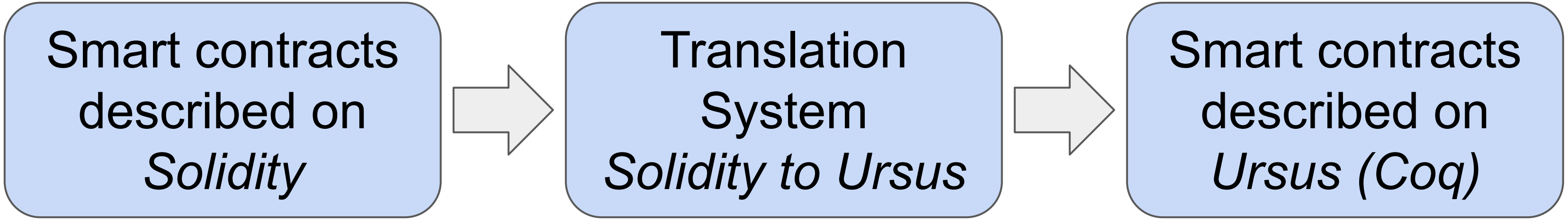
Translation

solidity to ursus

Problem

There is a task of verifying ***smart contracts for solidity***. For this goal, Coq is used, in which **smart contracts** are described in a special **Ursus** language, therefore, in order to verify a specific smart contract, it is necessary to describe it in Ursus. This process requires a lot of **manual labor and special skills**, and there is a human error factor. For this purpose, a **translation system** is being created and developed.

Translation scheme

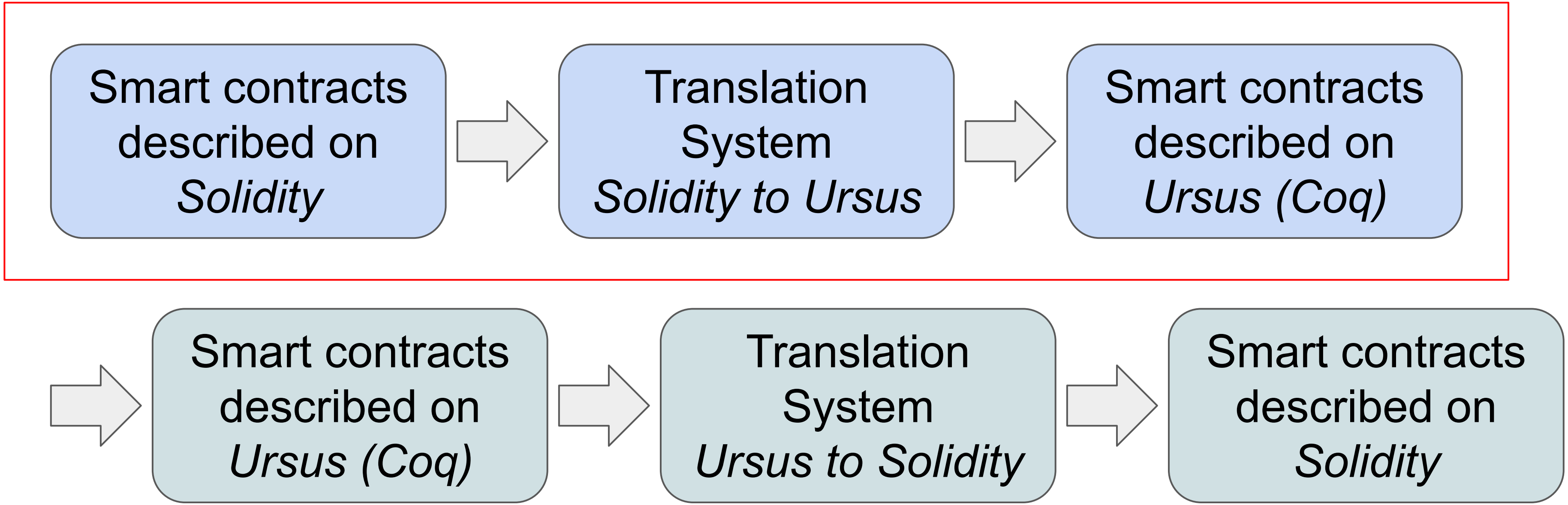


Smart contracts described on *Solidity*

Translation System *Solidity to Ursus*

Smart contracts described on *Ursus (Coq)*

Translation scheme in the full translation scheme



Smart contracts described on *Solidity*

Translation System *Solidity to Ursus*

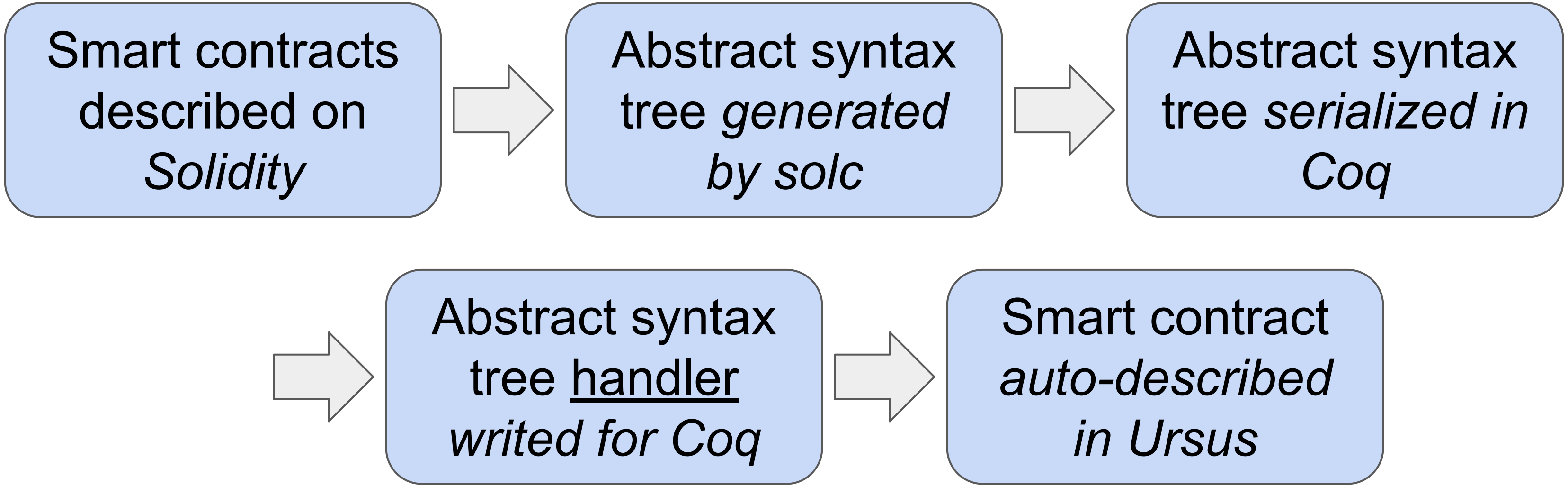
Smart contracts described on *Ursus (Coq)*

Smart contracts described on *Ursus (Coq)*

Translation System *Ursus to Solidity*

Smart contracts described on *Solidity*

Details of pypeline



Smart contracts described on *Solidity*

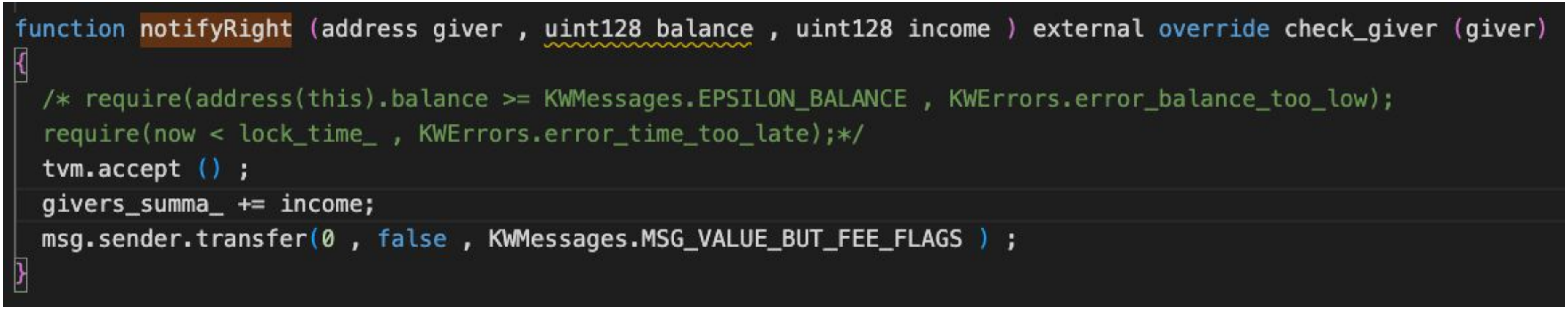
Abstract syntax tree *generated by solc*

Abstract syntax tree *serialized in Coq*

Abstract syntax tree handler *writed for Coq*

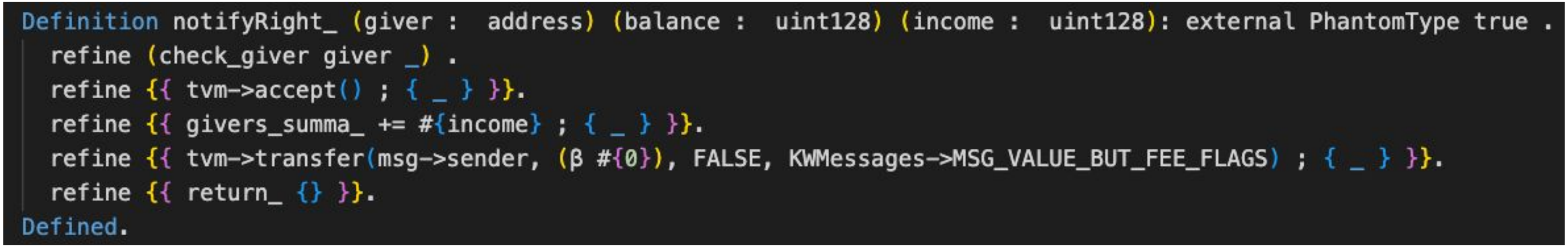
Smart contract *auto-described in Ursus*

Some examples *Solidity*



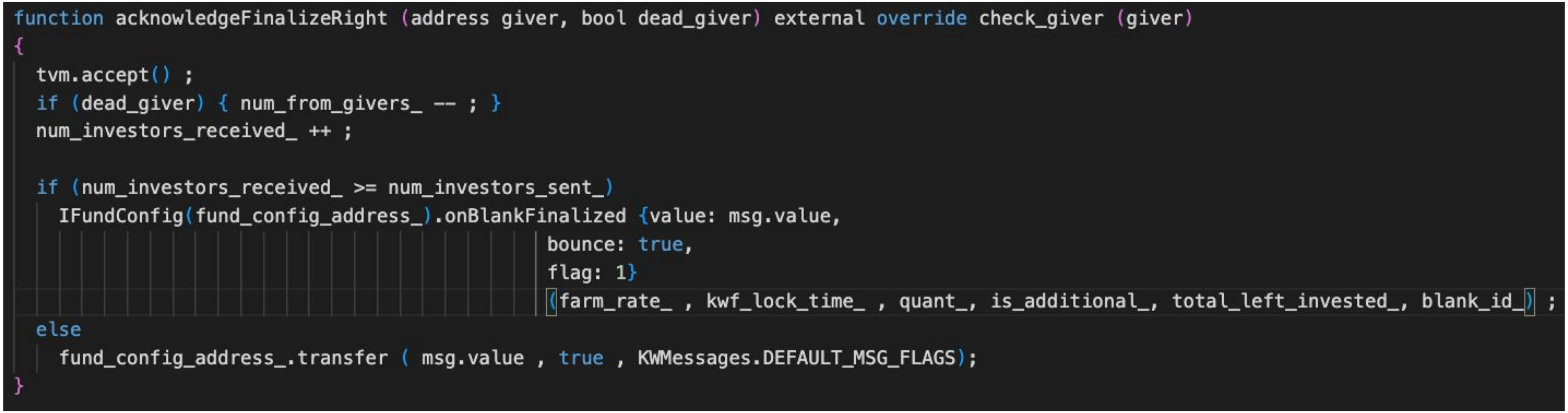
```
function notifyRight (address giver , uint128 balance , uint128 income ) external override check_giver (giver)
{
  /* require(address(this).balance >= KWMessages.EPSILON_BALANCE , KWErrors.error_balance_too_low);
  require(now < lock_time_ , KWErrors.error_time_too_late);*/
  tvm.accept () ;
  givers_summa_ += income;
  msg.sender.transfer(0 , false , KWMessages.MSG_VALUE_BUT_FEE_FLAGS ) ;
}
```

Some examples *Ursus*



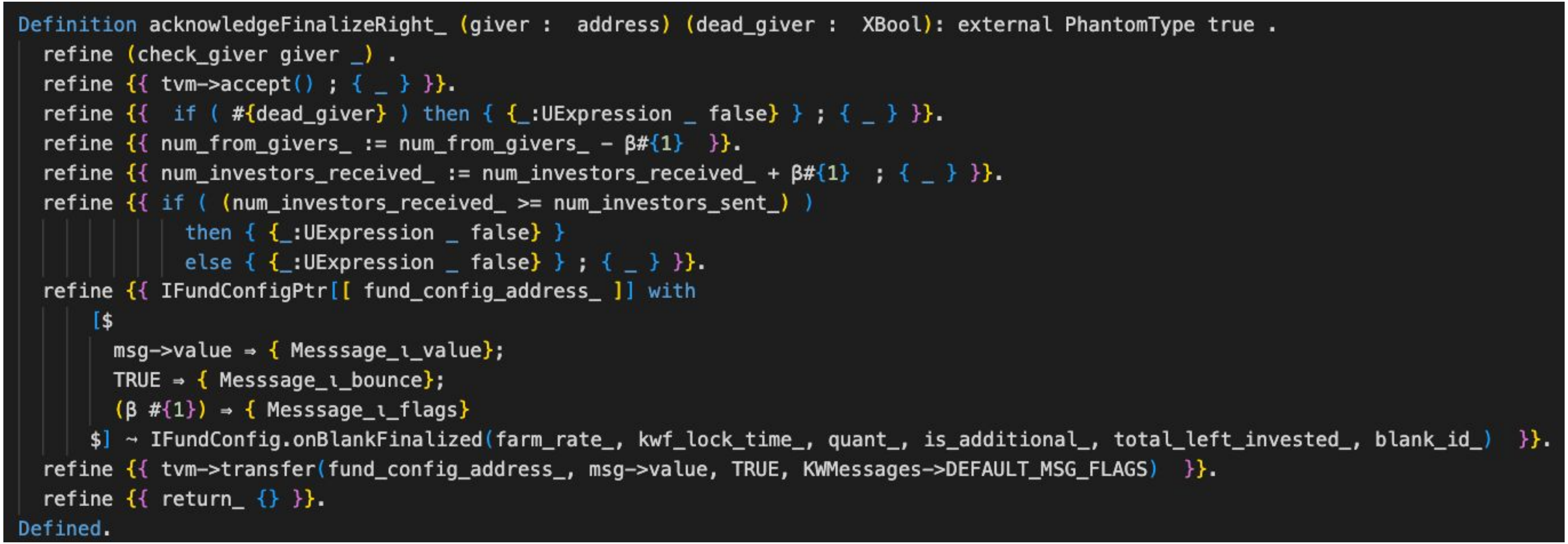
```
Definition notifyRight_ (giver :  address) (balance :  uint128) (income :  uint128): external PhantomType true .
  refine (check_giver giver _) .
  refine {{ tvm->accept() ; { _ } }}.
  refine {{ givers_summa_ += #{income} ; { _ } }}.
  refine {{ tvm->transfer(msg->sender, (β #{0}), FALSE, KWMessages->MSG_VALUE_BUT_FEE_FLAGS) ; { _ } }}.
  refine {{ return_ {} }}.
Defined.
```

Some examples *Solidity*



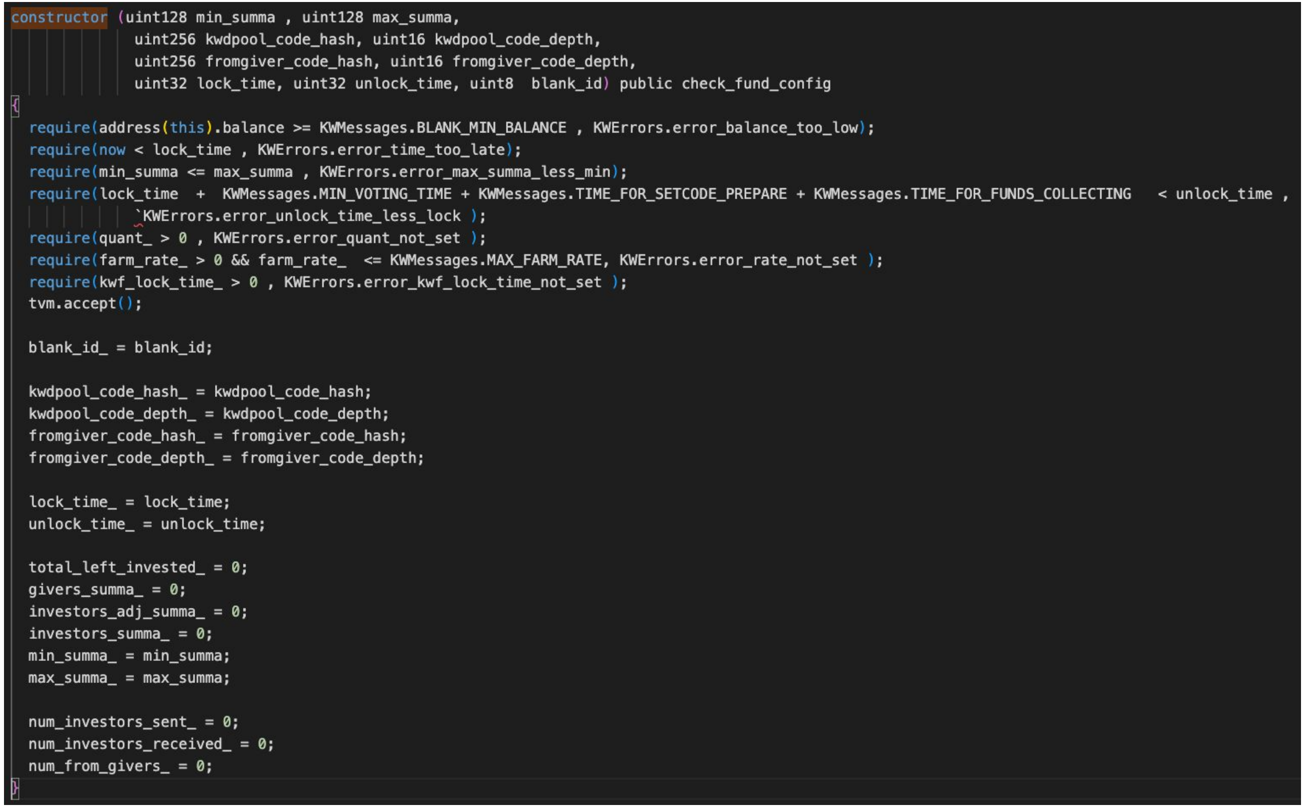
```
function acknowledgeFinalizeRight (address giver, bool dead_giver) external override check_giver (giver)
{
  tvm.accept() ;
  if (dead_giver) { num_from_givers_ -- ; }
  num_investors_received_ ++ ;

  if (num_investors_received_ >= num_investors_sent_)
    IFundConfig(fund_config_address_).onBlankFinalized {value: msg.value,
                                               bounce: true,
                                               flag: 1}
                                               (farm_rate_ , kwf_lock_time_ , quant_, is_additional_, total_left_invested_, blank_id_) ;
  else
    fund_config_address_.transfer ( msg.value , true , KWMessages.DEFAULT_MSG_FLAGS);
}
```

Some examples *Ursus*



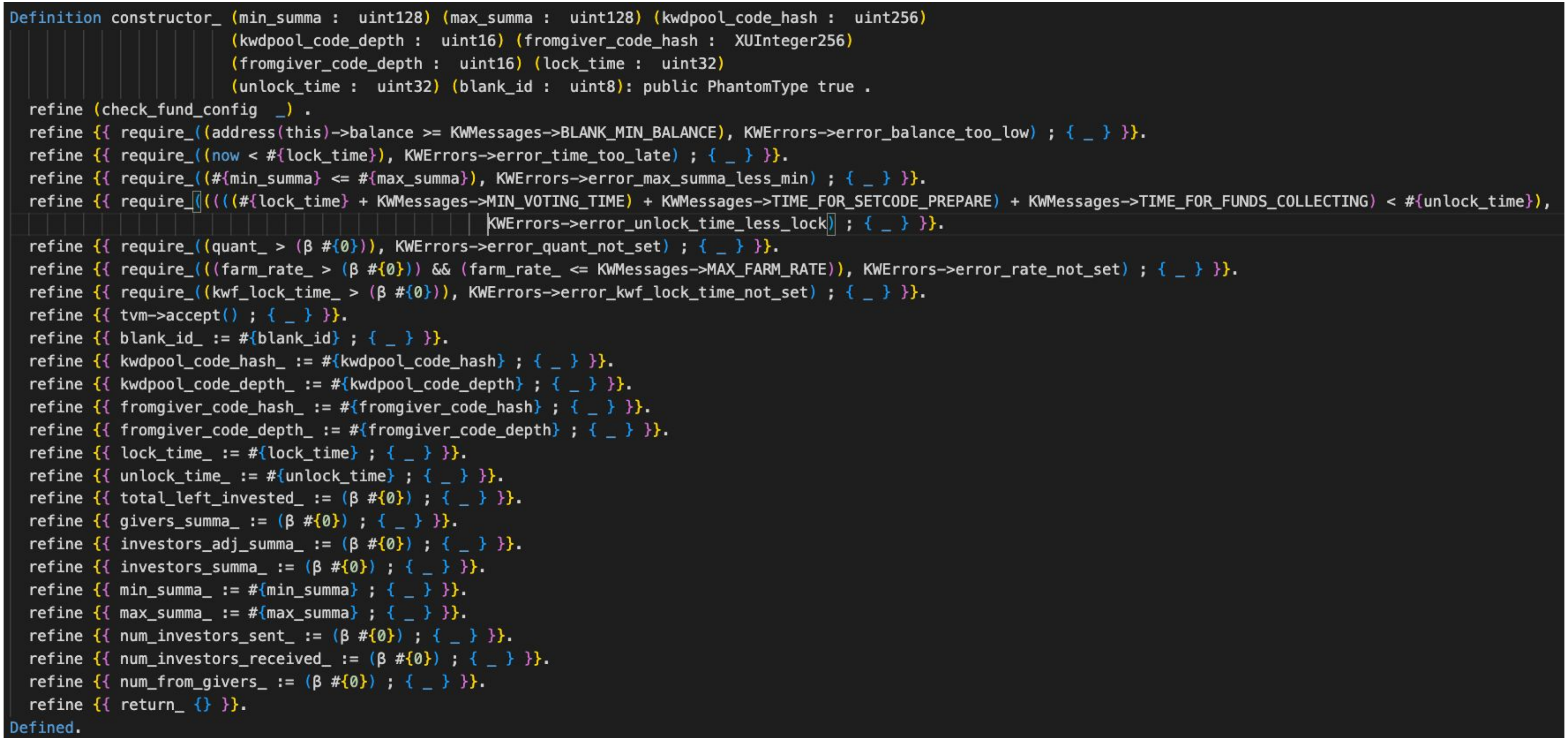
```
Definition acknowledgeFinalizeRight_ (giver :  address) (dead_giver :  XBool): external PhantomType true .
  refine (check_giver giver _) .
  refine {{ tvm->accept() ; { _ } }}.
  refine {{  if ( #{dead_giver} ) then { {_:UExpression _ false} } ; { _ } }}.
  refine {{ num_from_givers_ := num_from_givers_ - β#{1}  }}.
  refine {{ num_investors_received_ := num_investors_received_ + β#{1}  ; { _ } }}.
  refine {{ if ( (num_investors_received_ >= num_investors_sent_) )
              then { {_:UExpression _ false} }
              else { {_:UExpression _ false} } ; { _ } }}.
  refine {{ IFundConfigPtr[[ fund_config_address_ ]] with
      [$
        msg->value ⇒ { Messsage_ι_value};
        TRUE ⇒ { Messsage_ι_bounce};
        (β #{1}) ⇒ { Messsage_ι_flags}
      $] ⤳ IFundConfig.onBlankFinalized(farm_rate_, kwf_lock_time_, quant_, is_additional_, total_left_invested_, blank_id_)  }}.
  refine {{ tvm->transfer(fund_config_address_, msg->value, TRUE, KWMessages->DEFAULT_MSG_FLAGS)  }}.
  refine {{ return_ {} }}.
Defined.
```

Some examples *Solidity*



```
constructor (uint128 min_summa , uint128 max_summa,
              uint256 kwdpool_code_hash, uint16 kwdpool_code_depth,
              uint256 fromgiver_code_hash, uint16 fromgiver_code_depth,
              uint32 lock_time, uint32 unlock_time, uint8  blank_id) public check_fund_config
{
  require(address(this).balance >= KWMessages.BLANK_MIN_BALANCE , KWErrors.error_balance_too_low);
  require(now < lock_time , KWErrors.error_time_too_late);
  require(min_summa <= max_summa , KWErrors.error_max_summa_less_min);
  require(lock_time  +  KWMessages.MIN_VOTING_TIME + KWMessages.TIME_FOR_SETCODE_PREPARE + KWMessages.TIME_FOR_FUNDS_COLLECTING   < unlock_time ,
              `KWErrors.error_unlock_time_less_lock );
  require(quant_ > 0 , KWErrors.error_quant_not_set );
  require(farm_rate_ > 0 && farm_rate_  <= KWMessages.MAX_FARM_RATE, KWErrors.error_rate_not_set );
  require(kwf_lock_time_ > 0 , KWErrors.error_kwf_lock_time_not_set );
  tvm.accept();

  blank_id_ = blank_id;

  kwdpool_code_hash_ = kwdpool_code_hash;
  kwdpool_code_depth_ = kwdpool_code_depth;
  fromgiver_code_hash_ = fromgiver_code_hash;
  fromgiver_code_depth_ = fromgiver_code_depth;

  lock_time_ = lock_time;
  unlock_time_ = unlock_time;

  total_left_invested_ = 0;
  givers_summa_ = 0;
  investors_adj_summa_ = 0;
  investors_summa_ = 0;
  min_summa_ = min_summa;
  max_summa_ = max_summa;

  num_investors_sent_ = 0;
  num_investors_received_ = 0;
  num_from_givers_ = 0;
}
```

Some examples *Ursus*



```
Definition constructor_ (min_summa :  uint128) (max_summa :  uint128) (kwdpool_code_hash :  uint256)
                        (kwdpool_code_depth :  uint16) (fromgiver_code_hash :  XUInteger256)
                        (fromgiver_code_depth :  uint16) (lock_time :  uint32)
                        (unlock_time :  uint32) (blank_id :  uint8): public PhantomType true .
  refine (check_fund_config  _) .
  refine {{ require_((address(this)->balance >= KWMessages->BLANK_MIN_BALANCE), KWErrors->error_balance_too_low) ; { _ } }}.
  refine {{ require_((now < #{lock_time}), KWErrors->error_time_too_late) ; { _ } }}.
  refine {{ require_((#{min_summa} <= #{max_summa}), KWErrors->error_max_summa_less_min) ; { _ } }}.
  refine {{ require_((((#{lock_time} + KWMessages->MIN_VOTING_TIME) + KWMessages->TIME_FOR_SETCODE_PREPARE) + KWMessages->TIME_FOR_FUNDS_COLLECTING) < #{unlock_time}),
                                                    KWErrors->error_unlock_time_less_lock) ; { _ } }}.
  refine {{ require_((quant_ > (β #{0})), KWErrors->error_quant_not_set) ; { _ } }}.
  refine {{ require_(((farm_rate_ > (β #{0})) && (farm_rate_ <= KWMessages->MAX_FARM_RATE)), KWErrors->error_rate_not_set) ; { _ } }}.
  refine {{ require_((kwf_lock_time_ > (β #{0})), KWErrors->error_kwf_lock_time_not_set) ; { _ } }}.
  refine {{ tvm->accept() ; { _ } }}.
  refine {{ blank_id_ := #{blank_id} ; { _ } }}.
  refine {{ kwdpool_code_hash_ := #{kwdpool_code_hash} ; { _ } }}.
  refine {{ kwdpool_code_depth_ := #{kwdpool_code_depth} ; { _ } }}.
  refine {{ fromgiver_code_hash_ := #{fromgiver_code_hash} ; { _ } }}.
  refine {{ fromgiver_code_depth_ := #{fromgiver_code_depth} ; { _ } }}.
  refine {{ lock_time_ := #{lock_time} ; { _ } }}.
  refine {{ unlock_time_ := #{unlock_time} ; { _ } }}.
  refine {{ total_left_invested_ := (β #{0}) ; { _ } }}.
  refine {{ givers_summa_ := (β #{0}) ; { _ } }}.
  refine {{ investors_adj_summa_ := (β #{0}) ; { _ } }}.
  refine {{ investors_summa_ := (β #{0}) ; { _ } }}.
  refine {{ min_summa_ := #{min_summa} ; { _ } }}.
  refine {{ max_summa_ := #{max_summa} ; { _ } }}.
  refine {{ num_investors_sent_ := (β #{0}) ; { _ } }}.
  refine {{ num_investors_received_ := (β #{0}) ; { _ } }}.
  refine {{ num_from_givers_ := (β #{0}) ; { _ } }}.
  refine {{ return_ {} }}.
Defined.
```