フロー

概要

1. バイトコードを1つ受け取りprimary_contractsに追加(主たるコントラクト，一番はじめに実行されるコントラクト)
   バイトコードを0以上受け取りsecondary_contractsに追加(従たるコントラクト)

2. WorldStateを初期化

3. 各バイトコードに対してコントラクトアカウントを生成，アカウントアドレスを表すシンボル変数を生成し，WorldState内のaccountsにコントラクトアカウントとアドレスの対応を記録

4. primary_contracts内のコントラクトを順番に実行
   エッジの集合，ノードの集合 \in CFGmanager, call_stackを初期化
   途中でコントラクト生成があれば，そのアドレスを表すシンボル変数を生成してのバイトコードもprimary_contractsに加えてあとで実行
   外部呼び出しがあった場合は実行中以外の 全てのコントラクトに対して呼び出し処理を行う
   主たるコントラクト，外部呼び出しされるコントラクトそれぞれに対して,以下を実行

    1. 実行環境を初期化
    2. 機械状態，ニーモニックの集合，パス条件，分岐しない場合のジャンプ先，分岐した場合のジャンプ先，\in ブロック を初期化 (一通りの実行環境をブロックとして纏める)
    3. 停止命令に到達するまで実行
        1. プログラムカウンタ \in 機械状態　が指すバイトコードを取得
        2. 命令を実行
           命令がとる引数に応じてプログラムカウンタを進める call系やcreate系命令であればここで外部呼び出し 1.　プログラムカウンタを進め次の命令に合わせる

分岐

分岐する場合

1. 現在の状態をCFGに保存
2. ジャンプ先pc、続行先pcを用意
3. ジャンプ先pcをセットして現在の状態をdfs用スタックにpush
4. 続行先pcをセットして実行

ロールバック時

1. dfs用スタックから状態をpop
2. 実行

createによって新たなコントラクトが生成される場合

戻り先の作成

1. 現在の状態をCFGに保存
2. 戻り先となるpcを用意
3. 現在の状態をreturn_destとして維持

createの初期化コード実行時やcallにおいて外部ジャンプする際

戻り先の作成

1. 現在の状態をCFGに保存
2. 戻り先となるpcを用意
3. ~~現在の状態をreturn_destとして維持~~ ←　再びコールがあると上書きされる
4. 現在の状態をcall_stackにpush

外部コードの状態作成

1. 外部コントラクトの状態をセット
2. storage(, returndata)を空にする
3. 実行

外部から戻るとき(create以外)

1. 現在の状態をCFGに保存
2. call_stackの先頭の状態に現在のstack, memory(, returndata)をセット
3. call_stackからpopした状態を復元
4. 実行

外部から戻るとき(create時)

1. 現在の状態をCFGに保存
2. 現在のpcが参照している命令の次の番地~末端までを新しいコントラクトしてバイトコード群に追加
3. call_stackの先頭の状態に現在のstack, memory(, returndata)をセット
4. call_stackからpopした状態を復元
5. 実行

delegatecallにおいて外部ジャンプする際のBasicBlockの扱い

戻り先の作成

上に同じ

外部コードの状態作成

1. 外部コントラクトの状態をセット
2. 実行

外部から戻るとき

1. 現在の状態をCFGに保存
2. call_stackの先頭の状態に現在のstorage, stack, memory(, returndata)をセット
3. call_stackからpopした状態を復元
4. 実行

終端到達時(return以外)

1. 現在の状態をCFGに保存
2. dfs用スタックが空でなければロールバック