Warm-ups

- Domain of Troll
  - Description language for arbitrary dice rolls
  - Simulations of dice rolls from these descriptions
  - Calculation of probability distributions for dice rolls

- Examples of syntax
  - $5 < 3d6$: rolls 3 6-sided dice and filters to only keep rolls greater than 5
  - repeat x := 2d12 until sum x > 8: rolls 2 12-sided dice repeatedly until the sum of those dice is greater than 8
  - min 5d10: rolls five 10 sided dice and selects the minimum value.

- Intended users
  - Game designers (use probability calculations to make sure game is balanced, use syntax to convey dice rolling procedures to players)
  - Game players (use syntax to understand dice rolling procedures, use simulator to run dice rolls for game)

- Goals
  - Improve and standardize dice roll descriptions
    - Many games use notation like $3d6$ for simple rolls and most describe more complex rolls using natural language
  - Calculate probability distributions efficiently
    - Calculations could involve massive numbers of cases to calculate probability for
  - Intuitive syntax for non-programmers
    - Loops, filters, set notation, and other features might be immediately apparent to programmers but need to have very clear syntax so that an average gamer can easily understand the dice roll being described
  - Easy to read for non-programmers, especially for simple rolls
  - Precision in specifying dice rolls
  - Safe language for dice engines
  - Allow rapid prototyping of new games

Design Evaluation

- Language features
  - Dice roll primitive (e.g. $3d6$)
  - Loops (repeat...until)
  - Filters ($5 < \{ \}$)
  - Let bindings ($x := d6; x$)
  - Conditionals ($x := d6; \text{if } x=6 \text{ then } \{x,x\} \text{ else } \{x\}$)
Repeated expressions \((3\#(2d6))\)
- Set literals \(\{1, 2, 3, 4, 5, 6\}\)
- Set union \((\{1, 2, 3\} \@ \{4, 5, 6\})\)
- Multisets to represent rolls
- Built-in operations: count, sum, min, max, ...
- Probability distribution calculations

- Author motivates adequacy of set of features by describing many case studies and showing how easily they can be expressed in Troll
- Troll programs are not guaranteed to terminate
  - Loop with non-random condition that will never be satisfied
  - Mutual Recursion and Turing Completeness?
- Implementation of Troll
  - Implemented in Moscow ML
    - A good choice for implementing a language because of it's support for algebraic data types and pattern matching
  - Troll employs many algebraic optimizations and simplifications to make calculation of probability distributions more efficient
- Impact of intended users on the design
  - Simpler and more non-programmer-friendly syntax
    - Let bindings changed to look like standard variable assignment
    - Change to infix syntax for filters
    - \(3d6\) syntax instead of \(3\#d6\) from Roll
    - Multiset Literals (curly brace notation) added
- Author's evaluation of language
  - By usage (fairly poor)
  - By performance (good)
  - By expressiveness (good)

Evaluating Troll as a DSL

- Advantages of being a standalone DSL
  - Completely domain-specific syntax \(3d6, \text{etc}\)
  - More usable for non-programmers, because users don’t need to have familiarity with a host language
- Disadvantages of being a standalone DSL
  - No language features (function calls, logic, tool support) for free
  - Doesn’t seem very extensible
    - How do you make a module?
    - How do you make a library?
- Type system
  - No explicit type annotations, but empty sets, singleton sets, and multielement sets are implicitly different (different function compatibility)
- Troll interpreter itself is the extent of a runtime system
  - ML runtime (garbage collection)
● Tool support
  ○ Troll could benefit from tools to help explore probability distributions of expressions and subexpressions, etc
● Non-standard vs secondary semantics?
● Troll-specific libraries
  ○ Troll could benefit from libraries for common games which implement the common rolls used in those games
● How Troll could be improved
  ○ Troll could benefit from probability calculations providing an estimate of the accuracy of imperfect calculations, or a user-specified acceptable margin of error to determine when to truncate infinite approximations
  ○ Troll could benefit from a static type system incorporating probabilistic and deterministic calculations, as well as singleton values.
● Troll seems to be clearly a DSL
  ○ Has its own syntax and interpreter
  ○ Designed with a specific set of use cases and users in mind
● List of games with unusual dice rules
  ○ Troll can express the rules clearly and succinctly. No other language can do so.

More detailed questions
● Ways of calculating the finite probability map for a dice roll
  ○ Enumeration in space: combine finite maps for all subexpressions
    ■ Less repeated computations
    ■ Much worse space complexity
  ○ Enumeration in time: Prolog-style backtracking to find possible outcomes one at a time and calculate probability for each
    ■ Redundant computation
    ■ Much better space complexity