# Row/Column Dominance

# Implementation Details

```python
class Record:
    binary: str
    used: bool = False
    one_count: int
    representing_numbers: Set[int]

    def __init__(self, binary, representing_numbers):
        self.binary = binary
        self.one_count = self.binary.count("1")
        self.representing_numbers = representing_numbers

    def xor(self, x: str):
        # equivalent to hamming distance
        lhs = self.binary[:]
        rhs = x[:]
        ret_string = ""
        for lhs_char, rhs_str in zip(lhs, rhs):
            if lhs_char == rhs_str:
                ret_string += "0"
            else:
                ret_string += "1"

        return ret_string

    def __repr__(self):
        return f"({self.binary}: {','.join([str(x) for x in sorted(self.representing_numbers)])})"

    def __str__(self):
        return_string = ""
        for index, char in enumerate(self.binary):
            if char == "-": continue
            elif char == "0":
                return_string += f"x{index + 1}'"
            else:
                return_string += f"x{index + 1}"

        return return_string

    def __eq__(self, other):
        return self.binary == other.binary
```

- binary: string representation
- used: was this term used during combination?
- one_count: how many ‘1’s are present?
- representing_numbers: what terms does this cover?

- xor: hamming distance
- rest are for debugging and printing

# Function details

```
n_vars = 4; minterms = {0, 2, 5, 6, 7, 8, 10, 12, 13, 14, 15}; dontcares = set()
keep_running = True
n_iterations = 0

pis = find_pi(n_vars, minterms, dontcares)
epis = set()
while keep_running:
    print("Iteration", n_iterations)

    #pis = pis.intersection(find_pi(n_vars, minterms, dontcares))

    epis = epis.union(find_epi(pis, dontcares))  # calculate epi

    pis = pis - epis

    epi_covered_numbers = set()
    for epi in epis:
        epi_covered_numbers.update(epi.representing_numbers)

    if (minterms - dontcares).issubset(epi_covered_numbers):  # check if EPI covers all implicants
        pis = set()
        break

    excluded_columns = column_dominance(pis, epis, minterms, dontcares)  # column dominance
    if not excluded_columns: keep_running = False
    dontcares = dontcares.union(excluded_columns)
    excluded_terms = row_dominance(pis, epis, minterms, dontcares)  # row dominance
    for binary in excluded_terms:
        remove_something = None
        for record in pis:
            if record.binary == binary:
                remove_something = record
                break

        if remove_something:
            pis.remove(remove_something)

    if not excluded_terms:
        keep_running = keep_running | False
```

1) Find PI and EPI
2) Check if EPI covers all minterms
3) Perform column dominance, removing columns
4) Perform row dominance, removing Pis
5) Run until no further updates are possible

# Column Dominance

```python
def column_dominance(pi_records: Set[Record], epi_records: Set[Record], minterms: Set[int], dontcares: Set[int]) -> Set[int]:
    # returns columns to exclude
    epi_covered_terms = set()
    for epi in epi_records:
        epi_covered_terms.update(epi.representing_numbers)

    target_columns = (minterms - dontcares) - epi_covered_terms
    column_dict = {}
    for column in target_columns:
        column_dict[column] = set()
    for pi in pi_records:
        if pi in epi_records: continue
        for number in pi.representing_numbers:
            if number in                                                    ):
                column_di
                Instance attribute representing_numbers of DLD2022.epi.Record
                representing_numbers: set[int] = representing_numbers

    for col1, cover1 in column_dict.items():
        if not col1: continue
        for col2, cover2 in column_dict.items():
            if col1 == col2: continue
            if not cover2: continue
            if cover2.issubset(cover1):
                print(f"minterm {col1} dominates minterm {col2}")
                if col1 in target_columns:
                    target_columns.remove(col1)

    return (minterms - dontcares) - epi_covered_terms - target_columns
```

Record data structure makes it easy to compare implicants Just check if representing numbers of an implicant is a subset of another implicant

# Row Dominance

```python
def row_dominance(pi_records: Set[Record], epi_records: Set[Record], minterms: Set[int], dontcares: Set[int]) -> Set[str]:
    # returns binary strings of records to exclude
    epi_covered_numbers = set()
    for epi in epi_records:
        epi_covered_numbers.update(epi.representing_numbers)

    excluded_terms = set()

    for c1_record in pi_records - epi_records:
        for c2_record in pi_records - epi_records:
            if c1_record.binary == c2_record.binary: continue
            c1_cover = (c1_record.representing_numbers - dontcares) - epi_covered_numbers
            c2_cover = (c2_record.representing_numbers - dontcares) - epi_covered_numbers

            if c2_cover.issubset(c1_cover) and c1_record.binary not in excluded_terms:
                print(f"{c1_record} dominates {c2_record}")
                excluded_terms.add(c2_record.binary)

    return excluded_terms
```

Even easier for row dominance!

# Example 1

- From the lecture notes
- $f(x1,...,x4) = \sum(m(0,4,8,10,11,12) + d(13,15))$
- Result:

```
Iteration 0
x1x2'x3 dominates x1x2x4
x1x2'x3 dominates x1x2'x4'
x1x2'x3 dominates x1x2x3'
x1x2'x3 dominates x1x3x4
pi: {(101-: 10,11)}
epi: {(--00: 0,4,8,12)}
final terms: {(--00: 0,4,8,12), (101-: 10,11)}
standard form: f = x3'x4' + x1x2'x3

Process finished with exit code 0
```

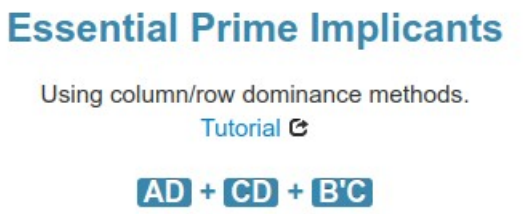
# Example 2

- F = m(2, 3, 7, 9, 11, 13) + d(1, 10, 15)

```
Iteration 0
final terms: {(-01-: 2,3,10,11), (--11: 3,7,11,15), (1--1: 9,11,13,15)}
standard form: f = x2'x3 + x3x4 + x1x4

Process finished with exit code 0
```

Verify:



# Example 3

- F = m(0, 2, 5, 6, 7, 8, 10, 12, 13, 14)

```
Iteration 0
minterm 14 dominates minterm 12
minterm 14 dominates minterm 6
x1x4' dominates x1x2
x2x3 dominates x3x4'
pi: {(1--0: 8,10,12,14), (-11-: 6,7,14,15)}
epi: {(-0-0: 0,2,8,10), (-1-1: 5,7,13,15)}
Iteration 1
final terms: {(1--0: 8,10,12,14), (-0-0: 0,2,8,10), (-11-: 6,7,14,15), (-1-1: 5,7,13,15)}
standard form: f = x1x4' + x2'x4' + x2x3 + x2x4

Process finished with exit code 0
```

Verify: