Card Stock: Abstract Card Game Design and Analysis

Connor Bell and Dr. Mark Goadrich
Department of Mathematics and Computer Science
Hendrix College, Conway AR

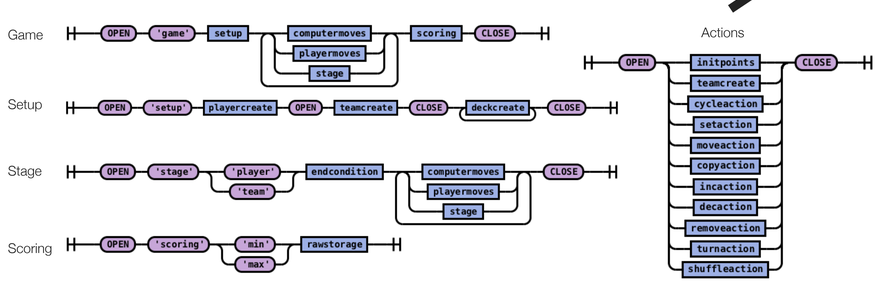
ABSTRACT

Card games have long been enjoyed by children and adults of all ages, with several companies producing cards for centuries. However, some card games are more popular than others. Our research goal is to computationally simulate various card games and quantify their quality automatically (Browne, 2012). We have thus far developed two components: a card game description language (Font et al., 2013). and an interpreter for this language. Using these components, we can generate sample run transcripts and analyze their game flow.

SPADES IN RECYCLE

```
(game
   (setup
      ;; Set up the players, 2 teams of 2 players, alternating
      (create players 4)
      (create teams (0, 2) (1, 3))

      ;; Create the deck source
      (create deck (game loc DISCARD) (deck (rank (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K))
                                            (color (red (suit (hearts, diamonds)))
                                                   (black (suit (clubs, spades))))))
   )

   ;; Stages of the game
   (stage player
      (end (>= ((any team) sto SCORE) 500))
      (comp (() (move (top (game loc DISCARD))
                      (top (game loc STOCK)) all)
                (remember (top (game loc STOCK where (all (== (cardatt suit each) spades))))
                          (top (game mem TRUMP)))
                (shuffle (game loc STOCK))
                (move (top (game loc STOCK))
                      (top ((all player) loc HAND)) 13)
                (set (game sto BROKEN) 0)
                (set ((all player) sto TRICKSWON) 0)
                (set ((all player) sto BID) 14)
            )
      )

      ;; bidding for number of tricks expected
      (stage player
         (end (<= ((all player) sto BID) 13))
         (choice
            (() (set ((current player) sto BID) 1))
            (() (set ((current player) sto BID) 2))
            (() (set ((current player) sto BID) 3))
            (() (set ((current player) sto BID) 4))
         )
      )

      ;; players play a round 13 times
      (stage player
         (end (== (size ((all player) loc HAND)) 0))

         ;; players play a hand once
         (stage player
            (end (> (size ((all player) loc TRICK)) 0))

            (choice


               ;; if following player and cannot follow suit
               ;;   play any card, and end your turn
               ((and (== (size (game loc LEAD)) 1)
                     (== (size ((current player) loc HAND where (all (== (cardatt suit each)
                                                                         (cardatt suit (top (game mem LEAD)))))) 0))
                (move (any ((current player) loc HAND))
                      (top ((current player) loc TRICK))))

               ;; if following player and can follow suit
               ;;   play any card that follows suit, and end your turn
               ((and (== (size (game loc LEAD)) 1)
                     (!= (size ((current player) loc HAND where (all (== (cardatt suit each)
                                                                         (cardatt suit (top (game mem LEAD)))))) 0))
                (move (any ((current player) loc HAND where (all (== (cardatt suit each)
                                                                     (cardatt suit (top (game mem LEAD)))))))
                      (top ((current player) loc TRICK))))

               ;; if first player and spades not broken and have non-spades cards
               ;;   play one of these, remember it in the lead spot, and end your turn
               ((and (== (size (game loc LEAD)) 0)
                     (== (game sto BROKEN) 0)
                     (> (size ((current player) loc HAND where (all (!= (cardatt suit each)
                                                                        (cardatt suit (top (game mem TRUMP)))))) 0))
                (move (any ((current player) loc HAND where (all (!= (cardatt suit each)
                                                                     (cardatt suit (top (game mem TRUMP)))))))
                      (top ((current player) loc TRICK)))
                (remember (top ((current player) loc TRICK))
                          (top (game mem LEAD))))
                                . . . . . .
```

SELECTED RECYCLE GRAMMAR RULES

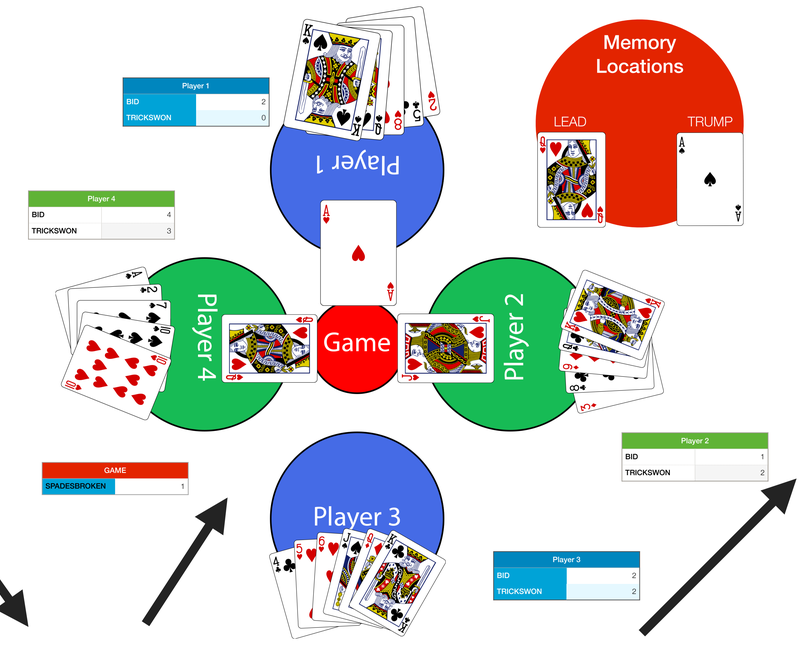


IMPLEMENTATION

First, we created a language, titled RECYCLE, in which turn-based games can be represented. We restrict ourselves to games which use only cards and numeric tokens and where all card locations are spatially independent.

Second, we implemented a library, written in C#, called Card Stock, which contains the functions and mechanisms necessary to run RECYCLE programs and executes the basic operations performed in these card games. The games written in RECYCLE are simulated with random players to collect statistics, such as player branching factor, average game length, game complexity, etc.

SPADES IN ACTION



CARD STOCK ENGINE

ANALYTICS DATABASE

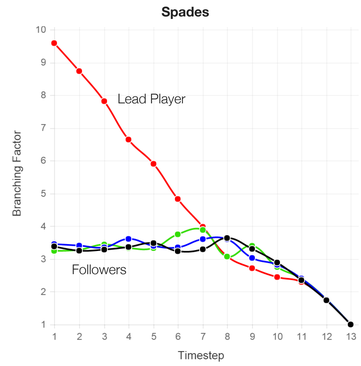
Actions

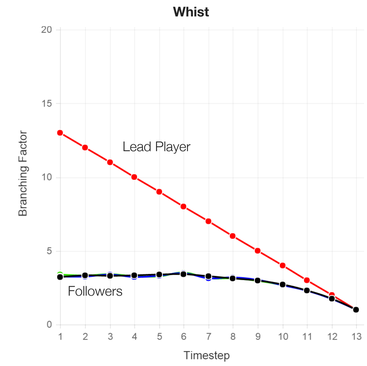
GAME TRANSCRIPT

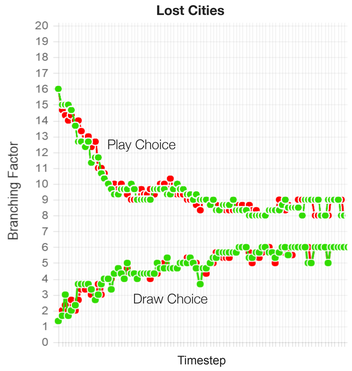
...
Moved 4 of Clubs from Player 3 HAND to Player 3 TRICK
Remember 4 of Clubs as Game LEAD
Moved 7 of Clubs from Player 4 HAND to Player 4 TRICK
Moved 6 of Clubs from Player 1 HAND to Player 1 TRICK
Moved Jack of Clubs from Player 2 HAND to Player 2 TRICK
Initialize Scoring PRECEDENCE
Forget Game LEAD
Cycle Next Player 2 (the owner of the max of all players's TRICK using PRECEDENCE)
Increment Player 2 TRICKSWON by 1
Move all players's TRICK to Game DISCARD
Moved Jack of Diamonds from Player 2 HAND to Player 2 TRICK
...

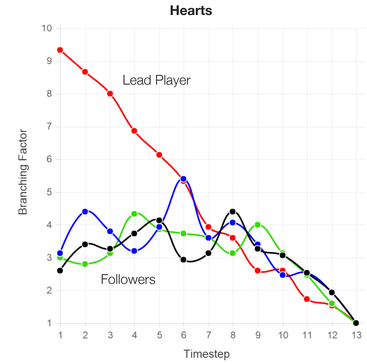
RESULTS

We have demonstrated that RECYCLE can capture the mechanics of several traditional card games including Spades, Hearts, Whist, Game of Pure Strategy, Crazy Eights, and Cribbage, plus a number of commercial games with custom decks of cards, such as Lost Cities, Pairs, and Sushi Go!. Using RECYCLE and Card Stock, we can run each of these games thousands of times, allowing us to understand the strategic and tactical implications of the rules in action through statistics and graphical analysis.









Of the four graphs above, Spades, Whist, and Hearts are trick-taking games. We can see that the rule of following suit constrains the branching factor for the followers. Spades and Hearts have a restriction on when trump can be led, thus limiting the branching factor for the leader, whereas in Whist, there is no such restriction. In Lost Cities, we can observe how play choices are limited as the game progresses while draw choices increase.

REFERENCES

Browne, C. (2012). Elegance in game design. *Computational Intelligence and AI in Games, IEEE Transactions on*, *4*(3), 229-240.

Font, J. M., Mahlmann, T., Manrique, D., & Togelius, J. (2013). *A card game description language* (pp. 254-263). Springer Berlin Heidelberg.

SOFTWARE ACKNOWLEDGEMENTS

Visual Studio Code
Chart JS
MS SQL
Visual Studio
Microsoft Azure
ANTLR 4
RRD For ANTLR 4
Spyresca's Vector Cards