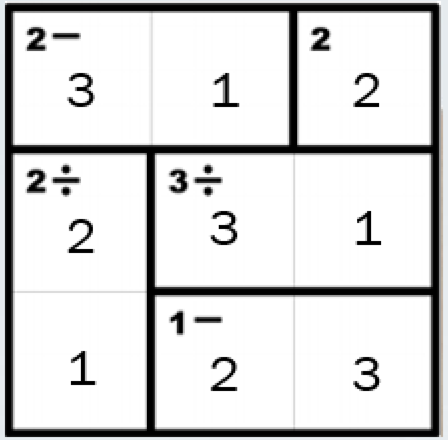


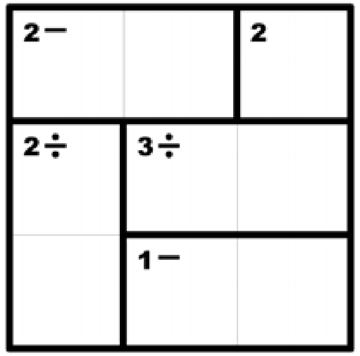
107: CONFIRMING KENKEN

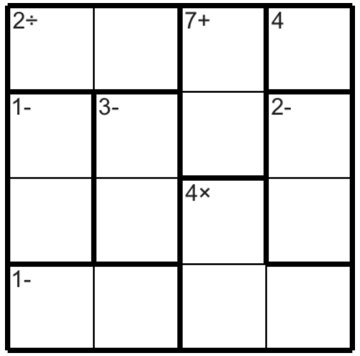
LUKAS BAUER, MICHAEL ASSHETON-SMITH, NICOLE OOI, ALEX INGHAM

PROBLEM

KenKen is a popular arithmetic and logic puzzle that appears in 150 publications worldwide and is the favorite logic game of members of this project.

- Like sudoku, a successful solution adheres to a defined set of conditions
- This project aims to determine whether a given board is a good KenKen board. A "good" KenKen board has only 1 valid configuration. It will also rate the difficulty of the board by measuring the time it took to solve.
- A KenKen configuration is valid if the following rules are met:
  - Each column contains the natural numbers between 1 and the length of the board, inclusive.
  - Each row contains the natural numbers between 1 and the length of the board, inclusive.
  - The operator in the top left of each bolded region can be applied to the numbers in that region to result in the number in the top left of that region – in either direction (ie. Left to right or up to down)

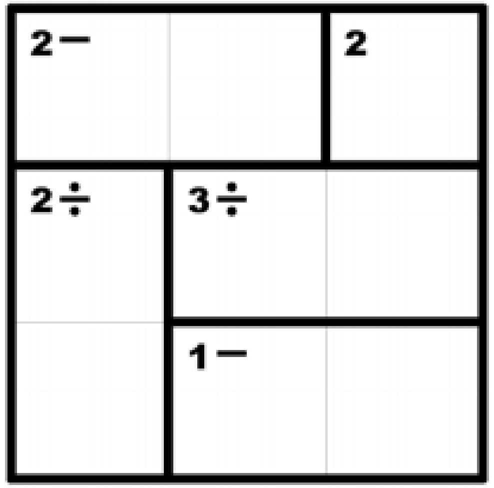




*KenKen boards only have 1 correct solution,

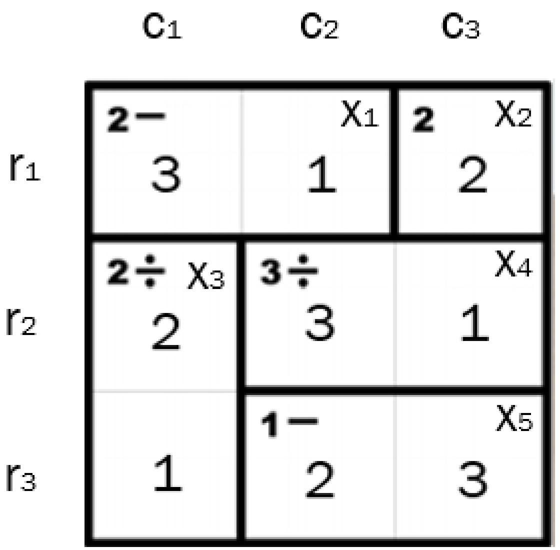
PROPOSITIONS

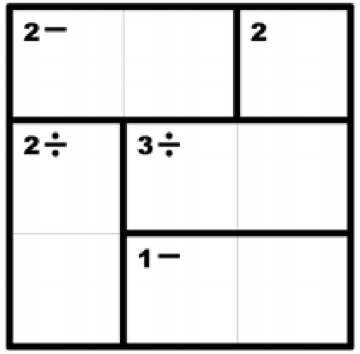
- r_i: This is true when the row i has all the required numbers (1,2,3)
- c_j : This is true when column j has all the required numbers (1,2,3)
- $O_{x,o,n}$: This is true if the region x uses the mathematical operator o and evaluates to n
  - E.g. $O_{1,-}$ is true
- $m_{i,j,k}$: This is true if the square at row i and column j, or (i,j) contains a number k
  - E.g. $m_{1,1,1}$ is false but $m_{2,1,1}$ is true

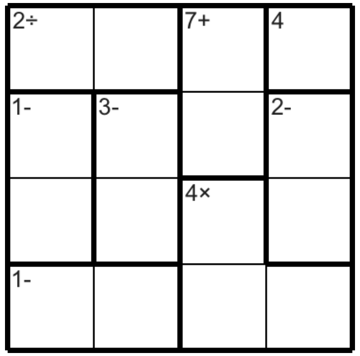


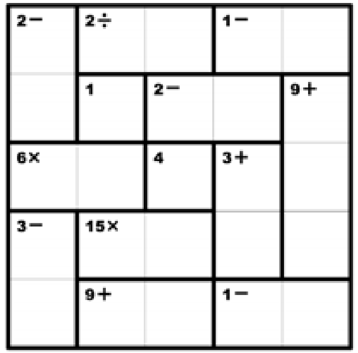
CONSTRAINTS

- Each square can only contain the numbers 1 – N, N being the size of the board
- Each region’s arithmetic constraints will be correctly met in the final solution
- The board’s smallest size is 3x3 and the largest size it can go up to is 4x4
- The board can be solved in a way that doesn’t break any of the game’s rules (ie. Each row and column is satisfied with all the numbers, and region constraints are met)









IDEAS FOR MODEL EXPLORATION

- **Solution Analysis:**
  - Use the Python library to provide a solution to the board
  - Time how long it took to provide a solution, and create a rating from it
  - Count the number of possible solutions
    - 1 solution = valid KenKen board
    - > 1 solution = invalid KenKen board
- **Testing Constraints:**
  - Test if the constraint that each square must contain a number from (1-N) is redundant
    - It may be covered by propositions r_i, c_j where each row and column must contain all numbers from (1-N)

FIRST-ORDER EXTENSION: PROPOSITIONS

- This model using predicate logic could be extended to incorporate first-order logic and ultimately assist in the verification of KenKen boards. Having universal quantifiers would allow us to create logical statements with rows and columns represented as predicates (seen below), as well as having predicates to map from points on the board (i, j) to the integer in that square.
- One(i, j) = True if the integer at (i, j) is 1. A predicate like this will be created for each number from 1-N
- R(i) = True if the row i contains all integers from 1 to N (With N being the size of the board)
- C(j) = True if the row j contains all integers from 1 to N
- O(x, o, n) = True if operator o results in n on all values in region x
- A(x, o, n) = True if operator o is addition, and the sum of all numbers in region x results in n. Predicates M, S and D would each be defined the same but for respective operators multiplication, subtraction and division.

FIRST-ORDER EXTENSION: CONSTRAINTS

- Each square must have valid numbers from 1 to N, N = 3 in this example:
  - $\forall$ i ($\forall$ j (One(i, j) $\vee$ Two(i, j) $\vee$ Three(i, j)))
- Each row and column are satisfied with numbers from 1 to N, N = 3 in this example:
  - $\forall$i R(i) $\rightarrow$ (One(i, x) ^ Two(i, y) ^ Three(i, x))
  - $\forall$j C(j) $\rightarrow$ (One(x, j) ^ Two(y, j) ^ Three(z, j))
- Each region must have its operator and result satisfied:
  - $\forall$ x O(x, o, n) $\longrightarrow$ A(x, o, n) $\vee$ S(x, o, n) $\vee$ M(x, o, n) $\vee$ D(x, o, n)

FEEDBACK REQUESTED:

- 1. Feedback on how to use Boolean variables to represent the arithmetic operations, constraints using these operators and a finite range of numbers, and ranges of values
- 2. Feedback on how to represent a region
- 3. How would you extend the model if this was your project setting? We are trying to have extensions feasible enough to be implemented in python with reasonable complexity – scaling the board to more than 4x4 takes too long to run on our computers unfortunately.
- 4. Feedback on our JAPE sequents, what other conclusions could we arrive at given the premises?

END