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., read((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 $i_1\, i_2\, i_3\, i_4\, i_5\, i_6\, i_7\, i_8\, i_9$
  - Encode that these are all 9 different digits by

    $2^{i1} + 2^{i2} + 2^{i3} + 2^{i4} + 2^{i5} + 2^{i6} + 2^{i7} + 2^{i8} + 2^{i9}$ = 1111111110 (base 2) = 0x3FE
  - This can be encoded as

    $1 << i_1 \mid 1 << i_2 \mid \ldots\ldots \mid 1 << i_9$ = 03XFE