Troll, A Language for Specifying Dice-Rolls

Torben Mogensen, DIKU

ACM SAC 2009
Why have a DSL for dice-rolls?

- Concise and unambiguous descriptions for communicating between people.
- Internet dice servers.
- Probability calculations for
  - Figuring your chances (player).
  - Deciding difficulty level (GM).
  - Design-space exploration (game designer).
The role-playing game “Dungeons & Dragons” from 1974 introduced use of non-cubical dice.
The role-playing game “Dungeons & Dragons” from 1974 introduced use of non-cubical dice.

... and notation such as 3d10+2. This notation has been used in many later games.
The role-playing game “Dungeons & Dragons” from 1974 introduced use of non-cubical dice

...and notation such as 3d10+2. This notation has been used in many later games.

Many games use dice-rolls that can’t be described by the notation from D&D.
The role-playing game “Dungeons & Dragons” from 1974 introduced use of non-cubical dice

...and notation such as 3d10+2. This notation has been used in many later games.

Many games use dice-rolls that can’t be described by the notation from D&D.

In 2002 I designed Roll as an attempt at a universal notation for dice-rolls and made programs for rolling and analysing rolls described in Roll.

Roll was used in the design of the latest version of the game “World of Darkness” from 2004.
The role-playing game “Dungeons & Dragons” from 1974 introduced use of non-cubical dice and notation such as $3d10+2$. This notation has been used in many later games.

Many games use dice-rolls that can’t be described by the notation from D&D.

In 2002 I designed Roll as an attempt at a universal notation for dice-rolls and made programs for rolling and analysing rolls described in Roll.

Roll was used in the design of the latest version of the game “World of Darkness” from 2004.

Some dice-rolls were not easy to describe in Roll, so in 2006 I made the successor Troll.
Elements of Troll

- A roll is a *collection* (multiset) of numbers:
  - Order is irrelevant
  - Number of occurrences is significant.
- A collection with one element can be used as a number. Some operations require this.
- Collections can be combined, filtered, counted, summed and in other ways manipulated to find a final result.
- Two different semantics:
  - Random rolling
  - Calculation of probability distribution
Basic Troll operations

- $dN$ rolls a single $N$-sided die.
- $M_dN$ rolls $M$ $N$-sided dice and makes a collection of the results.
- $\text{sum } C$ adds the elements in the collection $C$.
- $\text{counts } C$ counts the elements in the collection $C$.
- $+, -, *, /$ do arithmetic on numbers.
- $@$ finds the union of two collections.
- $M < C$ returns the elements of $C$ that are greater than $M$. Also for $=, >, <=, >=, =/=$.
- $\text{min}$ and $\text{max}$ find the smallest or largest element in a collection, respectively.
- $\text{least } N$ and $\text{largest } N$ find the least or largest $N$ elements of a collection.
**Simple Troll definitions**

- **sum 2d10 + 3**
  Adds two ten-sided dice and adds 3 to the result.

- **sum largest 3 4d6**
  adds the largest 3 of 4 six-sided dice.

- **count 7 < 6d10**
  counts how many out of six d10s are greater than 7.

- **max 3d20**
  finds the largest of three d20.
- $M \# e$ makes $M$ independent samples of expression $e$ and combines the results using @.
- $\text{if } C \text{ then } e_1 \text{ else } e_2$ If $C$ is non-empty, do $e_2$, otherwise do $e_3$.
- $x := e_1; e_2$ defines $x$ to be the value of $e_1$ inside $e_2$. $x$ is sampled once and this value used for every occurrence of $x$ inside $e_2$.
- $\text{repeat } x := e_1 \text{ until } e_2$ repeats rolling $e_1$ until the expression $e_2$ evaluates to non-empty, then returns last value of $e_1$.
- $\text{accumulate } x := e_1 \text{ until } e_2$ repeats rolling $e_1$ until the expression $e_2$ evaluates to non-empty, then returns the union of all values of $e_1$.
- $\text{foreach } x \text{ in } e_1 \text{ do } e_2$ calculates $e_1$, and for each number $n$ in the result evaluates $e_2$ with $x$ bound to $n$, then unions the results of $e_2$. 
b := 2d6; if (min b) = (max b) then b@b else b
Backgammon dice.

count 7< N#(accumulate x:=d10 while x=10)
Die roll for World of Darkness.

repeat x := 2d6 until (min x) < (max x)
Roll two d6 until you don’t have a double.

x := 7d10; max foreach i in 1..10 do sum i= x
Largest sum of identical dice.
The two semantics:

- **Random rolls** is implemented fairly straightforwardly using a PRNG.
- **Probability distribution** implemented by enumerating all possible rolls and counting results.
Enumerating all possible rolls can be done in several ways:

**In time:** Backtrack over all possible rolls, counting at top-level. Advantage: Low space use (only top-level distribution is stored).

**In space:** Find distributions for subexpressions and combine these to find distribution for complete expression. Advantage: Can combine identical subresults and exploit certain properties of functions.

It turns out that the latter far outweighs the former (details in paper).
Representation of probability distributions

Simple representation: Set of (value, probability) pairs:

\[ \{(2, 0.25), (3, 0.5), (4, 0.25)\} \]

Unnormalised representation to exploit algebraic properties of functions:

\[
D \equiv M! + D \cup D + D \mid_p D + 2 \times D
\]

- \( M! \) means “\( M \) with probability 1” where \( M \) is a multiset of numbers.
- \( d_1 \cup d_2 \) combines all outcomes of \( d_1 \) and \( d_2 \) by union.
- \( d_1 \mid_p d_2 \) chooses between the outcomes of \( d_1 \) and \( d_2 \) with probability \( p \) of choosing from \( d_1 \).
- \( 2 \times d \) is an abbreviation of \( d \cup d \).

Main idea: Avoid combinatorial explosion of unioning two distributions.
Linear functions

\[ f(M_1 \cup M_2) = f(M_1) \cup f(M_2) \]

Examples: \(<\), \(\leq\), \(\text{foreach}\)
Can be lifted to unnormalised distributions:

\[
\begin{align*}
    f(M!) &= f(M)! \\
    f(d_1 \cup d_2) &= f(d_1) \cup f(d_2) \\
    f(d_1 |_p d_2) &= f(d_1 |_p f(d_2) \\
    f(2 \times d) &= 2 \times f(d)
\end{align*}
\]
Homomorphic functions

\[ \exists \oplus : f(M_1 \cup M_2) = f(M_1) \oplus f(M_2) \]

Examples: sum, count, min, least \( N \), if, different

Can be lifted to unnormalised distributions:

\[
\begin{align*}
 f(M!) &= f(M)!
 f(d_1 \cup d_2) &= f(d_1) \hat{\oplus} f(d_2)
 f(d_1 \mid_p d_2) &= f(d_1) \mid_p f(d_2)
 f(2 \times d) &= \oplus^2 f(d)
\end{align*}
\]

\[
\begin{align*}
 M! \hat{\oplus} N! &= (M \oplus N)!
 (d_1 \mid_p d_2) \hat{\oplus} d_3 &= (d_1 \hat{\oplus} d_3) \mid_p (d_2 \hat{\oplus} d_3)
 d_1 \hat{\oplus} (d_2 \mid_p d_3) &= (d_1 \hat{\oplus} d_2) \mid_p (d_1 \hat{\oplus} d_3)
\end{align*}
\]

\[
\begin{align*}
 \oplus^2 M! &= (M \oplus M)!
 \oplus^2 (d_1 \mid_p d_2) &= (\oplus^2 d_1) \mid_p^2 ((\oplus^2 d_2) \mid_{\frac{(1-p)^2}{(1-p^2)}} (d_1 \hat{\oplus} d_2))
\end{align*}
\]
Exploit that repeat and accumulate have unchanged conditions in all iterations:

- Distribution of body calculated once, then rewritten into the form

\[ d_1 \mid_p d_2 \]

where the values in \( d_1 \) fulfil the condition and values in \( d_2 \) don’t.

- For repeat-until, the resulting distribution is \( d_1 \).
- For accumulate-until, the resulting distribution \( d' \) is given by the equation

\[ d' = d_1 \mid_p (d_2 \cup d') \]

Solution is infinite, but cut off after specified limit.
Experiences with **Troll**

- Non-programmers can write simple definitions.
- While optimisations help a lot, sometimes **Troll** needs to enumerate all combinations, which may be slow.
- New features added occasionally by request from users (latest: text and recursive function definitions).
- Download from [www.diku.dk/~torbenm/Troll](http://www.diku.dk/~torbenm/Troll) (Requires Moscow ML).