Contents

Chapter 1

util

1.1 1-2-3 Stops

```
1m - (1X); 2m
2X*: inv+
2N : inv
3m : s/o
```

```
1M - (X); 2X-1*: (5)6+X, 6-10; or 4+X, GF
3X : s/o against weak variant
```

1.2 1m - 1M; 2N

```
1m - 1H(-1); 2N -
3C*: ask
   - 3D : 4+D [m = C], or catchall [m = D]
   - 3H : 3H
   - 3S : 4S  # could have 3H if m = C
   - 3N*: 5+C [m = C], or 4S3H [m = D]
3D : 5+D [m = C], or 3+D [m = D], MST+
3H : 6+H, MST+
3S*: (4)5+C
3N : s/o
4m : 5H5m
4H : s/o
```

```
1m - 1S(-1); 2N -
3C*: ask
   - 3D : 4+D [m = C], or catchall [m = D]
   - 3H : 4H  # could have 3S if m = C
   - 3S : 3S
   - 3N*: 5+C [m = C], or 3S4H [m = D]
3D : 5+D [m = C], or 3+D [m = D], MST+
3H*: (4)5+C
3S : 6+S, MST+
3N : s/o
4X : 5H5X, MST+
4S : s/o
```

1.3 2NT

```
# B = bid, all (B) here are NF raise, or bids that (may be) weak
 1Y -(2X)- 2N : nat
 1X -(2Y)- 2N : nat
 1N -(2X)- 2N : tr. Leb
(2C)- X* -(P) - 2N : nat
(2X)- X* -(P) - 2N : Leb
(1C)- X* -(2C)- 2N : nat
(1X)- X* -(2X)- 2N : Leb
(1X)- 1Y -(B) - 2N : nat
(1Y)- 2X -(B) - 2N : nat
 1X -(2M)- X* -(P) - 2N : good-bad
 1X -(1Y)- X/B-(2Y)- 2N : good-bad
 1X -(1Y)- X/B-(2Z)- 2N : good-bad
 1Y -(2C)- X/B-(P) - 2N : nat
 1Y -(2X)- X/B-(P) - 2N : good-bad
 1Y -(2X)- P  -(P) - 2N : t/o, usually 64+mms
 1Y -(B) - P  -(2X)- 2N : t/o, usually 64+mms
(2M)- P  -(P) - X* -(P) - 2N : Leb
(1X)- P  -(2X)- X* -(P) - 2N : Leb
(1X)- X* -(B) - X* -(P) - 2N : normal Leb (good-bad)
(1X)- 1Y -(B) - X* -(P) - 2N : normal Leb (good-bad)
(1Y)- 2X -(B) - X* -(P) - 2N : normal Leb (good-bad)
(1C)- 2X -(2C)- X* -(P) - 2N : nat
(1X)- 1N -(2X)- X* -(P) - 2N : min nat

# general rules for other situations:
# tr. Leb applies only after 1N - (2X) and (1S)- 2H -(2S)
# ... X* -(P) - 2N = usually normal Leb (good-bad)
# ... X* -(B) - 2N = normal Leb (good-bad) if X = neg or t/o
# otherwise, free bid 2N is nat if inv is possible; otherwise t/o
# if there is not possible for inv+ (ex: balancing X by 1N opener), then 2N
    = nat
```

1.4 Forcing Pass

1.4.1 XX = Q

1.4.2 vs 5X pre

```
(3X) - X - (5X) - P = F
(3X) - P - (5X) - P = F
```

1.5 suit GT

```
1S - 2S -  # or anytime showing 4-4 fit in 2S
2N*: ask
   - 3X*: feature in X  # at least KJ/QJT
   - 3S : min, w/o feature
   - 3N*: max, w/o feature
   - 4X*: spl X
3X*: HSGT/ST in X  # request void/x/xx/Qx/A(+)/K(+)
3S : 6+S inv
3N : s/o
4C+: spl
```

```
1H - 2H -  # or anytime showing 5-3 fit in 2M
2S*: ask
   - 2N*: feature in S  # at least KJ/QJT
   - 3m*: feature in m
   - 3H : min, w/o feature
   - 3S*: max, w/o feature
   - 3N*: spl S
   - 4X*: spl X
2N*: HSGT/ST in S  # request void/x/xx/Qx/A(+)/K(+)
3m*: HSGT/ST in m
3H : 6+H inv
3S+: spl
3N : s/o
```

```
1m - 1S[-1] - 2S -  # or anytime showing 4-4 fit in 2S
2N*: ask
   - 3m*: good m
   - 3X*: spl X
   - 3S : min, w/o short
   - 3N*: max, w/o short
2N*: HSGT/ST in S  # request void/x/xx/Qx/A(+)/K(+)
3m*: HSGT/ST in m
3H : 6+H inv
3S+: spl
3N : s/o
```

```
1m - 1H[-1] - 2H -  # or anytime showing 4-4 fit in 2H
2S*: ask
   - 2N*: spl S
   - 3m*: good m
   - 3om: spl om
   - 3H : min, w/o short
   - 3S*: max, w/o short
2N*: HSGT/ST in S  # request void/x/xx/Qx/A(+)/K(+)
```

```
3m*: HSGT/ST in m
3H : 6+H inv
3S+: spl
3N : s/o
```

1.6 transfer Lebensohl

```
1N -(2M)  # or (1M)- 1N -(2M)
2S : NF
2N*→3C*- P* : s/o
        - 3D*: s/o
        - 3oM: s/o
        - 3M*: 5+C, GF
        - 3N : half stop
   - 3X : 18+, nat
3X*→3X+1: 5+[X+1], inv+; if X+1 = M, then Stayman
         - 3M : max, but ask stop
3S*: ask stop
3N : s/o
```

1.7 normal Lebensohl

```
normal Leb over (2M) -
(2H/) 2S : NF
2N*→3C*- 3X : s/o
        - 3M*: 4+oM, GF, have stopper
        - (2H/) 3S : 5+S, inv
3X [X < M]: 5+X, inv
3M*: 4+oM, GF, no stopper
3X [X > M]: 5+X, GF
```

1.8 maximum X

```
fit in 2M - (opp. comp to 3X) -
# if X = M - 1
X*  : inv+  # allow pen with low probability
# otherwise
3M-1: inv+
```

```
(1m) - 2H [V/NV] - (3D) - X*  : 2+H, inv
(1m) - 2H [V/NV] - (3C) - 3D* : 2+H, inv
```

1.9 Rubens

```
(1X)- 1Y -(P/X)-
XX*: honor in Y (lead-directing)
1N : nat, 9-11
2N : nat, 12-14
1Z : 8+, 5+Z, F1
# if Z < X
2Z : 8-12, 5+Z, NF
# if X < Z < Y (transfer from opp's suit)
2Z-1 → 2Z*: 10+, nat F1; or s/o in Z
     - 3Z : s/o
2Y-1: cuebid. 10+, 3+Y; or GF w/o stopper
2Y : constr raise
# Z > Y
2Z : 13-15, 6+Z, inv
# jumps
3Z : 13-15, 6+Z, inv
3X*: mixed raise. 9-11, 4+Y
3Y : pre
```

```
(1Y)- 2X -(P/X)-
XX*: honor in X (lead-directing)
2N : nat inv
# if Z < Y
2Z : 10+, nat F1
# if Y < Z (transfer from opp's suit)
2Z-1 → 2Z*: 10+, nat F1; or s/o in Z
# if Z < Y (transfer from opp's suit in 3rd-level)
3Z-1 → 3Z*: GF; or s/o in Z
3Y-1: cuebid. 10+, 3+Y
3Y : constr raise
```

1.10 Slam bidding

1.10.1 cuebid

```
cue = 1/2nd ctrl
# if opener shows a suit (unless 1C - 1X; 1N/2N), then
opener's cue on that suit = 2 of AKQ, usually source of tricks
resp's cue on that suit = never shortness, can be Q
cuebid denies lower control
```

1.10.2 FF

```
[fit in 3M] -
4M : min
3M+1*: FF, mild slam interest
4X*: cuebid, strong slam interest
```

1.10.3 kickback RKC

```
[fit in S] - 4N*: ask number of keycards  # 4 Ace + Trump K
5C*: 0/3 keycards
   - 5H*: escape to 5S if 0-keycards
5D*: 1/4 keycards
   - 5S*: P if 1-keycard
5H*: 2/5 keycards w/ Trump Q
5S*: 2/5 keycards w/o Trump Q
5N*: 0/2/4 keycards, some void
   - 6C*: ask
        - 6X*: void in X
6X*: 1/3 keycards, void in X
```

```
[fit in X] - [4X+1]*: ask number of keycards
# similar responses, 5N replaces void in [X+1]
[fit in H] - (4S); 4N*: RKC
```

1.10.4 ERKC

```
[fit in X] -
# if opener already shows non-void Y, then it is just cue-bid
5Y*: ask number of keycards, excluding Y
   - +1*: 0/0+Q
   - +2*: 1
```

```
   - +3*: 1+Q
   - +4*: 2
   - +5*: 2+Q
   - +6*: 3
   - +7*: 3+Q
   ...
```

1.10.5 ORKC

```
4X*: ORKC
+1*: min
+2*: same as resp. to RKC
```

1.10.6 2-suied RKC

```
1M - 2X; 3X - 3M; ... [4M+1]: 2-suited RKC
# also 1N - 3DH- 4H+
# Queen of M and X act as 0.5 keycards
+1*: 0/3/6 keycards  # may +0.5
   - +2*: ask if there's extra 0.5
        - 5M*: no
+2*: 1/4/7 keycards  # may +0.5
   - +3*: ask if there's extra 0.5
        - 5M*: no  # +4 = 5M
+3*: 2/5 keycards
+4*: 2.5/5.5 keycards
```

1.11 UwU

TBD (low-low, high-high)

1.12 XYZW

1.12.1 2wPCB

(https://www.ptt.cc/man/BridgeClub/D6D1/D49B/D130/M.924860463.A.html)

```
1X - 1Y; 1N
2C*→2D*: transfer accepted
        - P  : s/o
        - 2M : s/o, choose a partial [M <= Y]; inv, 5+Y, 4+M [M > Y]
        - 2N*: inv
        - 3Z : inv, 6+Z [Z = Y] or 4+Z [Z = X] or 5+Z and 5+Y [otherwise]
        - 3N*: 5332, CoG  # different from BTUBWS
   - 2Y*: max, 3Y
2D*: GF, ask
   - 2M : 3M [M = Y] or 6M [M = X] or 4M [otherwise]
   - 2N : nat
   - 3X : good 5+X
2M : inv, 5+M [M = Y] or 4+M [otherwise], NF
2N*→3C*: transfer accepted
        - P  : s/o
        - 3D : 4-5Y, CoG, no slam interest. spl D.
            - 3H*: ask if 5Y
        - 3H : 4-5Y, CoG, no slam interest. spl H.  # spl C if Y = H
            - 3S*: ask if 5Y
        - 3S : 5Y, CoG, no slam interest. spl S.  # spl C if Y = S
        - 3N : 4Y, CoG, no slam interest. spl S.  # spl C if Y = S
        # a bit diff from BTUBWS. similar to 1N - 2S; any - 3M*
3Z : ST, 4+Z [Z = X] or semi-solid 6+Z [Z = Y] or 5+Z [otherwise]
3N : s/o
4C+: 7+Y, spl
   - 4M : waste
4Y : s/o
```

1.12.2 PLOB

```
1C - 1D*; 1H*-
1S*: any (10)11-14
   - 1N : 11-13, 2H bal
        - 2C : s/o
   - 2C : 11-14, 2-H, (5)6+C
   # bids below applies to both 1N and 2C
        - 2D*: F, not prefer to declare NT
        - 2H : s/o
        - 2S : s/o
        - 2N+: nat inv
```

```
   - 2D*: GF
   - 2H : 3H, F1
   - 2S*: GF
1N : nat NF
2X : s/o
   - 2S*: F
   - 2N+: nat inv
   - 3S*: 6+C, 5+S, F
2N*: 15+, catchall
3C*: fit in C, ST
3D*: 5+H, 5+D, ST
3H*: 6+H, ST
3S*: 4+S, ST
3N*: 18-19, 4H
```

```
1D - 1H; 1S -
1N : nat NF
2C*: any (10)11-14
   - 2D : 11-14, 2-H
        - P  : s/o
        - 2H : s/o
        - 2S*: F, not prefer to declare NT
             - 2N : min
             - 3N : max
        - 2N+: nat inv
   - 2H : 3H, F1
   - 2S*: general GF
   - 2N+: nat GF
2X : s/o
2N*: 15+, catchall
3C*: 5+H, 5+C, ST
3D*: fit in D, ST
3H*: 6+H, ST
3S*: 4+S, ST
3N*: 18-19, 4H
```

1.12.3 after 2N = 18-19 bal

```
1m - 1M(-1); 2N-
3C*: major-oreiented ask, promises 5+M
3D*: fit in opener's suit, ST
3M : 6+M, ST
3oM: nat, 4+oM [M = H]; or 5+oM [M = S]
3N : s/o
4om: nat 5+M, 5+om
4m : RKC(om)  # usually 6+om
```