



CAB203: Discrete Structures Definition

Author not responsible for consequences caused by any inaccuracies.

Markdown Setup

Open in Visual Studio Code and install Markdown Preview Enhanced:

<https://marketplace.visualstudio.com/items?itemName=shd101wyy.markdown-preview-enhanced>

Typesetting Powered by K^AT_EX

K^AT_EX documentation: <https://katex.org/>

which uses L^AT_EX-*like* syntax to display mathematical symbols

Equivalence

Equals: $=$ (1)

Greater than: $>$ (2)

Greater than or equal to: \geq (3)

Less than: $<$ (4)

Less than or equal to: \leq (5)

Not equal to: \neq (6)

Modular equivalence: \equiv (7)

Operators

$$\text{Multiply: } \cdot \quad (8)$$

$$\text{Mod: } \bmod \quad (9)$$

$$\text{Implication: } \models \quad (10)$$

$$\text{NOT: } \neg \quad (11)$$

$$\text{AND, } \wedge \quad (12)$$

$$\text{OR, } \vee \quad (13)$$

$$\text{XOR, } \oplus \quad (14)$$

$$\text{IF..THEN: } \rightarrow \quad (15)$$

$$\text{IF AND ONLY IF: } \leftrightarrow \quad (16)$$

Formula Classification

$$\text{Tautologies: } \textit{always} T \quad (17)$$

$$\text{Contradictions: } \textit{always} F \quad (18)$$

$$\text{Contingent formulas: } T \textit{ or } F \text{ depending on variables} \quad (19)$$

$$\text{Satisfiable formulas: } \textit{tautologies or contingent formulas} \quad (20)$$

Set Theory Abstractions

$$\text{Set: } S \quad (21)$$

$$\text{Universe: } U \quad (22)$$

$$\text{Member in set: } \in \quad (23)$$

$$\text{Not member in set: } \notin \quad (24)$$

$$\textbf{Real, any number: } \mathbb{R} \quad (25)$$

$$\textbf{Integer, only whole: } \mathbb{Z} \quad (26)$$

$$\textbf{Natural}_0, \text{ whole non-negatives: } \mathbb{N}_0 \quad (27)$$

$$\textbf{Natural}_1, \text{ whole positives: } \mathbb{N}_1 \quad (28)$$

$$\text{Subset: } A \subseteq B \quad (29)$$

$$\text{Proper subset: } A \subset B \quad (30)$$

$$\text{Equality through subsets: } S \subseteq T \ \& \ T \subseteq S \quad (31)$$

Set Constructors

Set listing:

$$SMALLPRIMES = \{ 1, 2, 3, 5, 7 \} \quad (32)$$

Implied pattern: (*bad practice*)

$$EVENS = \{ 2, 4, 6, 8, \dots \} \quad (33)$$

Setbuilder notation: “set comprehension”

Subset of elements, that match a condition

$$\{ x \in S : \phi(x) \} \quad (34)$$

$$SQUARES = \{ x \in \mathbb{Z} : x = y^2 \text{ for some } y \in \mathbb{Z} \} \quad (35)$$

Setbuilder notation: “replacement”

Apply a theory to each member and collect results

$$\{ f(x) : x \in S \} \quad (36)$$

$$SQUARES = \{ x^2 : x \in \mathbb{Z} \} \quad (37)$$

Set Operators

Union: everything in either

Must define $A \cup B$ to be the smallest set, such that $A \subseteq S$ & $B \subseteq S$

$$A \cup B = \{ x : x \in A \vee x \in B \} \quad (38)$$

Intersection: everything in *both* sides

$$A \cap B = \{ x \in A : x \in B \} \quad (39)$$

Difference: remove items in one set from the other

$$A \setminus B = \{ x \in A : x \notin B \} \quad (40)$$

Predicates

Existential quantification, *there exists*: \exists (41)

Universal quantification, *for all*: \forall (42)

Exponents

Exponent in a^3 : 3 (43)

Base in a^3 : a (44)

kilo: 2^{10} (45)

mega: $2^{20} = (2^{10})^2$ (46)

giga: $2^{30} = (2^{10})^3$ (47)

tera: $2^{40} = (2^{10})^4$ (48)

peta: $2^{50} = (2^{10})^5$ (49)

exa: $2^{60} = (2^{10})^6$ (50)

Laws of Exponents

$$(ab)^n = a^n \cdot b^n \quad (51)$$

$$a^m \cdot a^n = a^{m+n} \quad (52)$$

$$a^{m-n} = \frac{a^m}{a^n} \quad (\text{when } a \neq 0) \quad (53)$$

$$a^{-n} = \frac{1}{a^n} \quad (\text{when } a \neq 0) \quad (54)$$

$$a^0 = 1 \quad (55)$$

$$(a^m)^n = a^{m \cdot n} \quad (56)$$

Laws of Logarithms

$$\log_a 1 = 0 \quad (57)$$

$$\log_a a = 1 \quad (58)$$

$$\log_a (x \cdot y) = \log_a x + \log_a y \quad (59)$$

$$\log_a x^y = y \log_a x \quad (60)$$

$$\log_a \frac{1}{y} = -\log_a y \quad (61)$$

$$\log_a \frac{x}{y} = \log_a x - \log_a y \quad (62)$$

$$\log_b x = (\log_b a) \cdot \log_a x \quad (63)$$

Ceiling and Floor

Ceiling, round up: $\lceil a \rceil$ (64)

Floor, round down: $\lfloor a \rfloor$ (65)

Bits

Bit string: \bar{x} (66)

Mod: mod (67)

NOT: \sim (68)

AND: $\&$ (69)

OR: $|$ (70)

XOR: \wedge (71)

Scientific Notation

Sign: \pm (72)

Significant digits: $a.bc$ (73)

Exponent: e (74)

Base: 10 (75)

IEEE Half-Precision

$$\overbrace{0}^s \overbrace{10101}^e \overbrace{11010101010}^f \quad (76)$$

$$\text{Sign: } s(1 - \text{bit}) \quad (77)$$

$$\text{Exponent: } e(5\text{bits}) \quad (78)$$

$$\text{Significant digits: } f(10 \text{ bits dropping leading } 1) \quad (79)$$

Recursion

Factorial function on \mathbb{N} defined as:

$$n! = \prod_{j=1}^n j = 1 \cdot 2 \cdots (n-1) \cdot n \quad (80)$$

Also $n!$ recursively:

$$n! = \begin{cases} 1 & : n = 1 \\ n(n-1)! & : n > 1 \end{cases} \quad (81)$$

Fibonacci sequence:

$$f(n) = \begin{cases} 1 & : n = 1 \\ 1 & : n = 2 \\ f(n-1) + f(n-2) & : n > 2 \end{cases} \quad (82)$$

Python

```
"""
```

MODULO OPERATOR

```
"""
```

```
print(17 % 4) # Prints 1 to the console  
# 1
```

```
"""
```

EXPONENTS AND LOGARITHMS

```
"""
```

```
>>> import math          # Must import math library  
>>> math.log2(8)          # log2 means log base 2 and returns a float (decimal)  
# 3.0  
>>> 2 ** 3                # ** is exponentiation  
# 8  
>>> math.log2(100) / math.log2(10) # base transformation  
# 2.0
```

```
"""
```

UTF-8

```
"""
```

```
>>> ord('A')             # Convert character to UTF code point  
# 65  
>>> chr(65)              # Convert UTF code point to character  
# 'A'
```

```
"""
```

HEXADECIMAL

```
"""
```

```
>>> hex(65532)            # Hex string from integer  
# '0xffffc'  
>>> "The number is {:x}".format(65532) # Alternative method  
# 'The number is fffc'  
>>> 0xffffc               # Hex literal integer  
# 65532
```

```
"""
```

NUMBERS

```
"""
```

```
>>> 9/2      # Regular division always returns float
# 4.5
>>> 9//2     # Floor division returns integer
# 4
```

```
"""
```

FIBONACCI

```
"""
```

```
def F(n):
    if n == 1: return 1
    elif n == 2: return 1
    else: return F(n-1) + F(n-2)
```

```
"""
```

SET THEORY

```
"""
```

```
>>> S = {1,2,3}; T = {1,3,5}      # Braces define sets
>>> S.add(4); print(S)           # Sets are mutable
# {1, 2, 3, 4}
>>> S.remove(4); print(S)
# {1, 2, 3}

>>> 1 in S                        # Testing membership
# True
>>> 4 in S
# False

>>> S.issubset({1,2,3,4,5})       # Testing subset
# True
>>> S <= {1,2,3,4,5}             # Alternative subset test
# True

>>> S.union(T)                   # Using union
# {1, 2, 3, 5}
>>> S|T                          # Alternative union
```

```

# {1, 2, 3, 5}
>>> S.union(T, {8,9}, {10,11})    # Multiple union
# {1, 2, 3, 5, 8, 9, 10, 11}
>>> someSets = [S, T, {8,9}, {10,11}]
>>> set.union(*someSets)           # Splat operator
# {1, 2, 3, 5, 8, 9, 10, 11}

>>> S.intersection(T)              # Splat and multi-sets would also work
# {1, 3}
>>> S & T                          # Alternative intersection
# {1, 3}

>>> S - T                          # Testing difference
# {2}

>>> S.isdisjoint(T)                # is S & T empty?
# False

>>> len(S)                         # Size of S
# 3

# Setbuilder notation combines of comprehension and replacement:
>>> S = {0, 2, 4, 6, 8}
>>> def p(x): return x % 2 == 0
...
>>> def p(x): return x * 5
...
>>> { f(x) for x in S if p(x) }
# {0, 40, 10, 20, 30}
>>> { s * 5 for s in range(0, 10) if s % 2 == 0 }
# {0, 40, 10, 20, 30}

"""
PREDICATES AND QUANTIFIERS
"""

>>> def p(x,y):                    # Any function that returns T/F to be used as predicate
    return x >= y
>>> p(2,1)
# True

>>> def p(x): return x >= 0
>>> S = { -1, 0, 1 }; T = { 0, 1, 2 }
>>> Sp = [ p(x) for x in S ]      # List of booleans

```

```
>>> Tp = [ p(x) for x in T ] # List of booleans
>>> all(Tp)                  # Check for all: ALL True
# True
>>> all(Sp)                  # Check there exists: at least ONE True
# True
>>> any(p(x) for x in S)     # Generator expression
# True
```