A Poker Player

Chris Brown

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report

January 2006

Abstract

This Texas Hold 'Em client relies on enumerations and probabilities. As parameters, it uses many functions that vary with the number of players or the round of the game. It incorporates some random behavior. It cannot bluff, but it can be bluffed. It models opponents generically (it is unlikely he has this particular hand since he should have dropped those hole cards). It models opponents as individuals (this guy is conservative, wins more hands than you would expect on showdowns, but is a money loser). It seems to play a reasonable game but may not be better than other approaches with fewer calculations and fewer parameters. It is written in (un-indented) Perl and lives at `/u/brown/src/perl/poker/mod/CBClient.pl`

1 Background

To replace the project of writing computerized Diplomacy players in CSC400, our introductory problem seminar for first-year graduate students, George Ferguson came up with the idea of a computer poker competition. Texas Hold 'Em (*cf., e.g.* [7]) seemed like a good choice, partly because of the limited number of choices (basically, just betting) available to the players, and partly because of its popularity in high-stakes games. We used rulebooks from Scarne and Cardoza ([9; 5]).

Ferguson wrote the Internet Poker Protocol (IPP) and an interactive graphic poker server, plus some automated poker-playing test clients [6]. I thought the idea would be good for a semester-long sophomore-level term team project as well [4]. There are many references and actual games on the web.

My player is crudely inspired by the work of Schaeffer, Billings, *et al.* [10; 3; 2; 8; 1]. At least I read their papers, feel like I forgot them, and then pretty much feel like I reinvented the first half of their method "independently". Whether one credits my subconscious or not, their approach seems "obviously" the right basic one, which is to use enumeration to figure out the probability of holding the best hand, and then to proceed from there.

Most of the play is dictated by probabilities, which are soon distorted by heuristic adjustments. There is some of "random" decisionmaking however, spread around throughout the strategy, for example whether to wait for the flop with lousy hole cards, whether to try to check and raise. My client can be bluffed, but does not bluff others. The check-and-raise ploy is its only way of sandbagging.

2 Basic Hand Strength

First, ranking the deal: Following Cardoza, I categorize the hole cards into four classes. My rankings are rather more liberal than his: in particular I put all pairs into category 3 (don't drop), where he would fold on anything less than a pair of 8's as I recall.

1. (highest): suited AK, AQ; unsuited AA, KK, QQ,AK.
2. suited AJ, AT, KQ; unsuited AQ, JJ, TT.
3. suited AT,KJ,QJ,KT,QT; unsuited: all pairs not ranked higher, AJ,KQ,KJ.
4. (lowest): everything else.

A basic capability is to compute the (IPP, say) description of a 5-card hand and to assign it a rank, so that it can be compared against any other to see whether it is higher, lower, or the same in a poker showdown.

Built on that is the still-basic capability to find the best hand in a 6- or 7-card deal (the turn or river, with four and five community cards respectively), using one or both hole cards. To compute all hands for a 7-card deal takes 20 evaluations of the ranking utility.

Perl let me keep previous (5- and 6-card) results in a hash table so no hands are ranked twice as more cards arrive. Since this is a basic utility it should be as fast as possible.

Built on this last basic capability are several applications of enumeration. The idea is to enumerate what other pairs of cards are hidden (either as hole cards or as the remaining two cards to be dealt after the flop) and to evaluate hands resulting from those two cards being dealt. That is, we evaluate either the opponent's hands (he holds the hidden two hole cards) or my hands (I get to use the two future community cards). Generally this enumeration involves taking all combinations of two cards from the population of cards not yet exposed on the table and computing the resulting (best) hands that an opponent or I could realize. Then some subset of that set of hands must be enumerated, and the ratio of the size of that subset to size of the set of all possible hands is an interesting percentage. Generally there are something like 1000 (1081, 1035, 990) hands in the possible future set. Billing's papers go into this in some detail, but the idea is simple.

- After the flop: Compute my hand rank. For all $N = 1081$ (47 choose 2) remaining pairs in the deck, each is equally likely to be an opponent's hole card. Compute B, the number of those hands that my hand outranks. The ratio $E = B/N$ is my *Hand Strength*, namely the probability that my hand will beat a random opponent's hand after the flop. Exponentiating E to the number of opponents gives my expectation of a win at that point.

- Also after the flop: There are two cards coming in the community cards. What are my chances of improvement? This is a similar enumeration except worse since for each of those 1081 pairs I need to compute the best hand for all 7 cards, multiplying the work by 20 over computing the hand effectiveness. Note however that Billings goes further...they seem also to enumerate how well the opponent could do for each of these 5-card future community layouts. I don't do that, I resort to one of many shabby heuristics.

- After the turn: Six cards are now visible. Opponent can have 46 choose 2, or 1035 pairs of hole cards. Enumerate them, see how many of the resulting hands I outrank, get an updated hand strength.

- After the river: Seven cards are now visible. Opponent can have 45 choose 2, or 990 pairs of hole cards. Enumerate them, see how many of the resulting hands I outrank, get an updated hand strength.

These calculations take essential all the computation time in my client. Perl's hash tables simplify things enormously.

One decision I made was to try to map all my strategic decisions onto the probability interval, so I have a sort of commmon representation for all the intuitions and hacks and tweaks. What is bad about that

For me, probabilities, or maybe expected utilities, seem absolutely the right decision-making tools here. Others, like the VALENTINE player of CS400, make very good use of rule-based decision-making.

2.1 Advanced Hand Strength (or Elementary Opponent Modeling)

The client's calculations rely heavily on mappings, tables, or functions if you like that give parameter values as a function of either the hand round (deal, flop, turn, river), or of the number of players in the game (up to 20, say). So far all these functions are monotonic, and some are constant. They are implemented as Perl arrays. They all are probably close to linear functions of i or $1/i$. Although many are now clipped or saturated, they probably all could be expressed as functions of a few parameters.

The probabilities computed in the last section are pretty much worthless, since they assume no one ever folds. Billings and colleagues deal with this problem; it is the beginning of modeling how opponents might behave if they use judgement.

I adapt an idea from Billings and have a table indexed by deal quality, which has four values from 1 to 4, as discussed above.

`HandWeights = (1,1, 1, .8, .6); .`

`HandWeight[i]` is my (made up) estimate of the probability that a particular hand in an enumeration will actually be there in an opponent's hand given that his hole cards had quality `i`. I "know" the hole-card condition because that is the pairs over which I am iterating.

Thus when enumerating hands, I do not simply add 1 for each hand to get its representation in the enumeration, I add its `HandWeight.` This de-weights the influence of good hands that can arise from bad hole cards, assuming some likelihood that the player will have dropped. This weighting is used in computations for the Turn and River, but for some reason I can't now recall I don't use them for computing flop hand strength probabilities. That seems wrong, and could change.

This apriori modeling comes to grief in, for instance, Ferguson's default client who never folds. Neither do some of the undergraduate programs.

3 Deal Policy

I have a parameter indexed by the number of players, called `StayProb[i]`. Before each deal I get a random number between 0 and 1 and if it is less than `StayProb[HandPlayers]` I stay in until the Flop no matter how bad my hole cards are, pretending they are quality 3.

Otherwise when possible I open and raise for quality 1 hands, open and call on 2 and 3 and fold on 4.

4 Flop Policy

Compute hand strength, or `HandStr`, as outlined in Section 2. (Currently without handweights. Seems silly, but that's how it is now.)

Exponentiate the hand strength to the number of players active in this hand. The result is the chance that I'm better than all of them.

Compute the possibility that I will improve my hand by at least one category (e.g. from HIGHCARD to PAIR or TWOPAIR or...).

Combine the hand strength with the improvement possibility.

`EffHandStr = HandStr + (1- HandStr) * ChanceImprove * Optimism;`.

Here the `Optimism` parameter is my optimism that the improved hand will be a winning one. Currently I ignore all information about the distribution of hand improvements and also what improvements are likely for opponents.

Attenuate the hand strength on the basis of advanced opponent modeling (Section 8.)

Now I have a hacked-up but still probality-like number for use in deciding whether to fold, check, open, call, raise, tapout, etc.

5 Expected Gain

Here is a tirade about "pot odds". What the hell are people trying to get at with this concept? What I've read makes no sense to me at all. Scarne goes on about "you forget what you've put into the pot, every pot is a new calculation", and Cardoza is the same. They seem to say pot odds are the current pot divided by the current bet. So this seems to me to be saying: if there's a 2000 pot and you are owing 20 to call, then your "pot odds" are 100:1, and you'd be comfy betting with a 0.01 hand strength. What's wrong with this picture? What's wrong is that if there are 5 players, you're guaranteed to have put in something like 2000/5 at this point or you wouldn't be here. So you're not risking just 20, you're risking something like 420.

So I don't get it. Why not just calculate your expected gain on the bet, which if you owe `Owing` dollars and have bet `HandBet` already into the pot, is simply what you expect to win that is not yours minus what you expect to lose that is yours: `ExpectedGain = (EffHandStr*(Pot- HandBet)) - (1 - EffHandStr)* (HandBet+ Owing);`

What's wrong with this logic? It does not work either (see next section), but it seems at least mathematically correct to me.

6 Back to the Flop

Anyway, I started out with expected gain calculated as above as my only criterion for deciding whether to bet, raise, fold. If expected gain was say twice what I'd bet so far, I'd raise. What happens with this idea is that with 10 people in the game, you can have a pretty big expected gain with a pretty small chance of winning. Perhaps because I am chicken, perhaps because I'm not thinking right, I backed off this simple, actuarial approach because I was constantly losing showdowns. I guess this means I was too optimistic on hand strength calculations. But it also means I take more chances the more opponents are out there, which seems crazy. Something like the doubling-up strategy, maybe it goes wrong because of finite stakes and betting limits.

One way to cope, which I in fact do, is to have a table indexed by the number of players, as in

`GainThresh = (0,0,1.6, 2.5, 3.5, 4, 4, 4, 4, 4, 4, 4, 4,...)`, which provides multipliers of the amount I have bet so far this hand to give me my expected gain threshold. Now the decision is based not on a constant threshold expected gain, but on the relation between how much I have bet so far and the expected gain. Here I have to win 1.6 times my bet to raise in a two-person game and 4 times my bet in games with more than 5 people.

`ExpectedGain > GainThresh[HandPlayers]* HandBet`.

Further, the decision is now supplemented with two tables of hand strength thresholds, indexed by round number (Flop = 2, River = 4): `CallThresh = (0,.50,.50, .50, .50); RaiseThresh = (0,.70,.75, .75, .80);`

(These values, like many of the examples, are defaults and can be changed with an input file). The final decision is that if I have a strong possibility of winning OR a big expected gain then I'll raise. I can make the needed expected gain large enough not to dominate the decsion entirely. In the example functions I need a better hand in later rounds to justify a raise.

So now the condition defining a hand I'll raise (or open) on is:

```
((ExpectGain > GainThresh[HandPlayers]*HandBet) ||
(EffHandStr > RaiseThresh[HandRound])).
```

If `((ExpectGain <= 0.0) || (EffHandStr < CallThresh[HandRound]))` I fold.

Otherwise, I call (or open).

7 Turn and River Policy

Compute hand strength, or `EffHandStr`, as outlined in Section 2, using handweights.

Attenuate the hand strength on the basis of advanced opponent modeling (Section 8.)

Decide action by method of Section 6.

Except that on Turn and River I can check. Normally I do this if I've been checked to and otherwise would just fold. But with some probability, in a table indexed by number of active players, I will check with a high-quality, raisable hand in the hope of suckering him in. This is not a winning idea for most of the algorithms I'm up against, but it might be effective against Capt. Napalm, who notices these things. This table gives the probability that I'll try that with the index number of players. And this example now looks high to me.

`CheckAndRaiseProb = (0,0,0,0,0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.3,...);`

8 Advanced Opponent Modeling

I keep some records of hands, opponent actions and outcomes. I record the number of hands, the players' buyin amounts, and keep track of their current cash. I record the number of

times they fold during deal, flop, and turn (rounds one through three), and the number of hands they have won. I can then categorize players as follows.

Moneywinner if his cash now exceeds his buyin.

Handwinner if (hands won) / [(hands played) - (hands folded in rounds one through three)] exceeds the value in a table indexed by the number of active players in the game at this point:

```
HandWinnerThresh = (0,0, .6, .5, .4, .3, .25, .25, .25,.....);
```

The thing I'm trying to measure is how many hands you win at a showdown, and (in this table) if that's greater than 25% in a game of 7 or more, you're a hand-winner.

Conservative if you fold a lot in the first three rounds: i.e. if (hands folded in rounds one through three)/(hands played) > FoldThresh. Currently the threshold is set at .33.

`RaisedThisRound` if you did: this round you made a raise I have to respond to.

I start running these statistics after some small number of hands has been played (currently 4, but settable as variable `SignificantHands`.)

Once I have players classified, I use the classifications to attenuate my hand strength if I am facing conservative players (C) who are money (CM) or game (CG) winners, or both (CGM).

For now I only attenuate my confidence when I sense winning conservatives...I could raise my confidence for flakes, (notC) but I'm not doing that right now. The following tables say how much I attenuate my effective hand strength in round[i] for CG, CM and CGM opponents. I use these attenuations on rounds 2 (flop) 3 (turn) 4 (river).

```
CG = (0,0,.99, .95, .9);
CM = (0,0,.99, .98, .95);
CGM = (0,0,.97, .95, .9);
RTR = (0,0, .99, .98, .95);
```

These weights are multiplied, one for each winning conservative player facing me. Finally, aThe RaisedThisRound (RTR) attenuation is applied for each conservative winner who raised this round. The final attenuation brings down my hand strength.

There could be other uses: adjusting my style of play in a one-on-one game, for instance, depending on the opponent's characteristics.

9 Compiled Parameters Summary

These parameters are currently compiled in: others can be read in before the game starts.

```
FoldThresh = .33;
SignificantHands = 4;
HandWinnerThresh = (0,0, .6, .5, .4, .3, .25, .25, ...);
```

10 Initialization Parameters Summary

A command line argument gives the file name of an initialization file that overrides default parameters. Here are two samples, from which the easily-changeable parameters can be identified. Array elements are just strung out after the identifier for their Array, which is the same as in the code (but without the Perl "at" and "dollar" sign).

Here is an input file that sets the settable parameters aggressively. As in FORTRAN, C is for comment. `[....]` is NOT in the file, it is shorthand for *etc., etc.* The series should have at least as many terms as the maximum number of players supported (currently 20).

```
C Parameters for an aggressive player
C First, he doesn't care about any other player's characteristics
C and so there'll be no attenuation based on that
CG 0 0 1 1 1
CM  0 0 1 1 1
CGM  0 0 1 1 1
RTR  0 0 1 1 1

C next, his threshold for staying in and raising are lower than
C a "normal" player
CallThresh 0 .4 .4 .4 .4
RaiseThresh 0 .70 .70 .70 .70

C It takes less expected gain for him to raise
GainThresh  0 0 1.4  1.8  2.0  2.5  3.0  3.5  4.0 4  4 4 [....]

C He may have dropped by now if hole cards weak
HandWeights 1 1 .8 .7 .6

C Always ready to check and raise
CheckAndRaiseProb  0 0 .1 .1 .1 .1 [....]

C Never drop with bad hole cards
StayProb 1 1 1 1 1 1 1 1 1 1 1 [....]

C Optimistic about improving his hand after flop
Optimism .8


C Parameters for an overly-conservative player
C First, he is very sensitive to other conservative players

CG 0 0 .98 .92 .9
CM  0 0 .98 .95 .92
```

```
CGM  0 0 .95 .92 .9
RTR  0 0 .96 .93 .9

C next, his threshold for staying in and raising are higher than
C a "normal" player
CallThresh 0 .55 .55 .55 .55
RaiseThresh 0 .7 .75 .77 .8

C next, it takes more expected gain  to raise
GainThresh  0 0 1.6  2.7   3.7  4  4.5  5.0 5.5  6 6 [....]


C No expectation opponent will have dropped with weak hole cards
HandWeights 1 1 1  1   1

C Never  check and raise-- is this aggressive or conservative? arguable!
CheckAndRaiseProb  0 0 0 [....]

C Likely to drop with bad hole cards.   These are prob of staying
StayProb 1 1 .5 .45 .4 .35 .3 .25 .2 .2 [....]

C Not optimistic about effects of improving his hand after flop
Optimism .4
```

11 Results

Here is a transcript that shows the default player with an initially promising hand: the introspection illustrates some of the capabilities and concepts described above. The accounting error is from a pot-split.

```
Rpt: Accting err: players+pot = 5001 ; Total should be 5000
Rpt: I'm Dealt: AHQD
Rpt: stayprob 0.4 randnum: 0.387237548828125
Rpt: I'm going to stick this deal out for a while
>>>>
CALL 10
<<<<

Rpt: Flop: AS JS 9D
Rpt: My Best Hand after Flop: 15MKJH ONEPAIR A Q J 9
Rpt: Handstrength (Chances of being ahead) at flop with 2 players:0.961147086031452
Rpt: Handstrength at flop with 4 players: 0.88791125449143
Rpt: My chance of improving some : 0.68348623853211
Rpt: My chance of improving by a hand rank: 0.492660550458716
```

```
Rpt: Bogus Effective Hand Strength: 0.921044276328917
Rpt: EffHandStr before attenuate: 0.921044276328917
Rpt: 4 cons. other players -- conserv. bel atten: 0.92217609
Rpt: EffHandStr after attenuate: 0.84936500946188
Rpt: Expected Gain for bet after Flop: 51.4428508515692
>>>>
RAISE 20
<<<<

Rpt:
TurnComm: AS JS 9D 8S
Rpt: This hand rank:  15;  Before Turn, rank was :15
Rpt: My Best Hand after Turn: 15MKJH ONEPAIR A Q J 9
Rpt:  Weighted HandsAbove mine: 95.2
Rpt:  Total weighted hands: 1035
Rpt: Turn: EHS for 2: 0.908019323671498EHS for  4 players: 0.748661108031653
Rpt: 4 cons. other players -- conserv. bel atten: 0.81450625
Rpt: EffHandStr after attenuate: 0.609789151623707
Rpt: Turn: Expected Gain for bet after Turn: 45.6152100179116
>>>>
OPEN 20
<<<<

Rpt: RivComm: AS JS 9D 8S 7S
Rpt: My Best Hand after River: 15MKJH ONEPAIR A Q J 9
Rpt:  Weighted HandsAbove mine: 357.2
Rpt:  Total weighted hands: 990
Rpt: River: EHS for 2: 0.639191919191919EHS for  3 players: 0.408566309560249
Rpt: 4 cons. other players -- conserv. bel atten: 0.6561
Rpt: EffHandStr after attenuate: 0.268060355702479
Rpt: River: Expected Gain for bet after River: -14.6864199669422
>>>>
CHECK
<<<<

Rpt: River: Expected Gain for bet after River: -3.964005738843
>>>>
FOLD
<<<<
```

So far this rather complex client performs at about the level of CaptainNapalm or VALENTINE from the CSC400 class. Luck at getting good cards still is a major issue, of course. I cannot guarantee that there are not conceptual bugs or evaluation and enumeration mistakes, though I have made several tests of the component algorithms and the "thinking

aloud" numbers as shown in this section seem fairly reasonable.

Below is a snapshot from the client playing its two most serious rivals, VALENTINE and CaptainNapalm from the CSC400 class. I have spent essentially no time analyzing or tuning parameters, but have made up functions as in Section 10. The three CB clients feature an aggressive and a very conservative one as well as the default, but all are in need of adjustment.

The positions of the players after 41 hands is as follows. "Won" is games won that were not dropped. "F123" is number of early folds. "Cons, HWin, Mwin" are ratings for conservative, hand winner, money winner. CB has folded about as much as the conservative CB player, a sign of bad cards. Captain Napalm has surged back from near bankruptcy with a flush and a straight flush in two consecutive hands. This example is not to be taken as statistically significant. In my experience it takes hundreds of hands before reliable patterns emerge.

```
 Hands: 41; Pot  25; MyCash: 725
Name  Pos  Buyin  Cash   In? Won? F123? RtR Cons? HWin? MWin?
VALENT 0   1000   230    1   6     33    0      1   1   -1
Captai 1   1000   923    1   13    26    0      1   1   -1
CBagg  2   1000   1765   1   16    24    0      1   1   1
CB     3   1000   725    1   3     35    0      1   1   -1
CBcons 4   1000   1333   1   5     36    0      1   1   1
```

12 Discussion and Future Work

VALENTINE uses a rule-based approach, while Captain Napalm starts from essentially the same calculations used herre. (This client does calculate the chances of improving the flop hand.) Napalm's approach is elegant in that it assigns odds to other players. This may be more intellectually satisfying than my approach, I have to think about it. It seems to get to the same sort of conclusion without having all those ad-hoc functions to invent, which is a major disadvantage of my current client. Deeper comparisons are obviously possible but I've no time for that now.

Ongoing questions:

How to map qualities (like "conservative", "aggressive") onto the threshold functions in a satisfying way?

What are pot odds really? What are they trying to tell us? Would a straight expected payoff version of my program do better than my heuristic one, say?

Why does the code say there are 1090 possible future hands in the flop calculation? seems same as opponents to me, or 1081.

Why not use handweights in flop eff. calcs? Don't people drop before the flop?

Switching the style of play depending on who is at the table. That is, especially if there is only one other player at the time, or only one type of player.

Even conservative players should bid up others when they have good hand.

13 Acknowledgements

Thanks to George Ferguson for inventing this idea, inspiration throughout, dazzling graphics, solving poker problems even professional poker players don't understand, thoroughly professional software support ("get lost!"), and continuing interest (I hope). Also thanks to the CSC400 and CSC172 students: we do it all for you.

References

[1] D. Billings. *Computerized Hold 'Em Player*. WWW, http://www.cs.ualberta.ca/ jonathan/Papers/Papers/ai98.poker.html.

[2] D. Billings. Computer poker. Technical report, Dept. of Computing Science, U. Alberta, Aug 1995.

[3] D. Billings, D. Papp, J. Schaeffer, and D. Szafron. Opponent modeling in poker. In *Proceedings: Conference of AAAI*, 1998.

[4] C. Brown. *CSC 172 Poker Home Page*. WWW, http://www.cs.rochester.edu/u/brown/assts/lab/poker.html.

[5] Avery Cardoza. *How to Play Winning Poker*. ISBN 0-940685-04-3, unk, unk.

[6] G. Ferguson. *U. Rochester Poker Home Page*. WWW, http://www.cs.rochester.edu/u/ferguson/poker/home.html.

[7] D. Kimberg. *Dan's Poker Dictionary*. WWW, http://cortex.med.upenn.edu/ kimberg/pokerdict.html.

[8] Alberta Poker Project. *Poker Project Home Page*. WWW, http://www.cs.ualberta.ca/ games/poker/index.html.

[9] John Scarne. *Guide to Modern Poker*. ISBN 0-617-53076-3, unk, unk.

[10] J. Schaeffer. *Experimental Computer Science in Game Playing*. Unknown.

A Server Bug

Players who have busted occasionally get back into the game. More often than not actually. Here we have a case where the server doesn't seem to notice that there is only one player left, and subsequently allows the last player to fold and the non-player to win. It is a little confusing since there's 2 Rich VALENTINES and one poor POORVAL who as we join the action is going broke...

```
<VALENTINE NO
>VALENTINE OK NO
>POORVAL FROM VALENTINE FOLD
>VALENTINE FROM VALENTINE FOLD
>ALL WINNER VALENTINE 185 FULLHOUSE 7 J
>ALL BUSTED POORVAL
>ALL BUTTON POORVAL
Starting hand: VALENTINE=955 POORVAL=0 VALENTINE=1205
>ALL ANTE 5
>VALENTINE DEAL JH 2D
>VALENTINE DEAL 9C AS
>VALENTINE ACTION? BLIND 5
<VALENTINE BLIND 5
>VALENTINE OK BLIND 5
>VALENTINE FROM VALENTINE BLIND 5
>POORVAL FROM VALENTINE BLIND 5
>VALENTINE ACTION? STRADDLE 10
<VALENTINE STRADDLE 10
>VALENTINE OK STRADDLE 10
>POORVAL FROM VALENTINE STRADDLE 10
>VALENTINE FROM VALENTINE STRADDLE 10
>VALENTINE ACTION? OWING 5
<VALENTINE CALL 5
>VALENTINE OK CALL 5
>VALENTINE FROM VALENTINE CALL 5
>POORVAL FROM VALENTINE CALL 5
>ALL FLOP 3S AH 6C
```

...the hand goes on and comes to a weird end below. Looks like both the VALENTINE programs fold, but Poorval is declared the winner. ...

```
>ALL RIVER 7D
>VALENTINE ACTION? OWING 0
<VALENTINE OPEN 20
>VALENTINE OK OPEN 20
>VALENTINE FROM VALENTINE OPEN 20
>POORVAL FROM VALENTINE OPEN 20
```

```
>VALENTINE ACTION? OWING 20
<VALENTINE FOLD
>VALENTINE OK FOLD
>POORVAL FROM VALENTINE FOLD
>VALENTINE FROM VALENTINE FOLD
>ALL WINNER POORVAL 110
>ALL BUTTON VALENTINE
```

...so now he's back, but it doesn't look like he's getting any cards. So it's a bit inconsistent, seemingly... I'm noticing this since I'm trying to keep track of money. Possible to toss these losers out for permanent?

```
Starting hand: VALENTINE=910 POORVAL=110 VALENTINE=1140
>ALL ANTE 5
>VALENTINE DEAL 5D JC
>VALENTINE DEAL 7H 8D
>VALENTINE ACTION? BLIND 5
<VALENTINE BLIND 5
>VALENTINE OK BLIND 5
>VALENTINE FROM VALENTINE BLIND 5
>POORVAL FROM VALENTINE BLIND 5
>VALENTINE ACTION? STRADDLE 10
<VALENTINE STRADDLE 10
>VALENTINE OK STRADDLE 10
>POORVAL FROM VALENTINE STRADDLE 10
>VALENTINE FROM VALENTINE STRADDLE 10
>VALENTINE ACTION? OWING 5
<VALENTINE FOLD
>VALENTINE OK FOLD
>VALENTINE FROM VALENTINE FOLD
>POORVAL FROM VALENTINE FOLD
>ALL WINNER VALENTINE 25
>ALL BUTTON VALENTINE
Starting hand: VALENTINE=920 POORVAL=110 VALENTINE=1130
```

I thought this might have something to do with the OnePlayerLeft Exception, but I couldn't find anything wrong. As of now the cause remains unclear and the problem persists.