A farmer with a wolf, a goat, and a cabbage must cross a river by boat. The boat can carry only the farmer and a single item. If left unattended together, the wolf would eat the goat, or the goat would eat the cabbage. How can they cross the river without anything being eaten? https://*en.m.wikipedia.org*/wiki/*Wolf*,*_goat_and_cabbage_problem*

– This spec tought me the difference between CHOICE and with: https://*groups.google.com*/g/tlaplus/c/*QJAulqYgC0E*/*m*/*Cm2PPJBDCQAJ*

> CHOOSE is definitely a confusing operator, many people have problems with it including myself.
> The idea is that CHOOSE always picks the same value, it does not represent arbitrary values.
> If you're trying to say "sub can be any element of *ClaimsData*", then the "there exists"
> operator (existential quantification) is what you're looking for, *i.e.*
> ∃ sub ∈ *ClaimsData*: *SomePredicate*(*sub*)

And: > What if multiple values satisfy CHOOSE ? In this case the only requirement is that the result is deterministic: the engine must always return the same value, no matter what. In practice this means that *TLC* will always choose the lowest value that matches the set. https://*learntla.com*/core/*operators.html* ≠ choose

CHOOSE has to be deterministic but in my previous implementation (in last commit) I was trying to use CHOOSE to pick any element in subset of side such that removing that item from side would have left it in a valid state. Because I was using CHOOSE , tlc was always picking the same value leading to a very small possible state transitions and a lot of headache to me.

```
──────────────── MODULE wolf_goat_cabbage ────────────────
LOCAL INSTANCE TLC
LOCAL INSTANCE FiniteSets
LOCAL INSTANCE Integers

WOLF ≜ "Wolf"
GOAT ≜ "Goat"
CABBAGE ≜ "Cabbage"
FinalResult ≜ {WOLF, GOAT, CABBAGE}
InvalidStates ≜ {{CABBAGE, GOAT}, {WOLF, GOAT}}
baitinv ≜ TRUE
 baitinv ≜ TLCGet("level") < 14

  --algorithm wolf_goat_cabbage{

variables side_start = FinalResult, side_end = {} ;

define {
    IsValidState(side) ≜ ¬(side ∈ InvalidStates)
    Inv ≜ side_end ≠ FinalResult
     this operator is used select an item
     from side such that removing this item
     will leave "side" in a valid state.
    ValidSubsets(side) ≜ {s ∈ SUBSET (side) : IsValidState(side_start \ s) ∧ Cardinality(s) = 1}
 }

macro PickFrom( side, other_side_is_valid ) {
    await ∧ transport = {}
          ∧ side ≠ {} ∧ other_side_is_valid ;
    choose an item such that this side is left valid:
    with ( $item \in ValidSubsets(side)$ ) {
        $transport := item$ ;
        $side := side \setminus transport$ ;
    }
}

macro $DropItemTo$( $side$ ) {
    leave an item to $side\_start$ side. If needed to avoid conflicts, load the other item.
    await $transport \neq \{\}$ ;
    $side := side \cup transport$ ;
    $transport := \{\}$ ;
}

process ( $Farmer = 1$ )
variable $transport = \{\}$ ; {
$W$:
    while ( TRUE ) {
        either {
            pick an item from $side\_start$ and load it.
            $PickFrom(side\_start, IsValidState(side\_end))$ ;
        } or {
            pick an item from $side\_start$ and load it.
            $PickFrom(side\_end, IsValidState(side\_start))$ ;
        } or {
            $DropItemTo(side\_start)$ ;
        } or {
            $DropItemTo(side\_end)$ ;
        } ;
    }
}
}
BEGIN TRANSLATION ($chksum(pcal) =$ "7c28162a" $\land chksum(tla) =$ "b19f62f8")
VARIABLES $side\_start, side\_end$

define statement
$IsValidState(side) \triangleq \neg(side \in InvalidStates)$
$Inv \triangleq side\_end \neq FinalResult$

$ValidSubsets(side) \triangleq \{s \in \text{SUBSET } (side) : IsValidState(side\_start \setminus s) \land Cardinality(s) = 1\}$

VARIABLE $transport$

$vars \triangleq \langle side\_start, side\_end, transport \rangle$

$ProcSet \triangleq \{1\}$
```

```
$Init \triangleq$  Global variables
          $\land side\_start = FinalResult$
          $\land side\_end = \{\}$
           Process $Farmer$
          $\land transport = \{\}$

$Farmer \triangleq$ $\lor \land \land transport = \{\}$
                  $\land side\_start \neq \{\} \land (IsValidState(side\_end))$
              $\land \exists item \in ValidSubsets(side\_start) :$
                  $\land transport' = item$
                  $\land side\_start' = side\_start \setminus transport'$
              $\land$ UNCHANGED $side\_end$
          $\lor \land \land transport = \{\}$
                  $\land side\_end \neq \{\} \land (IsValidState(side\_start))$
              $\land \exists item \in ValidSubsets(side\_end) :$
                  $\land transport' = item$
                  $\land side\_end' = side\_end \setminus transport'$
              $\land$ UNCHANGED $side\_start$
          $\lor \land transport \neq \{\}$
              $\land side\_start' = (side\_start \cup transport)$
              $\land transport' = \{\}$
              $\land$ UNCHANGED $side\_end$
          $\lor \land transport \neq \{\}$
              $\land side\_end' = (side\_end \cup transport)$
              $\land transport' = \{\}$
              $\land$ UNCHANGED $side\_start$

$Next \triangleq Farmer$

$Spec \triangleq \land Init \land \Box[Next]_{vars}$
         $\land \mathrm{WF}_{vars}(Next)$
         $\land \mathrm{WF}_{vars}(Farmer)$

 END TRANSLATION

$TypeOk \triangleq \land \forall el \in side\_start : el \in \{WOLF, GOAT, CABBAGE\}$
           $\land \forall el \in side\_end : el \in \{WOLF, GOAT, CABBAGE\}$
           $\land \forall el \in transport : el \in \{WOLF, GOAT, CABBAGE\}$
```