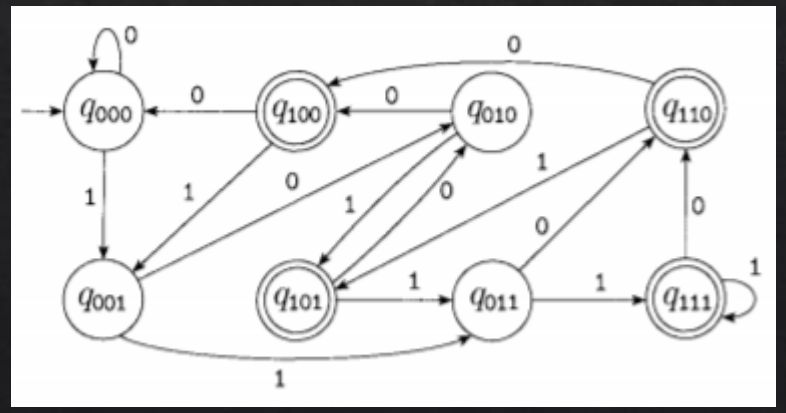


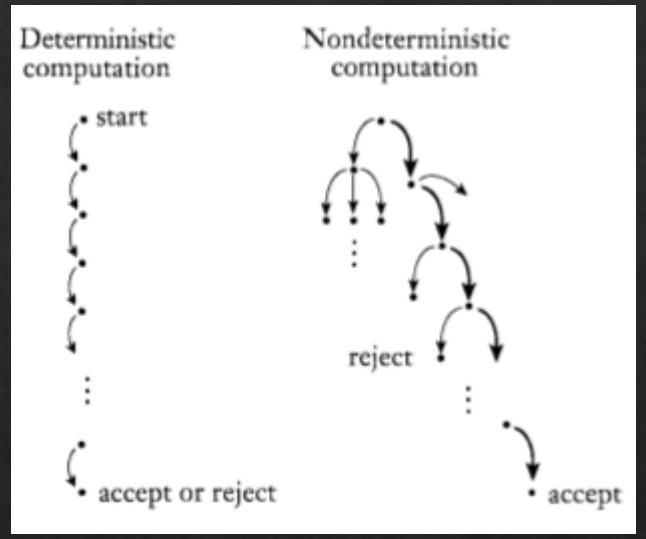
Non-Deterministic Finite Automaton

- For the same language
  - NFA

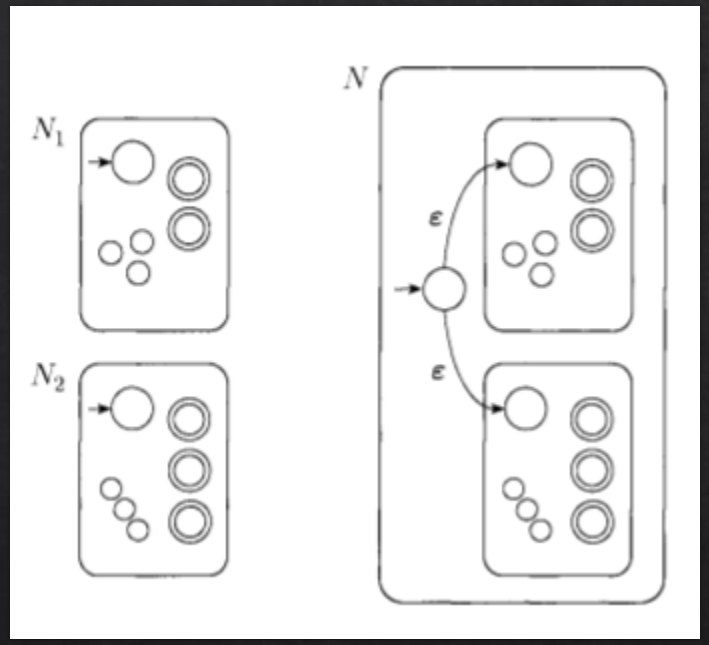


  - DFA

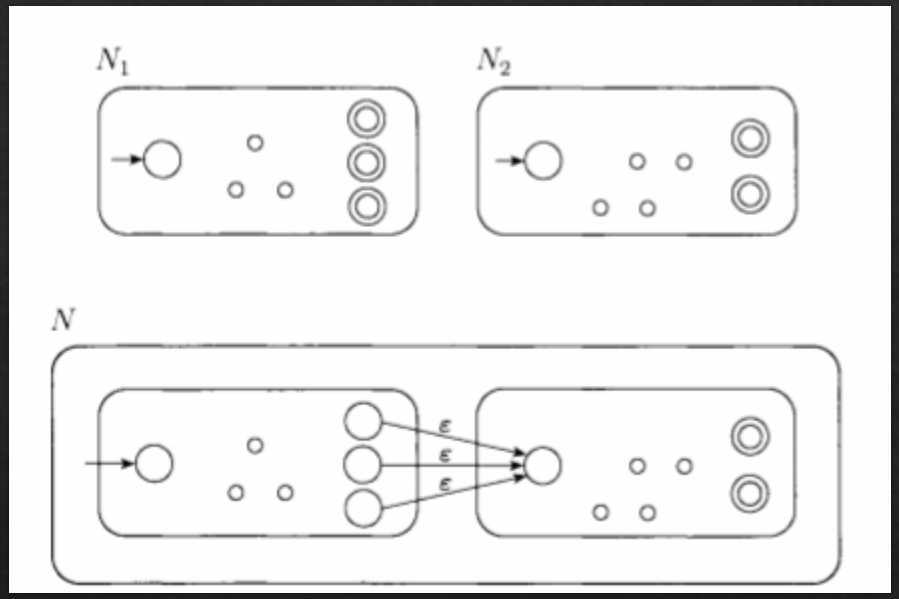




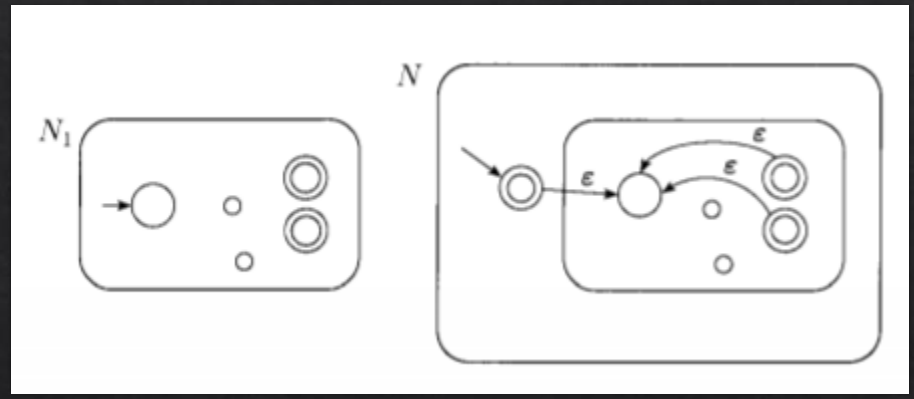
Proof of Closure Under Union



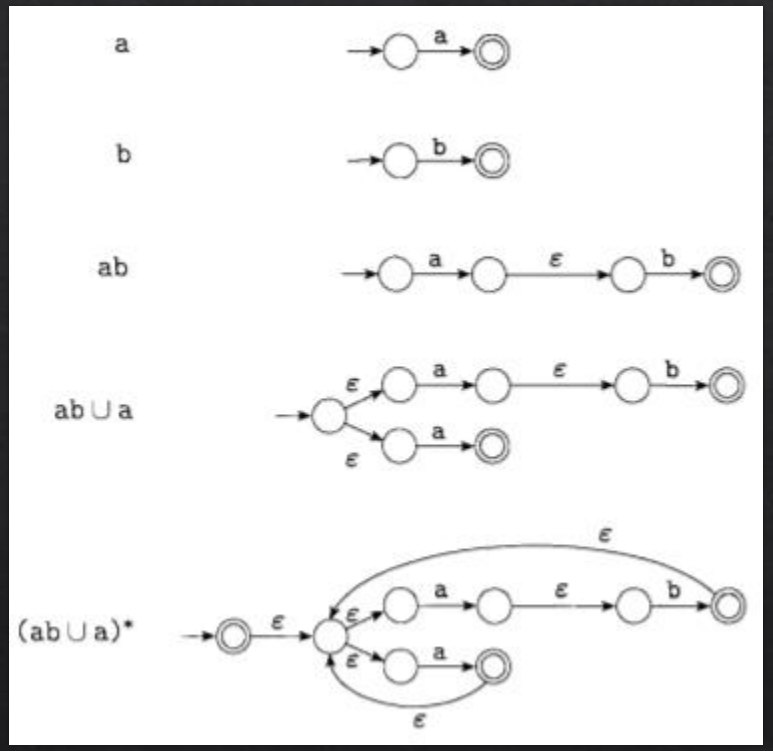
Proof of Closure Under Composition



Proof of Closure Under Kleene Star

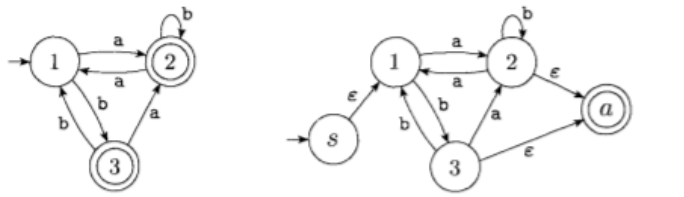


Different Notation: Regular Expressions



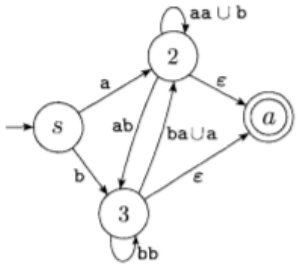
DFA to Regular Expression

In this example we begin with a three-state DFA. The steps in the conversion are shown in the following figure.

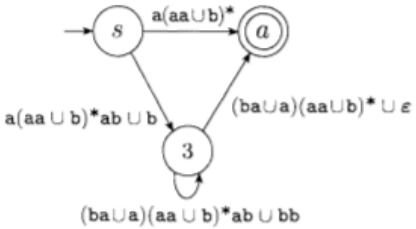


(a)

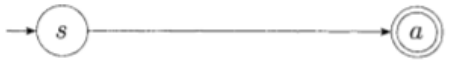
(b)



(c)



(d)



(a(aa∪b)*ab∪b)((ba∪a)(aa∪b)*ab∪bb)*((ba∪a)(aa∪b)* ∪ ε)∪a(aa∪b)*

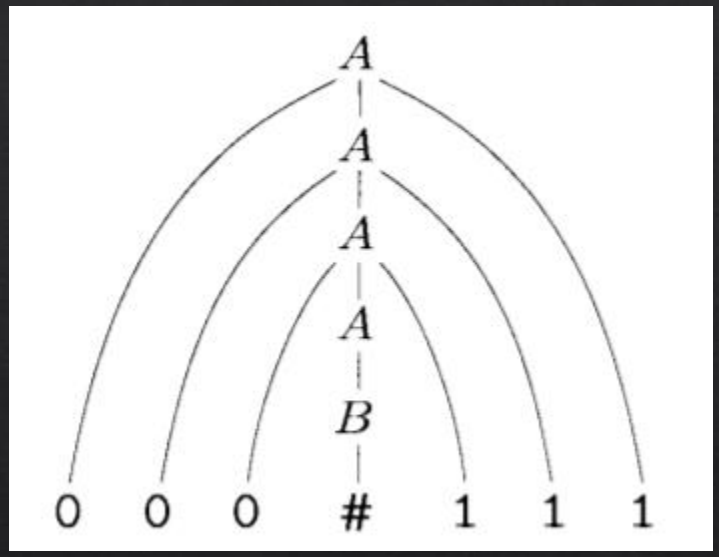
A Non-Regular Language

Let's take the language $B = \{0^n 1^n \mid n \geq 0\}$. If we attempt to find a DFA that recognizes B, we discover that the machine seems to need to remember how many 0s have been seen so far as it reads the input. Because the number of 0s isn't limited, the machine will have to keep track of an unlimited number of possibilities. But it cannot do so with any finite number of states.

Context Free Grammars

$$B = \{0^n 1^n \mid n \geq 0\}.$$

$$\begin{aligned} A &\to 0A1 \\ A &\to B \\ B &\to \# \end{aligned}$$



Context Free Grammars: English

⟨SENTENCE⟩ → ⟨NOUN-PHRASE⟩⟨VERB-PHRASE⟩
⟨NOUN-PHRASE⟩ → ⟨CMPLX-NOUN⟩ | ⟨CMPLX-NOUN⟩⟨PREP-PHRASE⟩
⟨VERB-PHRASE⟩ → ⟨CMPLX-VERB⟩ | ⟨CMPLX-VERB⟩⟨PREP-PHRASE⟩
⟨PREP-PHRASE⟩ → ⟨PREP⟩⟨CMPLX-NOUN⟩
⟨CMPLX-NOUN⟩ → ⟨ARTICLE⟩⟨NOUN⟩
⟨CMPLX-VERB⟩ → ⟨VERB⟩ | ⟨VERB⟩⟨NOUN-PHRASE⟩
⟨ARTICLE⟩ → a | the
⟨NOUN⟩ → boy | girl | flower
⟨VERB⟩ → touches | likes | sees
⟨PREP⟩ → with

```
a boy sees
the boy sees a flower
a girl with a flower likes the boy
```

Derivation Example

⟨SENTENCE⟩ → ⟨NOUN-PHRASE⟩⟨VERB-PHRASE⟩
⟨NOUN-PHRASE⟩ → ⟨CMPLX-NOUN⟩ | ⟨CMPLX-NOUN⟩⟨PREP-PHRASE⟩
⟨VERB-PHRASE⟩ → ⟨CMPLX-VERB⟩ | ⟨CMPLX-VERB⟩⟨PREP-PHRASE⟩
⟨PREP-PHRASE⟩ → ⟨PREP⟩⟨CMPLX-NOUN⟩
⟨CMPLX-NOUN⟩ → ⟨ARTICLE⟩⟨NOUN⟩
⟨CMPLX-VERB⟩ → ⟨VERB⟩ | ⟨VERB⟩⟨NOUN-PHRASE⟩
⟨ARTICLE⟩ → `a` | `the`
⟨NOUN⟩ → `boy` | `girl` | `flower`
⟨VERB⟩ → `touches` | `likes` | `sees`
⟨PREP⟩ → `with`

⟨SENTENCE⟩ ⇒ ⟨NOUN-PHRASE⟩⟨VERB-PHRASE⟩
⇒ ⟨CMPLX-NOUN⟩⟨VERB-PHRASE⟩
⇒ ⟨ARTICLE⟩⟨NOUN⟩⟨VERB-PHRASE⟩
⇒ `a` ⟨NOUN⟩⟨VERB-PHRASE⟩
⇒ `a boy` ⟨VERB-PHRASE⟩
⇒ `a boy` ⟨CMPLX-VERB⟩
⇒ `a boy` ⟨VERB⟩
⇒ `a boy sees`

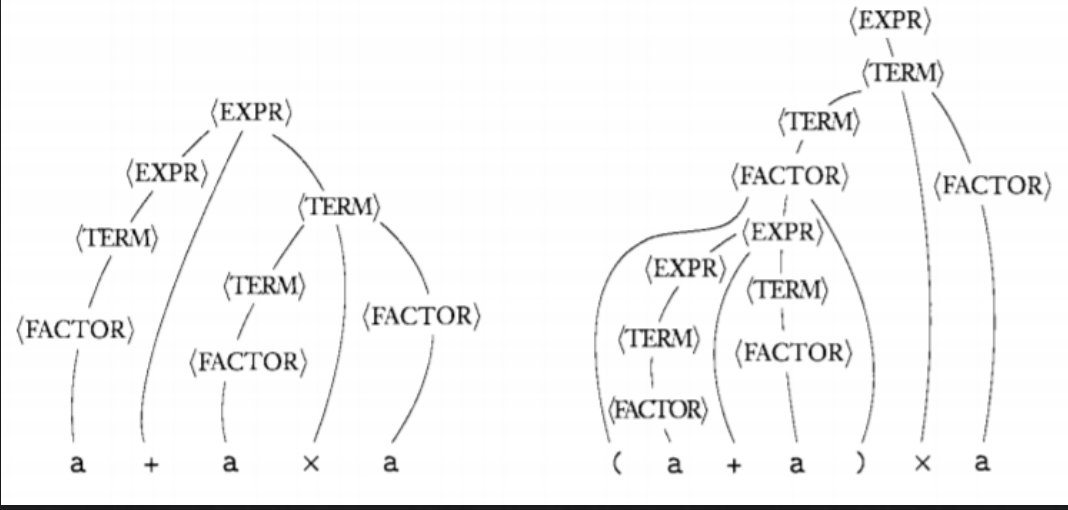
Context Free Grammars: Arithmetic Expressions

Consider grammar $G_4 = (V, \Sigma, R, \langle \text{EXPR} \rangle)$.

V is $\{\langle \text{EXPR} \rangle, \langle \text{TERM} \rangle, \langle \text{FACTOR} \rangle\}$ and Σ is {a, +, ×, (,)}. The rules are

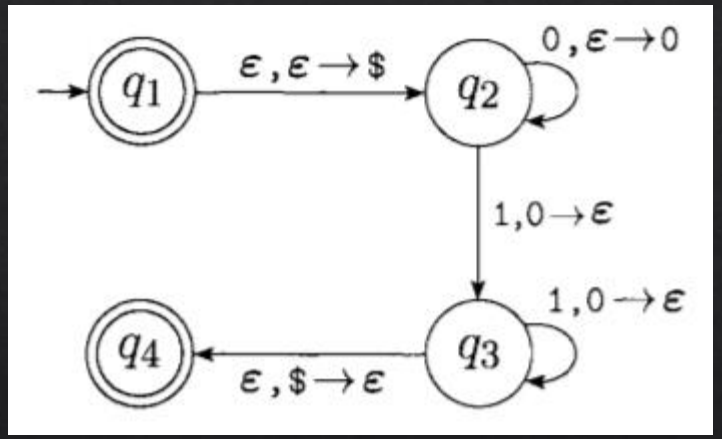
$$\begin{aligned} \langle \text{EXPR} \rangle &\rightarrow \langle \text{EXPR} \rangle + \langle \text{TERM} \rangle \mid \langle \text{TERM} \rangle \\ \langle \text{TERM} \rangle &\rightarrow \langle \text{TERM} \rangle \times \langle \text{FACTOR} \rangle \mid \langle \text{FACTOR} \rangle \\ \langle \text{FACTOR} \rangle &\rightarrow (\langle \text{EXPR} \rangle) \mid \text{a} \end{aligned}$$

The two strings a+a×a and (a+a)×a can be generated with grammar G_4. The parse trees are shown in the following figure.



Push Down Automaton

$$B = \{0^n 1^n \mid n \geq 0\}.$$



Other Machines

- The usual
  - DFA NFA – Regular languages
  - Pushdown Automata – CFG
  - Turing Machines – Recursively Enumerable Languages
- Alternatives & Variations
  - Probabilistic FA (Markov Chain with additional structure) – Stochastic Languages
  - Quantum FA -- ?? Generalization of PFA
  - Quantum TM
- Introduction to the Theory of Computation – Sipser

Lambda Calculus

- A calculus to capture the idea of applying arguments to functions.
- Computation by substitution.
- Syntax

Syntax	Name	Description
a	Variable	A character or string representing a parameter or mathematical/logical value
(λx.M)	Abstraction	Function definition (M is a lambda term). The variable x becomes bound in the expression.
(M N)	Application	Applying a function to an argument. M and N are lambda terms.

- Rules

Operation	Name	Description
(λx.M[x]) → (λy.M[y])	α-conversion	Renaming the bound (formal) variables in the expression. Used to avoid name collisions.
((λx.M) E) → (M[x:=E])	β-reduction	Substituting the bound variable by the argument expression in the body of the abstraction

- $(\lambda x.Fx) \leftrightarrow F$ Eta-conversion x can not be free in F

$$x^2 - 2 \cdot x + 5$$

$$\lambda x[x^2 - 2 \cdot x + 5]$$

$$\begin{aligned}
&(\lambda x[x^2 - 2 \cdot x + 5])2 \\
\rhd\ &\langle \text{ Substitute 2 for } x \rangle \\
&2^2 - 2 \cdot 2 + 5 \\
=\ &\langle \text{ Arithmetic} \rangle \\
&4 - 4 + 5 \\
=\ &\langle \text{ Arithmetic} \rangle \\
&5
\end{aligned}$$

Encoding Peano Arithmetic via Church Numerals

Church numerals are an extension of this. All Church numerals are functions with two parameters:

$$\lambda f \,.\, \lambda x \,.\, \textit{something}$$

The first parameter, f, is the successor function that should be used. The second parameter, x, is the value that represents zero. Therefore, the Church numeral for zero is:

$$C_0 = \lambda f \,.\, \lambda x \,.\, x$$

Whenever it is applied, it returns the value representing zero. The Church numeral for one applies the successor function to the value representing zero exactly once:

$$C_1 = \lambda f \,.\, \lambda x \,.\, fx$$

The Church numerals that follow just have additional applications of the successor function:

$$\begin{aligned} C_2 &= \lambda f \,.\, \lambda x \,.\, f(fx) \\ C_3 &= \lambda f \,.\, \lambda x \,.\, f(f(fx)) \\ C_4 &= \lambda f \,.\, \lambda x \,.\, f(f(f(fx))) \\ C_n &= \lambda f \,.\, \lambda x \,.\, f^n x \end{aligned}$$

Encoding Peano Arithmetic via Church Numerals

If we want to add 3 to 4 using Church numerals, we simply create a new Church numeral and use one of the summands as zero for the other:

$$C_{3+4} = \lambda f\,.\,\lambda x\,.\,C_3\ f\ (C_4\ f\ x)$$

C_{3+4} is a function with two parameters – just like any Church numeral – but it applies C_3 to f, the successor function, and $C_4 f x$, which now acts as value for zero. Written out, C_{3+4} is (parameters have been renamed to avoid shadowing)

$$\begin{aligned} C_{3+4} &= \lambda f\,.\,\lambda x\,.\,(\lambda f_3\,.\,\lambda x_3\,.\,f_3(f_3(f_3 x_3)))\ f\ (\lambda f_4\,.\,\lambda x_4\,.\,f_4(f_4(f_4(f_4 x_4)))\ f\ x) \\ &= \lambda f\,.\,\lambda x\,.\,f(f(f(\lambda f_4\,.\,\lambda x_4\,.\,f_4(f_4(f_4(f_4 x_4)))\ f\ x))) \\ &= \lambda f\,.\,\lambda x\,.\,f(f(f(f(f(f(f x)))))) \\ &= C_7 \end{aligned}$$

We can therefore define a function *add* that takes two Church numerals M and N and returns the sum of them:

$$\begin{aligned} \mathit{add} &= \lambda M\,.\,\lambda N\,.\,\lambda f\,.\,\lambda x\,.\,N\ f\ (M\ f\ x) \\ \mathit{add}\ C_4\ C_7 &= C_7 \end{aligned}$$

add actually has four lambdas, not just two for M and N, since the result is a Church numeral, which is a function with two parameters.

Other Encodings

$$multiply = \lambda M . \lambda N . \lambda f . \lambda x . N\ (M f)\ x$$

$$\begin{aligned} tru &= \lambda x . \lambda y . x \\ fls &= \lambda x . \lambda y . y \end{aligned}$$

$$\begin{aligned} and &= \lambda M . \lambda N . M\ (N tru\ fls)\ fls \\ or &= \lambda M . \lambda N . M\ tru\ (N\ tru\ fls) \\ not &= \lambda M . M\ fls\ tru \end{aligned}$$

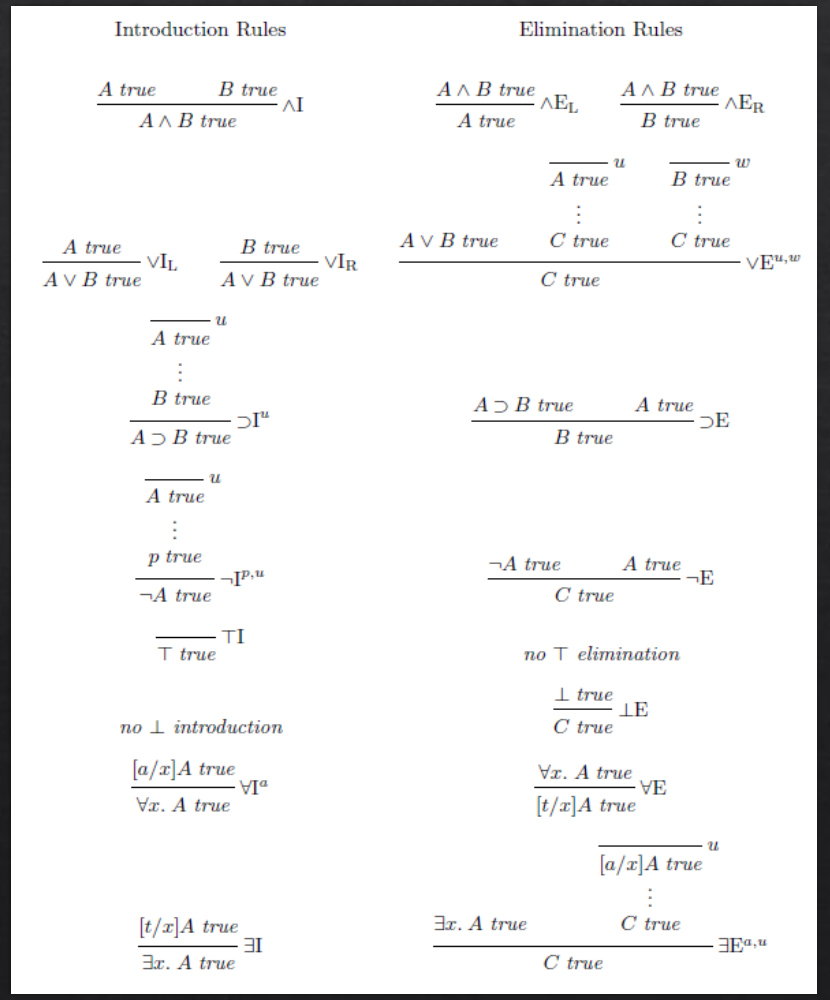
Recursion, Reduction, & Types

- Recursion
  - $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$
  - Y combinator: $Y \equiv (\lambda y.(\lambda x.y(xx))(\lambda x.y(xx)))$
- Reduction Order Matters
  - Consider $(\lambda x.\lambda y.y)[(\lambda z.zz)(\lambda z.zz)]$
  - Reduce argument yields: $(\lambda x.\lambda y.y)[(\lambda z.zz)(\lambda z.zz)]$
  - Substitution yields: $(\lambda y.y)$
- Simply Typed Lambda Calculus
  - $\lambda^{A\to A}x.x$
  - $x:A$

Logic: Perspective and Alternate Forms

- Many viewpoints, 2 prominent
  - Formalism – Hilbert
    - Higher mathematics is no more than a formal game. The statements of higher-order mathematics are uninterpreted strings of symbols. Proving such statements is no more than a game in which symbols are manipulated according to fixed rules.
  - Intuitionistic – Brouwer
    - Mathematics is essentially an activity of construction. The natural numbers are mental constructions, the real numbers are mental constructions, proofs and theorems are mental constructions, mathematical meaning is a mental construction… Mathematical constructions are produced by the *ideal* mathematician
- Classical vs Intuitionistic Logic
  - Classical: more axioms, fewer rules
  - Intuitionistic: less axioms, more rules

Intuitionistic Logic: Formulated in Gentzen-style Natural Deduction



Introduction Rules

$$\frac{A\ true \qquad B\ true}{A \wedge B\ true}\ \wedge\mathrm{I}$$

$$\frac{A\ true}{A \vee B\ true}\ \vee\mathrm{I}_{\mathrm{L}} \qquad \frac{B\ true}{A \vee B\ true}\ \vee\mathrm{I}_{\mathrm{R}}$$

$$\frac{\begin{array}{c}\overline{A\ true}^{\,u}\\ \vdots\\ B\ true\end{array}}{A \supset B\ true}\ \supset\mathrm{I}^{u}$$

$$\frac{\begin{array}{c}\overline{A\ true}^{\,u}\\ \vdots\\ p\ true\end{array}}{\neg A\ true}\ \neg\mathrm{I}^{p,u}$$

$$\frac{}{\top\ true}\ \top\mathrm{I}$$

no $\bot$ introduction

$$\frac{[a/x]A\ true}{\forall x.\ A\ true}\ \forall\mathrm{I}^{a}$$

$$\frac{[t/x]A\ true}{\exists x.\ A\ true}\ \exists\mathrm{I}$$

Elimination Rules

$$\frac{A \wedge B\ true}{A\ true}\ \wedge\mathrm{E}_{\mathrm{L}} \qquad \frac{A \wedge B\ true}{B\ true}\ \wedge\mathrm{E}_{\mathrm{R}}$$

$$\frac{A \vee B\ true \qquad \begin{array}{c}\overline{A\ true}^{\,u}\\ \vdots\\ C\ true\end{array} \qquad \begin{array}{c}\overline{B\ true}^{\,w}\\ \vdots\\ C\ true\end{array}}{C\ true}\ \vee\mathrm{E}^{u,w}$$

$$\frac{A \supset B\ true \qquad A\ true}{B\ true}\ \supset\mathrm{E}$$

$$\frac{\neg A\ true \qquad A\ true}{C\ true}\ \neg\mathrm{E}$$

no $\top$ elimination

$$\frac{\bot\ true}{C\ true}\ \bot\mathrm{E}$$

$$\frac{\forall x.\ A\ true}{[t/x]A\ true}\ \forall\mathrm{E}$$

$$\frac{\exists x.\ A\ true \qquad \begin{array}{c}\overline{[a/x]A\ true}^{\,u}\\ \vdots\\ C\ true\end{array}}{C\ true}\ \exists\mathrm{E}^{a,u}$$

Intuitionistic Logic Rules with Proof Terms

$$\frac{}{x : A \vdash x : A}\ \text{Id} \qquad \frac{\Gamma, \Delta \vdash t : A}{\Delta, \Gamma \vdash t : A}\ \text{Exchange}$$

$$\frac{\Gamma, y : A, z : A \vdash u : B}{\Gamma, x : A \vdash u[x/y, x/z] : B}\ \text{Contraction} \qquad \frac{\Gamma \vdash u : B}{\Gamma, x : A \vdash u : B}\ \text{Weakening}$$

$$\frac{\Gamma, x : A \vdash u : B}{\Gamma \vdash \lambda x.\, u : A \to B}\ \to\text{-I} \qquad \frac{\Gamma \vdash s : A \to B \qquad \Delta \vdash t : A}{\Gamma, \Delta \vdash s\,(t) : B}\ \to\text{-E}$$

$$\frac{\Gamma \vdash t : A \qquad \Delta \vdash u : B}{\Gamma, \Delta \vdash (t, u) : A \times B}\ \times\text{-I} \qquad \frac{\Gamma \vdash s : A \times B \qquad \Delta, x : A, y : B \vdash v : C}{\Gamma, \Delta \vdash \text{case } s \text{ of } (x, y) \to v : C}\ \times\text{-E}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \text{inl}\,(t) : A + B}\ +\text{-I}_1 \qquad \frac{\Gamma \vdash u : B}{\Gamma \vdash \text{inr}\,(u) : A + B}\ +\text{-I}_2$$

$$\frac{\Gamma \vdash s : A + B \qquad \Delta, x : A \vdash v : C \qquad \Delta, y : B \vdash w : C}{\Gamma, \Delta \vdash \text{case } s \text{ of inl}\,(x) \to v;\ \text{inr}\,(y) \to w : C}\ +\text{-E}$$

Linear Logic Rules

$$\frac{}{\langle A\rangle \vdash A}\ \langle\mathrm{Id}\rangle \qquad \frac{}{[A] \vdash A}\ [\mathrm{Id}] \qquad \frac{\Gamma, \Delta \vdash A}{\Delta, \Gamma \vdash A}\ \mathrm{Exchange}$$

$$\frac{\Gamma, [A], [A] \vdash B}{\Gamma, [A] \vdash B}\ \mathrm{Contraction} \qquad \frac{\Gamma \vdash B}{\Gamma, [A] \vdash B}\ \mathrm{Weakening}$$

$$\frac{[\Gamma] \vdash A}{[\Gamma] \vdash\ !A}\ !\text{-I} \qquad \frac{\Gamma \vdash\ !A \qquad \Delta, [A] \vdash B}{\Gamma, \Delta \vdash B}\ !\text{-E}$$

$$\frac{\Gamma, \langle A\rangle \vdash B}{\Gamma \vdash A \multimap B}\ \multimap\text{-I} \qquad \frac{\Gamma \vdash A \multimap B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B}\ \multimap\text{-E}$$

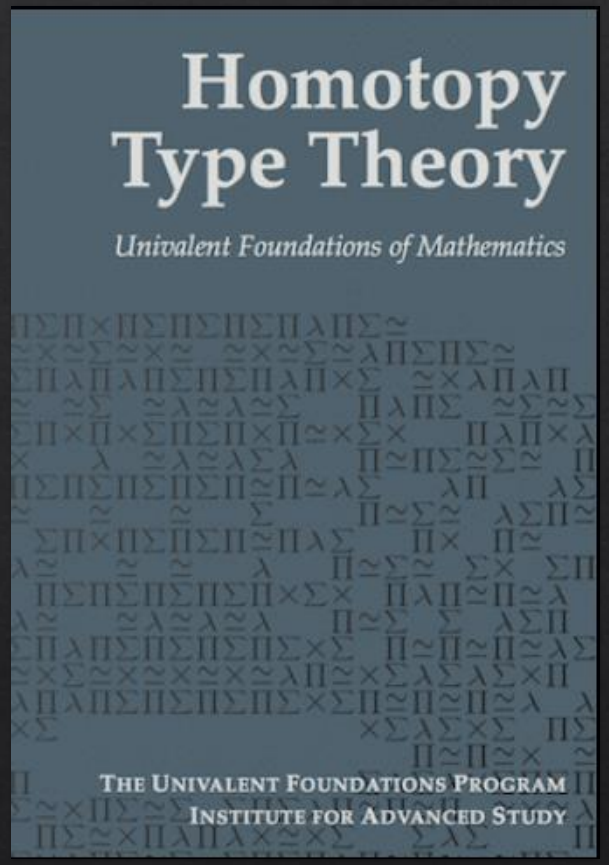
$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ \otimes\text{-I} \qquad \frac{\Gamma \vdash A \otimes B \qquad \Delta, \langle A\rangle, \langle B\rangle \vdash C}{\Gamma, \Delta \vdash C}\ \otimes\text{-E}$$

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}\ \&\text{-I} \qquad \frac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash A}\ \&\text{-E}_1 \qquad \frac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash B}\ \&\text{-E}_2$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\ \oplus\text{-I}_1 \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\ \oplus\text{-I}_2 \qquad \frac{\Gamma \vdash A \oplus B \qquad \Delta, \langle A\rangle \vdash C \qquad \Delta, \langle B\rangle \vdash C}{\Gamma, \Delta \vdash C}\ \oplus\text{-E}$$

Fig. 4. Linear logic

Recent Developments



Homotopy type theory

Homotopy type theory (HoTT) interprets type theory from a homotopical perspective. In homotopy type theory, we regard the types as "spaces" (as studied in homotopy theory) or higher groupoids, and the logical constructions (such as the product $A \times B$) as homotopy-invariant constructions on these spaces. In this way, we are able to manipulate spaces directly without first having to develop point-set topology (or any combinatorial replacement for it, such as the theory of simplicial sets). To briefly explain this perspective, consider first the basic concept of type theory, namely that the *term* a is of *type* A, which is written:

$$a : A.$$

This expression is traditionally thought of as akin to:

"a is an element of the set A".

However, in homotopy type theory we think of it instead as:

"a is a point of the space A".

Similarly, every function $f : A \to B$ in type theory is regarded as a continuous map from the space A to the space B.

Vladimir Voevodsky

- The first component is a formal deduction system: a language and rules of manipulating sentences in this language that are purely formal, such that a record of such manipulations can be verified by a computer program.
- The second component is a structure that provides a meaning to the sentences of this language in terms of mental objects intuitively comprehensible to humans.
- The third component is a structure that enables humans to encode mathematical ideas in terms of the objects directly associated with the language.

Review

- Introduction to Logic.
- Brief History of Logic
- Logic's Influence on the Construction of the Machine.
- Formal Language Theory & Automata Theory
- Lambda Calculus
- Intuitionistic Logic & Natural Deduction
- Curry Howard Isomorphism.
- Bleeding Edge: Homotopy Type Theory.