TRC404 智能合约竞争条件分析

目前 TRC404 的智能合约主要有两种操作，一种是 transfer FT，Transfer FT 操作的入口是 TRC404 wallet 合约，一种是 transfer NFT，Transfer NFT 操作的入口是 NFT Item 合约入口。因为两种操作分别在两个不同的入口，因此可能会产生因竞争条件冲突而产生异常情况。本文将会描述解决竞争条件的方案，并详细分析这两种操作的所有节点是否存在竞争条件问题，确保不会因为竞争条件而导致不期望的行为或资产损失或 double spend 问题。

竞争条件冲突的解决方案

为了更好的解决 TRC404 合约的竞争条件问题，对 TRC404 合约做出如下的改进和限制：

1.对于 transfer FT

发送方：发送者的 TRC404 wallet 合约的 jetton_balance 将会直接扣减转账的金额，对于需要 burn 的 NFT，合约首先会找到序号最小且状态不是 pending_delete 的 NFT index.然后 TRC404wallet 合约并不会直接将 NFT index 从 owned_nft_dict 中删除，而是把需要 burn 的 NFT index 的状态设置成 pending_delete，等 NFT 真正 burn 了后，再发消息通知发送者的 TRC404 wallet 合约，把这个 NFT index 从 owned_nft_dict 中删掉。

接收方：接收者的 TRC404 wallet 合约的 jetton_balance 将不会直接加上转账的余额，而是先加上转账的金额再减去需要 mintNFT 的数量，然后每 mint 完一个 NFT，再发消息通知接收者的 TRC404 wallet 合约的 jetton_balance 加 1.

2.对于 transfer NFT

为了可以兼容 A transfer NFT to B 和 B transfer NFT to C 这两种交易可以执行成功，以兼容 getgems.io 的 put on sale 操作。TRC404 wallet 合约增加两个属性 L：pending_reduce_jetton_balannce 和 pending_transfer_nft_queue.

当执行上述两种操作时:

如果执行 B transfer NFT to C 时，NFT index 还没添加到 B 的 TRC404 wallet 的 owned_nft_dict，则把这个 NFT index 添加到 pending_transfer_nft_queue，等 A transfer NFT to B 执行完时，再从 pending_transfer_nft_queue 删掉这个 NFT index。

如果执行 B transfer NFT to C 时，B 的余额为 0，则 jetton_balance 设置为 0，pending_reduce_jetton_balannce 加 1。等 A transfer NFT to B 执行完时，

再使 jetton_balance =jetton_balance +1 - pending_reduce_jetton_balannce, 并使 pending_reduce_jetton_balannce=0。

3.Gas fee 问题

为了避免因 owned_nft_dict 存储太多元素，而导致查找元素 gas fee 过高而可能退出的问题，合约强制一个钱包最多只能拥有 5 个 NFT，即 TRC404 wallet 合约的 owned_nft_dict 最多只会有 5 个元素，如果有用户 transfer 1 个 NFT 到一个已经有 5 个 NFT 的钱包，则接收者只会收到 1 个 TRC404 token，这个 NFT 会被合约 burn，确保接收者钱包最多只会存储 5 个 NFT 的 index。

注意：当本文描述 jetton_balance +1 或-1 的时候，实际在源代码实现的时候，会写成 jetton_balance + 1 * min_unit() 和 jetton_balance - 1 * min_unit()，其中 min_unit()等于 1 * 10^9。

1.合约核心数据和状态

对于 TRC404 wallet 合约，核心的数据和状态包括：

jetton_balance :current TRC404 token balance

owned_nft_number: current NFT number (user-owned)

owned_nft_limit: The max NFT number that one wallet can hold

owned_nft_dict: NFT dictionary (user-owned),key:NFT item index,value: null means normal status,0 means the status is pending delete

pending_reduce_jetton_balance: 只有在 addOneFtAndNFT 和 reduceOneFtAndNFT 涉及操作这个数据

pending_transfer_nft_queue: 只有在 addOneFtAndNFT 和 reduceOneFtAndNFT 涉及操作这个数据

对于 NFT Item 合约，核心的数据和状态包括：

owner: owner of this NFT

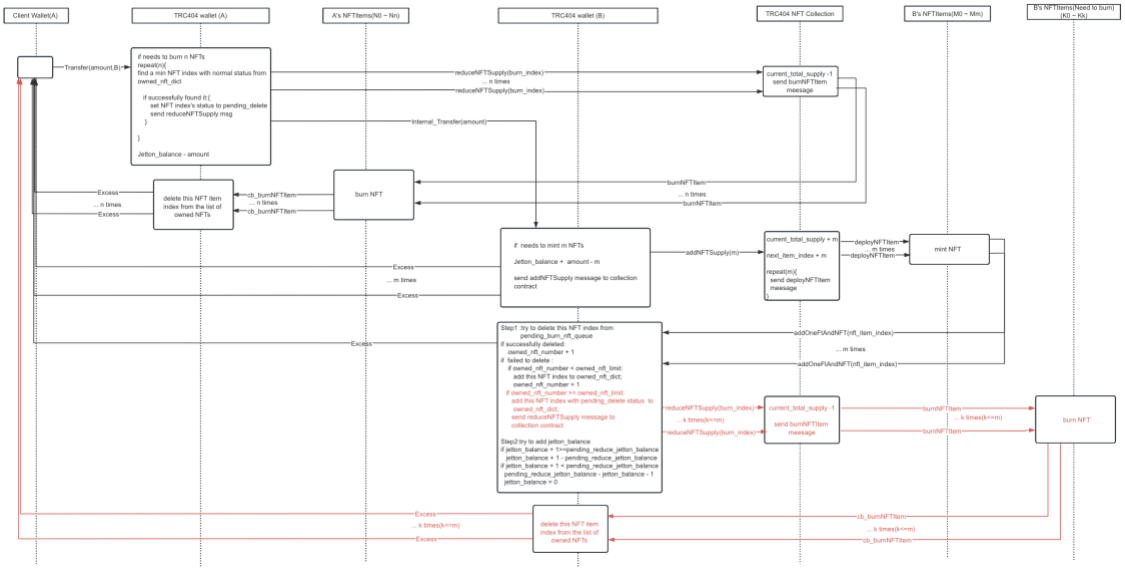
contract active status: after mint NFT,NFT status will be Active, after burn FT, NFT status will be non-active

对于 NFT Collection 合约，核心的数据和状态包括：

current_total_supply: current total NFT supply, burn NFT will decrease this number, mint new NFT will increase this number

next_item_index: The next NFT item index, it will increase when minting a new NFT

2. Transfer FT



上图是 Transfer FT 的完整时序图。

抽象简化后的步骤为：

1. Sender TRC404wallet 合约：

```
 jetton_balance -= amount;

repeat( need_to_burn_nft_number){

   find a min and valid NFT index from owned_nft_dict;

   set this NFT index's status to pending_delete;

   send a reduceNFTSupply message to collection contract;

}
```

（涉及数据修改：jetton_balance，owned_nft_number，owned_nft_dict）

2.1 TRC404 NFT collection 合约：

```
    current_total_supply -= 1;
```

```
send burnNFT message to NFT Item;
```

（涉及数据修改：current_total_supply）

2.2 NFT Item 合约：

```
burn NFT;

send cb_burnNFT message to sender TRC404 wallet;
```

（涉及数据修改：contract active status）

2.3 Sender TRC404wallet 合约：

```
delete this NFT item index from owned_nft_dict;
```

（涉及数据修改：owned_nft_dict）

3.1 Receiver TRC404 wallet 合约：

```
jetton_balance = jetton_balance    + amount - need_to_mint_nft_number;

send addNFTSupply to collection;
```

（涉及数据修改：jetton_balance）

3.2 TRC404 NFT collection 合约：

```
current_total_supply += amount;

repeat(need_to_mint_nft_number) {

    next_item_index += 1;

    send deployNFTItem message to new NFT Item contract;

}
```

（涉及数据修改：current_total_supply，next_item_index）

3.3 NFT Item 合约：

```
    mint NFT

    send addOneFtAndNft message to receiver TRC 404 wallet
```

（涉及数据修改：owner ，contract active status）

3.4 Receiver TRC404 wallet 合约：

```
Step1:try to delete this NFT index from Pending_transfer_nft_queue;

 if (successfully deleted){

      owned_nft_number += 1;

 }

 if (failed to delete){

      if (owned_nft_number < owned_nft_limit){

            add this NFT index to owned_nft_dict;

            owned_nft_number += 1;

      }

      if (owned_nft_number >= owned_nft_limit){

            add this NFT index with pending_delete status to owned_nft_dict;

            send reduceNFTSupply message to collection contract;

       }

 }



Step2: try to add jetton_balance;

if (jetton_balance + 1 >= pending_reduce_jetton_balance){
```

```
    jetton_balance = jetton_balance + 1 - pending_reduce_jetton_balance;

}

if (jetton_balance + 1 < pending_reduce_jetton_balance){

      pending_reduce_jetton_balance -= (jetton_balance + 1);

      jetton_balance = 0;

}
```

（涉及数据修改：jetton_balance，owned_nft_number，owned_nft_dict，pending_reduce_jetton_balance，Pending_transfer_nft_queue）

3.4.1 TRC404 NFT collection 合约：

```
current_total_supply -= 1;

send burnNFT message to NFT Item;
```

（涉及数据修改：current_total_supply）

3.4.2 NFT Item 合约：

```
    burn NFT;

    send cb_burnNFT message to sender TRC404 wallet;
```

（涉及数据修改：contract active status）

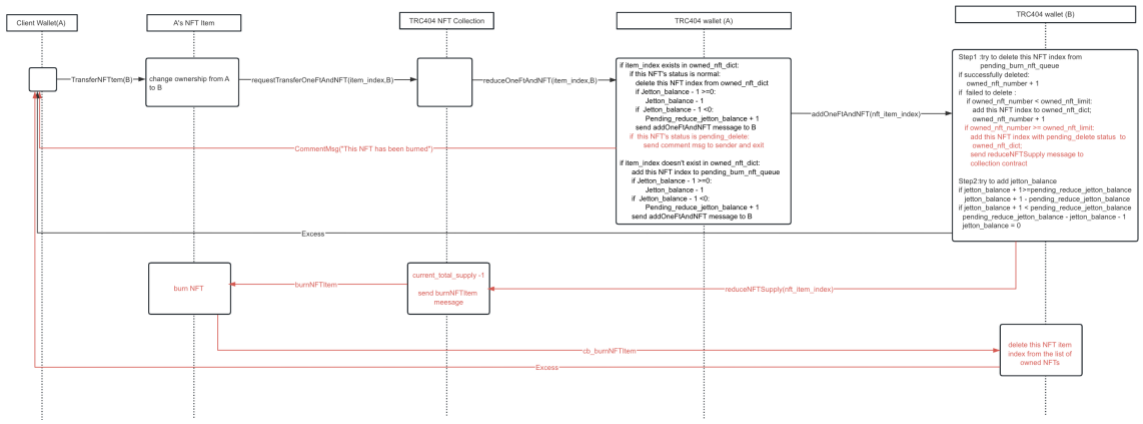
3.4.3 Receiver TRC404wallet 合约：

```
  delete this NFT item index from owned_nft_dict;
```

（涉及数据修改：owned_nft_dict）

注意：只有在执行 3.4 时，owned_nft_number >= owned_nft_limit 时，才会触发执行 3.4.1，3.4.2 和 3.4.3

3. Transfer NFT



上图是 Transfer NFT 的完整时序图。

抽象简化后的步骤为：

1. NFT Item 合约：

```
change ownership from sender to receiver;

send requestTransferOneFtAndNFT message to collection;
```

（涉及数据修改：owner ）

2. TRC404 NFT collection 合约：

```
send reduceOneFTAndNFT message to sender TRC 404 wallet;
```

（涉及数据修改：无）

3. Sender TRC404 wallet 合约：

```
if (item_index exists in owned_nft_dict){

    if (this NFT's status is normal){

        delete this NFT index from owned_nft_dict;

        if (Jetton_balance - 1 >=0){
```

```
                Jetton_balance -= 1;

            }

            if (Jetton_balance - 1 < 0){

                Pending_reduce_jetton_balance += 1;

                send addOneFtAndNFT message to B's TRC404 wallet;

            }

        }

        if (this NFT's status is pending_delete) {

            send comment msg to sender and exit;

        }



 }

if (item_index doesn't exist in owned_nft_dict){

            add this NFT index to Pending_transfer_nft_queue;

            if (Jetton_balance - 1 >= 0){

                Jetton_balance -= 1;

            }

            if (Jetton_balance - 1 < 0){

                Pending_reduce_jetton_balance += 1;

                send addOneFtAndNFT message to B;
```

```
        }

}
```

（涉及数据修改：jetton_balance，owned_nft_number，owned_nft_dict，pending_reduce_jetton_balance，Pending_transfer_nft_queue）

4. Receiver TRC404 wallet 合约：

```
Step1:try to delete this NFT index from Pending_transfer_nft_queue;

    if (successfully deleted){

          owned_nft_number += 1;

    }

    if (failed to delete){

        if (owned_nft_number < owned_nft_limit){

            add this NFT index to owned_nft_dict;

            owned_nft_number += 1;

        }

        if (owned_nft_number >= owned_nft_limit){

            add this NFT index with pending_delete status to owned_nft_dict;

            send reduceNFTSupply message to collection contract;

        }

    }
```

```
Step2: try to add jetton_balance;
    if (jetton_balance + 1 >= pending_reduce_jetton_balance){
        jetton_balance = jetton_balance + 1 - pending_reduce_jetton_balance;
    }

    if (jetton_balance + 1 < pending_reduce_jetton_balance){
        pending_reduce_jetton_balance -= (jetton_balance + 1);
        jetton_balance = 0;
    }
```

（涉及数据修改：jetton_balance，owned_nft_number，owned_nft_dict，pending_reduce_jetton_balance，Pending_transfer_nft_queue）

4.1 TRC404 NFT collection 合约：

```
current_total_supply -= 1;

send burnNFT message to NFT Item;
```

（涉及数据修改：current_total_supply）

4.2 NFT Item 合约：

```
burn NFT;

send cb_burnNFT message to sender TRC404 wallet;
```

（涉及数据修改：contract active status）

4.3 Receiver TRC404wallet 合约：

```
 delete this NFT item index from owned_nft_dict;
```

（涉及数据修改：owned_nft_dict）

注意：只有在执行 4 时，且 owned_nft_number >= owned_nft_limit，才会触发执行 4.1，4.2 和 4.3

3. 竞争条件分析

为了简化，对于 Transfer FT 的步骤，使用 FT1，FT2 这样的方式描述，对于对于 Transfer NFT 的步骤，使用 NFT 1，NFT 2 这样的方法描述。下面将详细描述 Transfer FT 和 Transfer NFT 的每个步骤是否有潜在竞争冲突问题。

从分类上看，可以分 3 大类冲突：即 transfer FT 和 transfer FT 冲突，transfer FT 和 transferNFT 冲突，transferNFT 和 transferNFT 冲突。另外，对于相同种类的冲突，除了考虑并发执行外，比如 A transfer 1 FT/NFT to B 和 A transfer 1 FT/NFT to B 这种情况，还需要考虑传递 transfer 的情况是否有问题：A transfer 1 FT/NFT to B，B transfer 1 FT/NFT to C。

3.1 transfer FT 的竞争分析

考虑一个用户并发执行 Transfer FT 操作是否存在潜在的冲突问题。假设用户同时执行两条 Transfer FT 交易，分别是 FT1 和 FT2. 下面将分析交易的每个步骤是否存在冲突，所谓冲突，表示会否因为并发执行而导致交易失败，资产减少或资产重复使用等异常情况。

3.1.1 A transfer FT to B 和 A transfer FT to B 竞争分析

FT1:A transfer FT to B 和 FT2: A transfer FT to B 交易可能存在的竞争节点分析如下表所示：

	FT2(A transfer FT to B)	Step 1(Sender TRC404wallet)	Step 2.1(collection)	Step 2.2(NFT Item)	Step2.3(Sender TRC404wallet)	Step3.1(Receiver TRC404 wallet)	Step3.2(collection)	Step3.3(NFT Item)	Step3.4(Receiver TRC404 wallet)	Step3.4.1 (collectiton)	Step3.4.2(NFT Item)	Step3.4.3(Receiver TRC404 wallet)
FT1(A transfer FT to B)												
Step 1(Sender TRC404wallet)	jetton_balance - owned_nft_number + owned_nft_dict:set NFT index to pending_delete			owned_nft_dict								
Step 2.1(collection)		current_total_supply - 1					current_total_supply			current_total_supply		
Step 2.2(NFT Item)			contract active status:non-active					contract active status				
Step 2.3(Sender TRC404wallet)	owned_nft_dict			owned_nft_dict: delete NFT index								
Step 3.1(Receiver TRC404 wallet)					jetton_balance +				jetton_balance			
Step 3.2(collection)		current_total_supply					current_total_supply+ next_item_index +			current_total_supply		
Step 3.3(NFT Item)			contract active status					owner :receiver contract active status:active			contract active status	
Step 3.4(Receiver TRC404 wallet)					jetton_balance				jetton_balance + owned_nft_number + owned_nft_dict: add NFT index pending_reduce_jetton_balance: + pending_burn_nft_queue:add NFT index			owned_nft_dict
Step 3.4.1(collectiton)		current_total_supply					current_total_supply			current_total_supply - 1		
Step 3.4.2(NFT Item)								contract active status			contract active status: non-active	
Step 3.4.3(Receiver TRC404 wallet)									owned_nft_dict			owned_nft_dict: delete NFT index

注意：涉及更新 collection 合约的 current_total_supply 和 next_item_index 的步骤,包括：

FT1 Step2.1 和 FT2 Step2.1，

FT1 Step2.1 和 FT2 Step3.2，

FT1 Step2.1 和 FT2 Step3.4.1，

FT1 Step3.2 和 FT2 Step3.2，

FT1 Step3.2 和 FT2 Step3.4.1，

FT1 Step3.4.1 和 FT2 Step3.4.1，

这些步骤无论执行加操作还是减操作，都是原子操作，不会涉及竞争条件冲突，因此不再分析。下面重点分析 TRC404 Wallet 合约和 NFT Item 合约可能发生的冲突。

3.1.1.1 FT1 Step1 和 FT2 Step1 （TRC404 Wallet）

有 3 个涉及竞争条件的数据，jetton_balance，owned_nft_number，owned_nft_dict。

对于 jetton_balance，在 Step1 中，合约直接尝试减去 transfer 请求包含的 amount，如果结果小于 0，则直接退出，不再执行后续操作，这是原子操作，不存在竞争冲突。

对于 owned_nft_number，在 Step1 中，合约会减去需要 burn 的 nft 数量，这是原子操作，不存在竞争冲突。

对于 owned_nft_dict，在 Step1 中，合约需要实现的逻辑是，找出需要 burn NFT 的 index，并且设置 index 的状态为 pending_delete。为了避免重复设置 index（潜在的竞争冲突），选择需要 burn NFT 的 index 的条件是，index 在 owned_nft_limit 中最小，而且 index 状态为正常，即不是 pending_delete。

3.1.1.2 FT1 Step1 和 FT2 Step2.3 （TRC404 Wallet）

有 1 个涉及竞争条件的数据， owned_nft_dict。

对于 owned_nft_dict，Step1 对 owned_nft_dict 执行的操作是寻找符合条件的 NFT index，并设置 NFT index 的状态为 pending_delete,Step2.3 对 owned_nft_dict 执行的操作是，删除指定的 NFT index。

如果发生竞争条件冲突，则表示 FT1 Step1 操作的 index 和 FT2 Step2.3 操作的 index 为一样，但因为如 3.1.1.1 所描述，Step1 每次会查找最小的且状态为正常的 index，然后 burn index 对应的 NFT。因此 FT1 Step1 和 FT2 Step2.3 操作 owned_nft_dict 的 index，不可能一样，亦即不会发生冲突。

3.1.1.3 FT1 Step2.2 和 FT2 Step2.2 （NFT Item）

有 1 个涉及竞争条件的数据， NFT item 的 contract active status

对于 contract active status，Step2.2 执行的操作是 burn NFT，从而 contract active status 会变成 non-active.

如果发生竞争条件冲突，则表示 FT1 Step2.2 操作的 NFT Item 和 FT2 Step2.2 操作的 NFT Item 为一样，但因为如 3.1.1.1 所描述，Step1 每次会查找最小的且状态为正常的 index，然后 burn index 对应的 NFT。因此 FT1 Step2.2 和 FT2 Step2.2 操作的 NFT Item，不可能一样，亦即不会发生冲突。

3.1.1.4 FT1 Step2.2 和 FT2 Step3.3 （NFT Item）

有 1 个涉及竞争条件的数据， NFT item 的 contract active status

对于 contract active status，Step2.2 执行的操作是 burn NFT，从而 contract active status 会变成 non-active. Step3.3 执行的操作是 mint NFT，同时 contract active status 会变成 active.

如果发生竞争条件冲突，则表示 FT1 Step2.2 操作的 NFT Item 和 FT2 Step3.3 操作的 NFT Item 为一样。因为在 Step3.2 中，需要 mint 的 NFT index 来源于 collection 合约的 next_item_index,并且每 mint 一个新的 NFT，next_item_index +1，因此 FT1 Step2.2 和 FT2 Step3.3 操作的 NFT Item，不可能一样，亦即不会发生冲突。

3.1.1.5 FT1 Step2.3 和 FT2 Step2.3 (TRC404 Wallet)

有 1 个涉及竞争条件的数据， owned_nft_dict.

对于 owned_nft_dict，Step2.3 执行的操作是从 owned_nft_dict 删除指定的 NFT item index。

如果发生竞争条件冲突，则表示 FT1 Step2.3 操作的 NFT Item 和 FT2 Step2.3 操作的 NFT Item index 为一样，但因为如 3.1.1.1 所描述，Step1 每次会查找最小的且状态为正常的 index，然后 burn index 对应的 NFT。因此 FT1 Step2.3 和 FT2 Step2.3 操作的 NFT Item index，不可能一样，亦即不会发生冲突。

3.1.1.6 FT1 Step3.1 和 FT2 Step3.1 (TRC404 Wallet)

有 1 个涉及竞争条件的数据， jetton_balance.

对于 jetton_balance，在 Step3.1 中，合约直接尝试加上 internal_transfer 请求包含的 amount,再减去需要 mint 的 NFT 的数量，从合约内部看，这是原子操作，不存在竞争冲突。

3.1.1.7 FT1 Step3.1 和 FT2 Step3.4 (TRC404 Wallet)

有 1 个涉及竞争条件的数据， jetton_balance.

对于 jetton_balance，在 Step3.1 中，合约直接尝试加上 internal_transfer 请求包含的 amount,再减去需要 mint 的 NFT 的数量。在 Step3.4 中，jetton_balance 可能会执行 jetton_balance +1 - pending_reduce_jetton_balance 或设置 jetton_balance=0.
从合约内部看，这都是原子操作，不存在竞争冲突。

3.1.1.8 FT1 Step3.3 和 FT2 Step3.3 (NFT Item)

有 1 个涉及竞争条件的数据， NFT item 的 contract active status

对于 contract active status， Step3.3 执行的操作是 mint NFT，同时 contract active status 会变成 active.

如果发生竞争条件冲突，则表示 FT1 Step3.3 操作的 NFT Item 和 FT2 Step3.3 操作的 NFT Item 为一样。但在 Step3.2 中，需要 mint 的 NFT index 来源于 collection 合约的 next_item_index,并且每 mint 一个新的 NFT，

next_item_index +1，因此 FT1 Step3.3 和 FT2 Step3.3 操作的 NFT Item，不可能一样，亦即不会发生冲突。

3.1.1.9 FT1 Step3.4 和 FT2 Step3.4（TRC404 Wallet）

有 5 个涉及竞争条件的数据，
jetton_balance,owned_nft_number,owned_nft_dict,pending_reduce_jetton_balance,Pending_transfer_nft_queue.
对于 jetton_balance，在 Step3.4 中，jetton_balance 可能会执行 jetton_balance + 1 - pending_reduce_jetton_balance 或设置 jetton_balance=0，从合约内部看，这都是原子操作，不存在竞争冲突。

对于 owned_nft_number，在 Step3.4 中，owned_nft_number 可能加 1 或保持不变，从合约内部看，这都是原子操作，不存在竞争冲突。

对于 owned_nft_dict，在 Step3.4 中，将添加一个 NFT index 到 owned_nft_dict 中或保持不变。如果发生竞争条件冲突，则表示 FT1 Step3.4 要添加的 NFT Item index 和 FT2 Step3.4 要添加的 NFT Item index 相同。但在 Step3.2 中，需要 mint 的 NFT index 来源于 collection 合约的 next_item_index,并且每 mint 一个新的 NFT，next_item_index +1，因此 FT1 Step3.4 和 FT2 Step3.4 将要添加的 NFT Item index，不可能一样，亦即不会发生冲突。

对于 pending_reduce_jetton_balance，在 Step3.4 中，pending_reduce_jetton_balance 可能会执行 pending_reduce_jetton_balance - jetton_balance -1 或保持不变，从合约内部看，这都是原子操作，不存在竞争冲突。

对于 Pending_transfer_nft_queue，在 Step3.4 中，Pending_transfer_nft_queue 将尝试删除一个 NFT item index。如果发生竞争条件冲突，则表示 FT1 Step3.4 要删除的 NFT Item index 和 FT2 Step3.4 要删除的 NFT Item index 相同。但在 Step3.2 中，需要 mint 的 NFT index 来源于 collection 合约的 next_item_index,并且每 mint 一个新的 NFT，next_item_index +1，因此 FT1 Step3.4 和 FT2 Step3.4 将要删除的 NFT Item index，不可能一样，亦即不会发生冲突。

3.1.1.10 FT1 Step3.4 和 FT2 Step3.4.3 (TRC404 Wallet)

有 1 个涉及竞争条件的数据，owned_nft_dict

对于 owned_nft_dict，在 Step3.4 中，将添加一个 NFT index 到 owned_nft_dict 中或保持不变。在 3.4.3 中，将在 owned_nft_dict 中删除一个 NFT index。

如果发生竞争条件冲突，则表示 FT1 Step3.4 要添加的 NFT Item index 和 FT2 Step3.4.3 要删除的 NFT Item index 相同。但在 Step3.2 中，需要 mint 的 NFT index 来源于 collection 合约的 next_item_index,并且每 mint 一个新的 NFT，next_item_index +1，因此 FT1 Step3.4 和 FT2 Step3.4.3 将要操作的 NFT Item index，不可能一样，亦即不会发生冲突。

3.1.1.11 FT1 Step3.4.2 和 FT2 Step3.4.2 (NFT Item)

有 1 个涉及竞争条件的数据，contract active status

对于 contract active status，Step3.4.2 执行的操作是 burn NFT，从而 contract active status 会变成 non-active.

这两个步骤不会发生竞争条件冲突的原因，和 3.1.1.10 描述的类似。

3.1.1.12 FT1 Step3.4.3 和 FT2 Step3.4.3 (TRC404 Wallet)

有 1 个涉及竞争条件的数据，owned_nft_dict。

对于 owned_nft_dict，在 3.4.3 中，将尝试在 owned_nft_dict 中删除一个 NFT index（可能会删除失败，因为 index 不存在）。

这两个步骤不会发生竞争条件冲突的原因，和 3.1.1.10 描述的类似。

3.1.2 A transfer FT to B 和 B transfer FT to C 竞争分析

假设 B 一开始有 0 FT 和 0 NFT。FT1 ：A transfer FT to B 和 FT2: B transfer FT to C 交易可能存在的竞争节点分析如下表所示：

	FT2(B transfer FT to C)	Step 1(Sender TRC404wallet)	Step 2.1(collection)	Step 2.2(NFT Item)	Step2.3(Sender TRC404wallet)	Step3.1(Receiver TRC404 wallet)	Step3.2(collection)	Step3.3(NFT Item)	Step3.4(Receiver TRC404 wallet)	Step3.4.1 (collectiton)	Step3.4.2(NFT Item)	Step3.4.3(Receiver TRC404 wallet)
2	FT1(A transfer FT to B)											
3	Step 1(Sender TRC404wallet)											
4	Step 2.1(collection)											
5	Step 2.2(NFT Item)											
6	Step 2.3(Sender TRC404wallet)											
7	Step 3.1(Receiver TRC404 wallet)	jetton_balance										
8	Step 3.2(collection)											
9	Step 3.3(NFT Item)			contract active status								
10	Step 3.4(Receiver TRC404 wallet)	jetton_balance, owned_nft_number, owned_nft_dict			owned_nft_dict							
11	Step 3.4.1(collectiton)											
12	Step 3.4.2(NFT Item)			contract active status								
13	Step 3.4.3(Receiver TRC404 wallet)	owned_nft_dict			owned_nft_dict							

注意：和 3.1.1 类似，涉及 collection 合约的数据更新不再分析，只分析 TRC404 Wallet 合约和 NFT Item 合约可能发生的冲突。

3.1.2.1 FT1 Step3.1 和 FT2 Step1（TRC404 Wallet）

有 1 个涉及竞争条件的数据，jetton_balance

对于 jetton_balance，Step3.1 执行的操作是，合约直接尝试加上 internal_transfer 请求包含的 amount,再减去需要 mint 的 NFT 的数量。在 Step1 中，合约直接尝试减去 transfer 请求包含的 amount，如果结果小于 0，则直接退出，不再执行后续操作。

Jetton_balance 的操作都是原子操作，不存在竞争条件问题。

3.1.2.2 FT1 Step3.3 和 FT2 Step2.2（NFT Item）*

有 1 个涉及竞争条件的数据，contract active status

对于 contract active status，Step3.3 执行的操作是 mint NFT，从而 contract active status 会变成 active. Step2.2 执行的操作是 burn NFT，从而 contract active status 会变成 non-active.

如果发生竞争条件冲突，则需要 Step3.3 操作的 NFT 和 Step2.2 执行的操作的 NFT 是同一个 NFT。但在 FT1 的 Step3.1 中，添加 NFT index 到 owned_nft_dict，并不是在 Step3.1 中写入，而是需要等执行完 Step3.2, Step3.3 和 Step3.4，才会加上对应的 NFT balance 和把 NFT item index 添加到 owned_nft_limit 中。因此对于 FT2 Step2.2,如果想令 Step2.2 burn 的 NFT index 和 FT1 Step3.3 mint 的 NFT index 一样，必须要等 FT1 Step3.4 执行完才有可能。而如果 FT1 Step3.4 已经执行完，那么执行 FT2 的 Step2.2 时，即使是同一个 NFT，也是正常操作，不

会出现类似 NFT 还没部署成功就尝试去 burn 这个 NFT 合约的异常情况。因此，这两个步骤不会有竞争条件冲突。

3.1.2.3 FT1 Step3.4 和 FT2 Step1（TRC404 Wallet）*

有 3 个涉及竞争条件的数据，jetton_balance，owned_nft_number,owned_nft_dict.

对于 jetton_balance，Step3.4 中，jetton_balance 可能会执行 jetton_balance + 1 - pending_reduce_jetton_balance 或设置 jetton_balance=0。Step1 中，合约直接尝试减去 transfer 请求包含的 amount，如果结果小于 0，则直接退出，不再执行后续操作。这都是原子操作，不会有竞争问题。

对于 owned_nft_number，Step3.4 中，owned_nft_number 可能加 1 或保持不变。在 Step1 中，owned_nft_number 会减去需要 burn 的 nft 数量。这都是原子操作，不会有竞争问题。

对于 owned_nft_dict，在 Step3.4 中，将添加一个 NFT index 到 owned_nft_dict 中或保持不变。在 Step1 中，将找出需要 burn NFT 的状态正常的 NFT index，并且设置这些 NFT index 的状态为 pending_delete。

关于 owned_nft_dict 的情况比较复杂，下面具体分析：

如果 FT2 Step1 先执行，Step3.4 后执行，则 Step1 操作的 NFT index 一定和 Step3.4 的不同。

如果 FT2 Step3.4 先执行，Step1 后执行，则可能存在 Step1 将要设置的 NFT index 和 Step3.4 添加的 NFT index 一样，但 Step3.4 已经执行完后，已完成所有的操作，因此即使是同一个 NFT，也是正常操作，不会出现类似 FT1 需要 mint 的 NFT 还没部署成功，FT2 就尝试去 burn 这个 NFT 合约等异常情况。综上所述，这两个步骤不会有竞争条件冲突。

3.1.2.4 FT1 Step3.4 和 FT2 Step2.3（TRC404 Wallet）*

有 1 个涉及竞争条件的数据， owned_nft_dict.

对于 owned_nft_dict，在 Step3.4 中，将添加一个 NFT index 到 owned_nft_dict 中或保持不变。在 Step2.3 中，将从 owned_nft_dict 删除指定的 NFT item index。

这两个步骤不会发生竞争条件冲突的原因，和 3.1.2.3 描述的类似。

3.1.2.5 FT1 Step3.4.2 和 FT2 Step2.2（NFT Item）*

有 1 个涉及竞争条件的数据，contract active status

对于 contract active status，Step3.4.2 执行的操作是 burn NFT，从而 contract active status 会变成 non-active. Step2.2 执行的操作是 burn NFT，从而 contract active status 会变成 non-active.

如果发生竞争条件冲突，则需要 Step3.4.2 操作的 NFT 和 Step2.2 操作的 NFT 是同一个 NFT。但如果进入 FT1 Step 3.4.2,则表示在 Step3.4 中，合约并没有把 NFT index 添加到 owned_nft_dict 中，因此 FT2 的 Step1 不可能获得 Step 3.4.2 将要 burn NFT 的 index，即 Step3.4.2 操作的 NFT 和 Step2.2 操作的 NFT 一定不同，所以不会有竞争条件冲突。

3.1.2.6 FT1 Step3.4.3 和 FT2 Step1（TRC404 wallet）*

有 1 个涉及竞争条件的数据， owned_nft_dict.

对于 owned_nft_dict，在 Step3.4.3 中，将尝试在 owned_nft_dict 中删除一个指定的 NFT index（可能会删除失败，因为 index 不存在）。在 Step1 中，将找出需要 burn NFT 的状态正常的 NFT index，并且设置这些 NFT index 的状态为 pending_delete。

如果发生竞争条件冲突，则需要 Step3.4.3 操作的 NFT 和 Step1 操作的 NFT 是同一个 NFT。但如果进入 FT1 Step 3.4.3,则表示在 Step3.4 中，合约并没有把 NFT index 添加到 owned_nft_dict 中，因此 FT2 的 Step1 不可能获得 Step 3.4.3 将要删除的 NF Tindex，即 Step3.4.3 操作的 NFT 和 Step1 操作的 NFT index 一定不同，所以不会有竞争条件冲突。

3.1.2.7 FT1 Step3.4.3 和 FT2 Step2.3（TRC404 wallet）*

有 1 个涉及竞争条件的数据， owned_nft_dict.

对于 owned_nft_dict，在 Step3.4.3 中，将尝试在 owned_nft_dict 中删除一个指定的 NFT index（可能会删除失败，因为 index 不存在）。在 Step2.3 中，将尝试在在 owned_nft_dict 中删除一个指定的 NFT index。

这两个步骤不会发生竞争条件冲突的原因和 3.1.2.6 类似。

3.2 transfer NFT 的竞争分析

3.2.1 A transfer NFT to B 和 A transfer NFT to C 竞争分析

因为 Transfer NFT 的入口合约为 NFT Item，NFT Item 收到 transfer_nft_item 消息，首先会检查发送者是否是 NFT 的 owner，如果不是，则退出，不会继续处理。因此，对于 A transfer NFT to B 和 A transfer NFT to C，无论哪条交易先执行，只会有一条交易可以成功更改 NFT 的 owner，另一条交易一定会执行失败，然后退出。因为这两类交易一定不会存在竞争条件的冲突。

3.2.2 A transfer NFT to B 和 B transfer NFT to C 竞争分析

NFT1: A transfer NFT to B 和 NFT2: B transfer FT to C 交易可能存在的竞争节点分析如下表所示：

	NFT2(B transfer FT to C)	Step 1(NFT Item)	Step 2(collection)	Step3(Sender(B) TRC404wallet)	Step4(Receiver(C) TRC404 wallet)	Step4.1 (collectiton)	Step4.2(NFT Item)	Step4.3(Receiver(C) TRC404 wallet)
2	NFT1(A transfer FT to B)							
3	Step 1(NFT Item)	owner					contract active status	
4	Step 2(collection)							
5	Step 3(Sender (A) TRC404wallet)							
6	Step 4(Receiver (B) TRC404 wallet)			jetton_balance, owned_nft_number owned_nft_dict				
7	Step 4.1(collectiton)							
8	Step 4.2(NFT Item)	contract active status					contract active status	
9	Step 4.3(Receiver(B) TRC404 wallet)			jetton_balance, owned_nft_number owned_nft_dict				

注意：和 3.1.1 类似，collection 合约的数据更新不再分析，只分析 TRC404 Wallet 合约和 NFT Item 合约可能发生的冲突。

3.2.2.1 NFT1 Step1 和 NFT2 Step1（NFT Item）

有 1 个涉及竞争条件的数据， owner。
对于 owner，在 Step1 中，会 transfer ownership from sender to receiver。 因此如果 NFT1 先执行，NFT2 后执行，两条交易在 Step1 都会执行成功，不存在竞争条件冲突。如果 NFT2 先执行，NFT1 后执行，则 NFT2 会失败，因为此时 NFT 的 owner 还是 A，还不是 B。综上，不存在竞争条件冲突。

3.2.2.2 NFT1 Step1 和 NFT2 Step4.2（NFT Item）

有 1 个涉及竞争条件的数据， contract_active_status.
对于 contract_active_status， Step1 只会更改 owner，不会更改合约 active status， Step 4.2 会 burn NFT，因此 contract_active_status 会变成 non-active。
这两个步骤不会发生竞争条件冲突的原因和 3.2.1.1 类似。

3.2.2.3 NFT1 Step4 和 NFT2 Step3（TRC404 wallet）***

有 5 个涉及竞争条件的数据，
jetton_balance,owned_nft_number,owned_nft_dict,pending_jetton_balance, pending_transfer_nft_queue.

考虑以下两种情况：
第一种情况：NFT1 Step4 先执行，NFT2 Step3 后执行。这时因为 Step4 已添加 index 到 B 的 owned_nft_dict，并且 B 的 jetton_balance+1，owned_nft_number+1，所以 NFT2 Step3 可以成功从 owned_nft_dict 删除这个 NFT index，可以成功执行 jetton_balance-1 和 owned_nft_number-1。

第二种情况：NFT2 Step3 先执行，NFT1 Step4 后执行。这时因为 Step4 还没添加 index 到 B 的 owned_nft_dict，并且 B 的 jetton_balance 还没+1，owned_nft_number 也还没+1，所以：
（1）NFT2 Step3 执行从 B 的 owned_nft_dict 删除这个 NFT index 会失败，因此把需要 transfer 的 index 放入 pending_transfer_nft_queue.
（2）另外因为此时 B 的 jetton_balance 为 0，再-1 则会负数，因此令 pending_jetton_balance +1，jetton_balance 仍为 0。

（3）对于 owned_nft_number，此时 owned_nft_number 为 0，且 owned_nft_number 为 int 类型，可以存储负数值，因此 owned_nft_number-1 为-1.

等 NFT2 Step3 执行完后，NFT1 Step4 将执行：

jetton_balance+1-pending_jetton_balance，结果为 0，符合预期。

owned_nft_number +1，结果为 0，也符合预期。

从 pending_transfer_nft_queue 中删除需要 transfer 的 NFT index。

（注：为什么需要增加 pending_jetton_balance 和 pending_transfer_nft_queue 属性？主要为了兼容以下的情况：B 当前的 FT 和 NFT 数量为 0，当 A transfer NFT to B，B transfer NFT to C 时，两条交易都会执行成功。在 getgems.io 平台 put NFT on sale 时，就需要执行 A transfer NFT to B 和 B transfer NFT to B 的情况。实现这两条交易都可以执行成功，才能兼容 getgems.io 等类似的 NFT marketplace 平台。 ）

3.2.2.4 NFT1 Step4.2 和 NFT2 Step1（TRC404 wallet）***

有 1 个涉及竞争条件的数据， contract active status

考虑以下两种情况：

第一种情况：NFT1 Step4.2 先执行，NFT2 Step1 后执行。因为执行 NFT1 Step4.2 后，NFT 会被 burn 了，因此 NFT2 Step1 一定会执行失败，因为 NFT 的状态已经是 non-active。

第二种情况：NFT2 Step1 先执行，NFT1 Step4.2 后执行。NFT1 Step4.2 后，会在 B 的 wallet 合约里添加这个 NFT index 到 owned_nft_dict,并设置 NFT index 的状态为 pending_delete.所以当 NFT2 执行到 Step3 时，如果发现这个 NFT index 的状态为 pending_delete，会直接终止后面的步骤。也就是说，第二种情况，NFT2 也会执行失败。

综上，两种情况都符合预期（1 个会被 burn 的 NFT 不应该可以成功 transfer 给其他用户），不会出现因竞争条件冲突而出现异常情况。

3.2.2.5 NFT1 Step4.2 和 NFT2 Step4.2（TRC404 wallet）

有 1 个涉及竞争条件的数据， contract active status

这两个步骤不会同时出现，因为只要有任意一个交易进入到 Step4.2，另外一个交易 Step3 就一定会失败，因此不会出现竞争条件冲突。

3.2.2.6 NFT1 Step4.3 和 NFT2 Step3（TRC404 wallet）

有 1 个涉及竞争条件的数据，owned_nft_dict

考虑以下二种情况：
第一种情况，NFT1 Step4.3 先执行，NFT2 Step3 后执行，如果 NFT1 Step4.3 已经执行完，表示 NFT 已被 burn，所以 NFT2 在 Step1 就已经会失败，无法进入 Step3.

第二种情况：NFT2 Step3 先执行，NFT1 Step4.3 后执行，NFT2 可以进入 Step3，代表 NFT2 的 Step1 一定在 NFT1 的 Step4.2 之前执行，否则，NFT2 因为 NFT 已经被 burn，无法进入 Step3.
对于第二种情况，又分两种子情况：
第一种子情况，NFT2 执行 Step3 时，NFT1 已执行 Step4.这种情况因为 NFT1 的 step4 会把 NFT index 添加到 owned_nft_dict,并设置 index 状态为 pending_delete,所以 NFT2 执行 Step3 一定会失败。
第二种子情况，NFT2 执行 Step3 时，NFT1 还没执行 Step4。此时 NFT2 的 Step3，会把 NFT index 放入 pending_transfer_nft_queue, pending_jetton_balance+1,owned_nft_number-1,发 addOneFtAndNFT 消息给 C 的 TRC404 wallet 合约。此时如果 NFT1 执行 Step4，会从 pending_transfer_nft_queue 删除 NFT index，令 pending_jetton_balance=0，owned_nft_number+1，而不会再进入 NFT1 4.1，4.2 和 4.3 的步骤。
综上，所有情况都符合预期，不会发生因为竞争条件冲突而产生异常情况。

3.3 transfer FT 和 Transfer NFT 的竞争分析

3.3.1 A transfer FT to B 和 A transfer NFT to B 竞争分析

FT1: A transfer FT to B 和 NFT2:A transfer NFT to B 交易可能存在的竞争节点分析如下表所示：

	NFT2(A transfer NFT to B)		Step 1(NFT Item)	Step 2 (collection)	Step3(Sender(A) TRC404wallet)	Step4(Receiver(B) TRC404 wallet)	Step4.1 (collectiton)	Step4.2(NFT Item)	Step4.3(Receiver(B) TRC404 wallet)
2	Involved data:		owner	/	jetton_balance, owned_nft_number, owned_nft_dict, pending_reduce_jetton_balance, Pending_transfer_nft_queue	jetton_balance, owned_nft_number, owned_nft_dict, pending_reduce_jetton_balance Pending_transfer_nft_queue	current_total_supply	contract active status	owned_nft_dict
3	FT1(A transfer FT to B)	Involed data							
4	Step 1(Sender(A) TRC404wallet)	jetton_balance, owned_nft_number, owned_nft_dict			jetton_balance owned_nft_number owned_nft_dict				
5	Step 2.1(collection)	current_total_supply							
6	Step 2.2(NFT Item)	contract active status	contract active status					contract active status	
7	Step 2.3(Sender(A) TRC404wallet)	owned_nft_dict			owned_nft_dict				
8	Step 3.1(Receiver(B) TRC404 wallet)	jetton_balance				jetton_balance			
9	Step 3.2(collection)	`current_total_supply,` `next_item_index`							
10	Step 3.3(NFT Item)	owner contract active status	contract active status					contract active status	
11	Step 3.4(Receiver(B) TRC404 wallet)	jetton_balance, owned_nft_number, owned_nft_dict, pending_reduce_jetton_balance Pending_transfer_nft_queue				jetton_balance, owned_nft_number, owned_nft_dict, pending_reduce_jetton_balance Pending_transfer_nft_queue			owned_nft_dict,
12	Step 3.4.1(collectiton)	current_total_supply							
13	Step 3.4.2(NFT Item)	contract active status	contract active status					contract active status	
14	Step 3.4.3(Receiver(B) TRC404 wallet)	owned_nft_dict				owned_nft_dict			owned_nft_dict

注意：和 3.1.1 类似，collection 合约的数据更新不再分析，只分析 TRC404 Wallet 合约和 NFT Item 合约可能发生的冲突。

3.3.1.1 FT1 Step1 和 NFT2 Step3（TRC404 wallet）

有 3 个涉及竞争条件的数据，
jetton_balance,owned_nft_number,owned_nft_dict。

对于 jetton_balance,owned_nft_number，FT1 Step1 和 NFT2 Step3 执行的都是减操作，因为都是原子操作，不会发生冲突。

对于,owned_nft_dict，FT1 Step1 执行的操作是设置 NFT index 状态为 pending_delete，在 NFT2 Step3，执行的操作是从 owned_nft_dict 中删除这个 NFT index 或将 index 放入 pending_transfer_nft_queue,或保持不变，直接退出并发送消息通知后 NFT 已经被 burn。

考虑以下两种情况：第一种情况，NFT2 Step3 先执行，FT1 Step1 后执行，因为 NFT2 Step3 会将 NFT index 删除，因此 FT1 Step1 操作的 NFT index 不会和 NFT2 Step3 操作的 index 相同。第二种情况，FT1 Step1 先执行，NFT2 Step3 后执行，因为 FT1 Step1 会将 NFT index 状态设置为 pending_delete，因为 NFT2 Step3 执行时，会因为操作的 NFT index 已经 pending_delete 而退出，符合正常预期（已经被 burn 的 NFT 不能 transfer 给其他人。）

综上，所有情况都符合预期，不会发生因为竞争条件冲突而产生异常情况。

3.3.1.2 FT1 Step2.2 和 NFT2 Step1（NFT Item）

有 1 个涉及竞争条件的数据，contract active status。
考虑以下两种情况：第一种情况，FT1 Step2.2 先执行，NFT2 Step1 后执行，如果 FT1 Step2.2 先执行，则 NFT 已被 burn，因此 NFT2 Step1 一定会执行失败，符合预期。第二种情况， NFT2 Step1 先执行，FT1 Step2.2 后执行，FT1 可以进入 Step2.2，表示 FT1 在 Step1 时已把 NFT index 状态为 pending_delete，因此 NFT2 即使可以执行完 Step1，在 NFT2 Step3 执行时也会失败，原因和 3.3.1.1 一样。
综上，所有情况都符合预期，不会发生因为竞争条件冲突而产生异常情况。

3.3.1.3 FT1 Step2.2 和 NFT2 Step4.2（NFT Item）

有 1 个涉及竞争条件的数据，contract active status
这两个步骤不会发生冲突，原因和 3.3.1.2 一样。

3.3.1.4 FT1 Step2.3 和 NFT2 Step3（TRC404 wallet）

有 1 个涉及竞争条件的数据，owned_nft_dict
这两个步骤不会发生冲突，原因和 3.3.1.2 类似。

3.3.1.5 FT1 Step3.1 和 NFT2 Step4（TRC404 wallet）

有 1 个涉及竞争条件的数据，jetton_balance
对于 jetton_balance，FT1 Step3.1 和 NFT2 Step4 执行的都是加操作，因为都是原子操作，不会发生竞争条件冲突。

3.3.1.6 FT1 Step3.3 和 NFT2 Step1（NFT Item）

有 1 个涉及竞争条件的数据，contract active status

FT1 Step3.3 是为 B 新 mint 的 NFT，和 NFT2 Step1 操作的 A 的 NFT，一定不是同一个 NFT，因此不会发生冲突。

3.3.1.7 FT1 Step3.3 和 NFT2 Step4.2 (NFT Item)

有 1 个涉及竞争条件的数据，contract active status.
这两个步骤不会发生冲突，原因和 3.3.1.6 类似.

3.3.1.8 FT1 Step3.4 和 NFT2 Step4 (TRC404 wallet)

有 5 个涉及竞争条件的数据，
jetton_balance,owned_nft_number,owned_nft_dict,pending_reduce_jetton_balance,pending_transfer_nft_queue.

对于 jetton_balance,owned_nft_number, pending_reduce_jetton_balance,两个步骤执行的都是加或减的操作，不会发生冲突。

对于 owned_nft_dict 和 pending_transfer_nft_queue，和 3.3.1.6 类似，操作的一定不是同一个 NFT index，因此也不会发生冲突。

3.3.1.9 FT1 Step3.4 和 NFT2 Step4.3 (TRC404 wallet)

有一个涉及竞争条件的数据，owned_nft_dict。
这两个步骤不会发生冲突，原因和 3.3.1.6 类似.

3.3.1.10 FT1 Step3.4.2 和 NFT2 Step1 (NFT Item)

有一个涉及竞争条件的数据，contract active status。
这两个步骤不会发生冲突，原因和 3.3.1.6 类似.

3.3.1.11 FT1 Step3.4.2 和 NFT2 Step4.2 (NFT Item)

有一个涉及竞争条件的数据，contract active status。
这两个步骤不会发生冲突，原因和 3.3.1.6 类似.

3.3.1.12 FT1 Step3.4.3 和 NFT2 Step4 (TRC404 wallet)

有一个涉及竞争条件的数据，owned_nft_dict。
这两个步骤不会发生冲突，原因和 3.3.1.6 类似.

3.3.1.13 FT1 Step3.4.3 和 NFT2 Step4.3 (NFT Item)

有一个涉及竞争条件的数据，owned_nft_dict。
这两个步骤不会发生冲突，原因和 3.3.1.6 类似.

3.3.2 A transfer FT to B 和 B transfer NFT to C 竞争分析

FT1: A transfer n(n>=1) FT to B 和 NFT2:B transfer NFT to C 交易可能存在的竞争节点分析如下表所示：

	NFT2(B transfer NFT to C)		Step 1(NFT Item)	Step 2 (collection)	Step3(Sender(B) TRC404wallet)	Step4(Receiver(C) TRC404 wallet)	Step4.1 (collectiton)	Step4.2(NFT Item)	Step4.3(Receiver(C) TRC404 wallet)
2	Involved data:		owner	/	jetton_balance, owned_nft_number, owned_nft_dict, pending_reduce_jetton_balance, Pending_transfer_nft_queue	jetton_balance, owned_nft_number, owned_nft_dict, pending_reduce_jetton_balance Pending_transfer_nft_queue	current_total_supply	contract active status	owned_nft_dict
3	FT1(A transfer FT to B)	Involed data							
4	Step 1(Sender(A) TRC404wallet)	jetton_balance, owned_nft_number, owned_nft_dict							
5	Step 2.1(collection)	current_total_supply							
6	Step 2.2(NFT Item)	contract active status							
7	Step 2.3(Sender(A) TRC404wallet)	owned_nft_dict							
8	Step 3.1(Receiver(B) TRC404 wallet)	jetton_balance			jetton_balance				
9	Step 3.2(collection)	`current_total_supply,` `next_item_index`							
10	Step 3.3(NFT Item)	owner contract active status	owner contract active status					contract active status	
11	Step 3.4(Receiver(B) TRC404 wallet)	jetton_balance, owned_nft_number, owned_nft_dict, pending_reduce_jetton_balance Pending_transfer_nft_queue			jetton_balance, owned_nft_number, owned_nft_dict, pending_reduce_jetton_balance Pending_transfer_nft_queue				owned_nft_dict
12	Step 3.4.1(collectiton)	current_total_supply							
13	Step 3.4.2(NFT Item)	contract active status	contract active status					contract active status	
14	Step 3.4.3(Receiver(B) TRC404 wallet)	owned_nft_dict			owned_nft_dict				owned_nft_dict

注意：和 3.1.1 类似，collection 合约的数据更新不再分析，只分析 TRC404 Wallet 合约和 NFT Item 合约可能发生的冲突。

3.3.2.1 FT1 Step3.1 和 NFT2 Step3（TRC404 wallet）

有 1 个涉及竞争条件的数据，jetton_balance。
这两个步骤不会发生冲突，原因和 3.2.2.3 类似.

3.3.2.2 FT1 Step3.3 和 NFT2 Step1（NFT Item）

有 2 个涉及竞争条件的数据，owner ,contract active status。
考虑以下两种情况：第一种情况，FT1 Step3.3 先执行，NFT2 Step1 后执行，如果 FT1 Step3.3 先执行，则 NFT 已 mint，并设置 owner 为 B，因此 NFT2 Step1 会执行成功，符合预期。第二种情况， NFT2 Step1 先执行，FT1 Step3.3 后执行，此时因为 NFT 还没 mint，因此 NFT2 Step1 一定会实行失败，也符合预期

3.3.2.3 FT1 Step3.3 和 NFT2 Step4.2（NFT Item）

有 2 个涉及竞争条件的数据，owner ,contract active status。
考虑以下两种情况：第一种情况，FT1 Step3.3 先执行，NFT2 Step4.2 后执行，如果 FT1 Step3.3 先执行，则 NFT 已 mint，并设置 owner 为 B，因此 NFT2 Step4.2 会执行成功，符合预期。第二种情况， NFT2 Step4.2 先执行，FT1 Step3.3 后执行，此时因为 NFT 还没 mint，因此 NFT2 Step4.2 一定会执行失败，也符合预期。

3.3.2.4 FT1 Step3.4 和 NFT2 Step3（TRC404 wallet）

有 5 个涉及竞争条件的数据，
jetton_balance,owned_nft_number,owned_nft_dict,pending_jetton_balance,
pending_transfer_nft_queue.

这两个步骤不会发生冲突，原因和 3.2.2.3 类似.

3.3.2.5 FT1 Step3.4 和 NFT2 Step4.3（TRC404 wallet）

有 1 个涉及竞争条件的数据， owned_nft_dict
这两个步骤不会发生冲突，原因和 3.2.2.3 类似。

3.3.2.6 FT1 Step3.4.2 和 NFT2 Step1（NFT Item）

有 1 个涉及竞争条件的数据， contract active status
考虑以下两种情况：第一种情况，FT1 Step3.4.2 先执行，NFT2 Step1 后执行，因为 FT1 Step3.4.2 会 burn NFT，因此 NFT2 Step1 一定会执行失败，符合预期。第二种情况，FT2 Step1 先执行，FT1 Step3.4.2 后执行，FT1 可以进入 Step3.4.2，表示 Step3.4 时已经把这个 NFT index 的状态设置为 pending_delete,因此即使 NFT2 Step1 执行成功，当执行到 NFT2 Step3 时也一定会失败，符合预期。

3.3.2.7 FT1 Step3.4.2 和 NFT2 Step4.2（NFT Item） *

有 1 个涉及竞争条件的数据， contract active status
这两个步骤不会同时出现，因为如果 FT1 进入到 Step3.4.2，则 NFT2 执行到 Step3 一定会失败。如果 NFT2 能够执行到 Step4.2，则 FT1 执行到 Step3.4 就会退出，不会执行后续的 Step3.4.2。综上，这两个不步骤不会出现竞争条件冲突。

3.3.2.8 FT1 Step3.4.2 和 NFT2 Step3（TRC404 wallet） *

有 1 个涉及竞争条件的数据， owned_nft_dict
这两个步骤不会同时出现，原因和 3.3.2.7 类似。

3.3.2.9 FT1 Step3.4.2 和 NFT2 Step4.3（TRC404 wallet） *

有 1 个涉及竞争条件的数据， owned_nft_dict
这两个步骤不会同时出现，原因和 3.3.2.7 类似。

3.3.3 A transfer NFT to B 和 B transfer FT to C 竞争分析

NFT1: A transfer NFT to B 和 FT2:B transfer FT to C 交易可能存在的竞争节点分析如下表所示：

	FT2(B transfer FT to C)		Step 1(Sender(B) TRC404wallet)	Step 2.1 (collection)	Step 2.2(NFT Item)	Step2.3(Sender(B) TRC404wallet)	Step3.1(Receiver (C) TRC404 wallet)	Step3.2(collection)	Step3.3(NFT Item)	Step3.4(Receiver(C) TRC404 wallet)	Step3.4.1 (collectiton)	Step3.4.2(NFT Item)	Step3.4.3(Receiver(C) TRC404 wallet)
1	Involved data:		jetton_balance, owned_nft_number owned_nft_dict	current_tot al_supply	contract active status	owned_nft_dict	jetton_balance	current_total_supply, next_item_index	owner contract active status	jetton_balance, owned_nft_number, owned_nft_dict, pending_reduce_jetton_balance Pending_transfer_nft_queue	current_total_supply	contract active status	owned_nft_dict
2	NFT1(A transfer NFT to B)	Involved data											
3	Step 1(NFT Item)	owner			contract active status								
4	Step 2(collection)	/											
5	Step 3(Sender (A) TRC404wallet)	jetton_balance, owned_nft_number, owned_nft_dict, pending_reduce_jetton_balance Pending_transfer_nft_queue											
6	Step 4(Receiver (B) TRC404 wallet)	jetton_balance, owned_nft_number, owned_nft_dict, pending_reduce_jetton_balance Pending_transfer_nft_queue	jetton_balance, owned_nft_number owned_nft_dict			owned_nft_dict							
7	Step 4.1(collectiton)	current_total_supply											
8	Step 4.2(NFT Item)	contract active status			contract active status								
9	Step 4.3(Receiver(B) TRC404 wallet)	owned_nft_dict	owned_nft_dict			owned_nft_dict							
10													

注意：和 3.1.1 类似，collection 合约的数据更新不再分析，只分析 TRC404 Wallet 合约和 NFT Item 合约可能发生的冲突。

3.3.3.1 NFT1 Step1 和 FT2 Step2.2（NFT Item）

有 1 个涉及竞争条件的数据， contract active status

考虑以下两种情况：第一种情况，NFT1 Step1 先执行，FT2 Step2.2 后执行。因为 NFT1 Step1 会把 NFT 的 owner 从 A 更改成 B，但还没执行到 NFT1 Step4 的增加这个 index 到 owned_nft_dict,所以 FT2 Step2.2 操作的 NFT Item 不会和 NFT1 Step1 操作的 NFT 一样，即不会发生冲突。第二种情况，FT2 Step2.2 先执行，NFT1 Step1 后执行。和第一种情况类似，FT2 Step2.2 操作的 NFT Item 不会和 NFT1 Step1 操作的 NFT item 一样，即不会发生冲突。

3.3.3.2 NFT1 Step4 和 FT2 Step1（TRC404 wallet）

有 3 个涉及竞争条件的数据，jetton_balance,owned_nft_number,owned_nft_dict

对于 jetton_balance,owned_nft_number，两个步骤执行的都是加或减的操作，不会发生冲突。

对于 owned_nft_dict，有两种情况，情况 1，如果 NFT1 Step1 先执行，FT2 Step1 后执行，两个步骤操作的可以是同一个 NFT index，此时表示 NFT1 已经完成交易，因此 FT2 transfer FT 时选择 burn 这个 NFT 是正常流程。情况 2，如果如果 FT2 Step1 先执行，NFT1 Step1 后执行，两个步骤操作的 NFT index 一定是不同的，不会发生冲突。

3.3.3.3 NFT1 Step4 和 FT2 Step2.3（TRC404 wallet）

有 1 个涉及竞争条件的数据，owned_nft_dict。
这两个步骤不会同时出现，原因和 3.3.3.2 类似。

3.3.3.4 NFT1 Step4.2 和 FT2 Step2.2（NFT Item）

有 1 个涉及竞争条件的数据， contract active status
这两个步骤不会同时出现，原因和 3.3.3.2 类似。

3.3.3.5 NFT1 Step4.3 和 FT2 Step1（TRC404 wallet）

有 1 个涉及竞争条件的数据， owned_nft_dict
对于 owned_nft_dict，有两种情况，情况 1，如果 NFT1 Step4.3 先执行，FT2 Step1 后执行，如果 NFT1 Step4.3 已执行完，表示 NFT 已被 burn，因此 FT2 的 Step1 操作的 index 一定和 NFT1 Step4.3 操作的 index 不同。情况 2，如果 FT2 Step1 先执行，NFT1 Step4.3 后执行.FT2 的 Step1 会把 NFT index 设置成 pending_delete 状态，因此 NFT1 执行到 Step 4 就会终止，不会进入 NFT1 Step4.3。

综上，这两个步骤不会发生因为竞争条件冲突而产生异常情况。

3.3.3.6 NFT1 Step4.3 和 FT2 Step2.3（TRC404 wallet）

有 1 个涉及竞争条件的数据， owned_nft_dict
这两个步骤不会发生因为竞争条件冲突而产生异常情况，原因和 3.3.3.5 类似。