KLEE and Memory

For each memory allocation, KLEE creates

- **A Memory object (w. metadata)**
  - Including a regular pointer to a memory region (called *external memory*), used when invoking external functions

- **An objectState object (contains symbolic or concrete values stored)**
  - Symbolic values and offsets handled by theory of arrays. //example1.c

  - e.g., \( \text{read}((\text{write}(a,42,x),42) = x \)
  - with optimizations (of course)
  - All bytes in the objectState represented symbolically (as bitvectors)
KLEE has problems with:

• Symbolic-size allocated memory object (w. metadata)
  – Symbolic sizes are immediately concretized

• When a symbolic pointer is dereferenced, the symbolic state is forked
  – One symbolic state for each memory object where the pointer can point

• Different allocations (corresponding to different object states) are handled separately.
Sudoku in KLEE:

- Allocate 81 bytes. (9 by 9 char array)
- Encode that a row of 9 bytes is all-different:
  - If the row is $i_1i_2i_3i_4i_5i_6i_7i_8i_9$
  - Encode that these are all 9 different digits by
    \[2^{i_1} + 2^{i_2} + 2^{i_3} + 2^{i_4} + 2^{i_5} + 2^{i_6} + 2^{i_7} + 2^{i_8} + 2^{i_9} = 1111111110 \text{ (base 2)} = 0x3FE\]
  - This can be encoded as
    \[1 << i_1 \mid 1 << i_2 \mid \ldots \ldots \mid 1 << i_9 = 03XFE\]