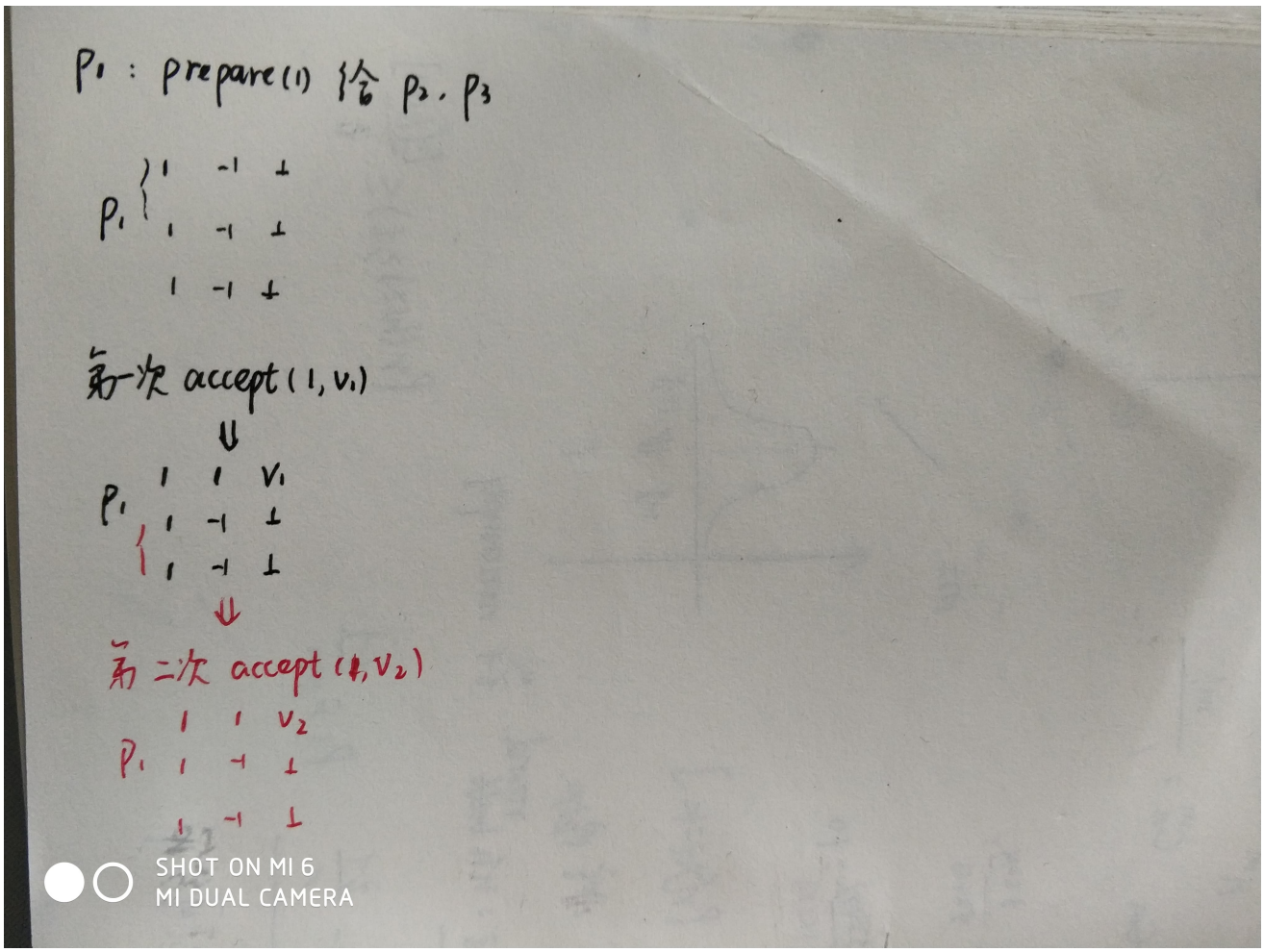
Accept动作限制条件

- **只有允许进行accept的条件（quorun）**

  Accept动作会直接更新状态，只有只一个条件会导致很多问题出现：

1. 参与者进行了两次accept(b, v)，v值不同。
2. 参与者进行accept之前make promise(编号更大)。

针对第一个情况给出反例：



为了限制这一情况的发生，我添加了maxVBal#b的条件。

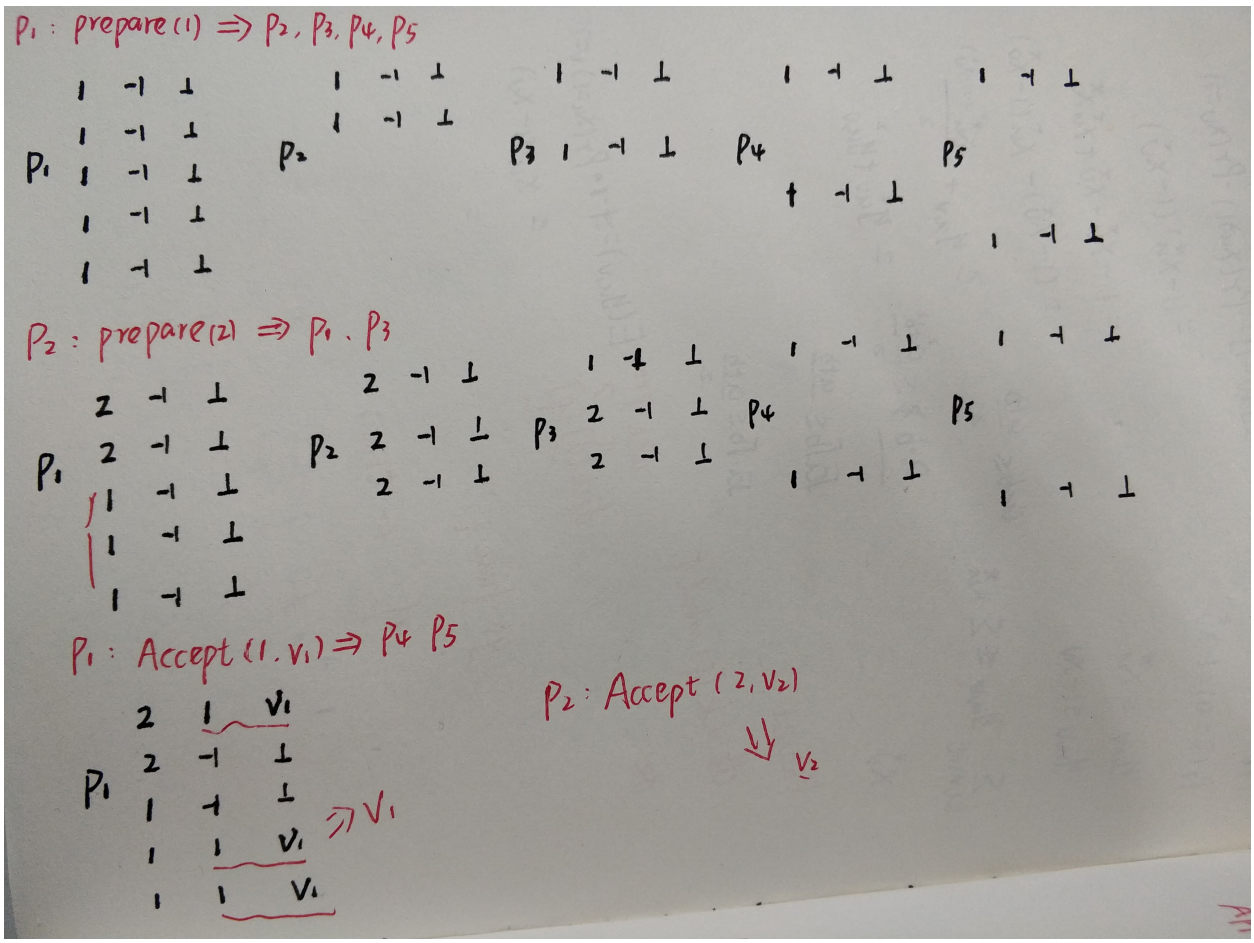
- **maxVBal # b**

  添加了这个条件，限制了上述同一轮投票两次Accept动作的发生，但是如果当程序执行到maxVBal > b的这种状态下，Accept动作能否进行(按理来说不可以)，放松一点，即如果程序执行到maxBal > b的这种状态下，Accept动作能否进行(Paxos不允许，因为Accept动作会更新接受过的提议值，此时maxBal > b，不会接受该提议)。

  即分析当maxBal > b的时候能不能进行Accept(b, v)动作：

  - 不可以：和Paxos算法相同，但是相对Paxos来说，就少了P2a(b, v)的发生，这是因为Accept动作相当于Paxos的P2a(b, v)+一次P2b(b, v)。
  - 可以：似乎符合伪代码，会不会产生问题。

原TPaxos算法没有给出这种情况的应对策略，伪代码里面似乎也没有对这种情况进行限制，不过我找到了一个反例：）



综上，需要限制maxVbal的同时限制maxBal:

maxBal $\leq$ b $\wedge$ maxVBal < b

Refinement角度

根据TPaxos $\Rightarrow$ EagerVoting的refinement mapping：

maxBal $\leftarrow$ states[p]p].maxBal

并且TPaxos执行的Accept(p, b, v)动作对应的应该是EagerVoting中的VoteFor(a, b, v)动作：

$$
\begin{aligned}
&VoteFor(a,\ b,\ v) \triangleq \\
&\quad \wedge \quad maxBal[a] \leq b \quad \text{keep promise} \\
&\quad \wedge \quad \forall\, vt \in votes[a] : vt[1] \neq b \\
&\quad \wedge \quad \forall\, c \in Acceptor \setminus \{a\} : \\
&\qquad\quad \forall\, vt \in votes[c] : (vt[1] = b) \Rightarrow (vt[2] = v) \\
&\quad \wedge \quad \exists\, Q \in Quorum : ShowsSafeAt(Q,\ b,\ v) \quad \text{safe to vote} \\
&\quad \wedge \quad votes' = [votes \ \text{EXCEPT} \ ![a] = votes[a] \cup \{\langle b,\ v\rangle\}] \quad \text{vote}
\end{aligned}
$$

Accept(p, b, v)应该满足VoteFor的动作，先分析第一条，即转换成states[p]p].maxBal $\leq$ b，当p能进行b轮的accept阶段意味着该轮的prepare阶段已经进行过了，即states[p]p].maxBal $\geq$ b，所以states[p]p].maxBal = b。

对于第二条而言，我们需要在Accept中限制b轮的Accept动作不能发生第二次，可以添加条件states[p]p].maxVBal # b 即 states[p]p].maxVBal < b（根据上面maxBal分析得出）。

添加了这两个条件后，严格限制了Accept动作的执行时间，当且仅当b对应的参与者通过了prepare请求并且它**没有对更高的编号make promise**。Accept阶段相当于Paxos中的P2a+一个P2b，相对于Paxos的p2a阶段由于p2b加了限制。