Υ^{z}, the status code of this transaction:

$$\Upsilon^{\mathrm{g}}(\boldsymbol{\sigma}, T) \equiv T_{\mathrm{g}} - g^* \tag{80}$$

$$\Upsilon^{\mathrm{l}}(\boldsymbol{\sigma}, T) \equiv A_{\mathrm{l}} \tag{81}$$

$$\Upsilon^{\mathrm{z}}(\boldsymbol{\sigma}, T) \equiv z \tag{82}$$

These are used to help define the transaction receipt and are also used later for state and nonce validation.

7. Contract Creation

There are a number of intrinsic parameters used when creating an account: sender (s), original transactor[4] (o), available gas (g), gas price (p), endowment (v) together with an arbitrary length byte array, $\mathbf{i}$, the initialisation EVM code, the present depth of the message-call/contract-creation stack (e), the salt for new account's address (ζ) and finally the permission to make modifications to the state (w). The salt ζ might be missing ($\zeta = \varnothing$); formally,

$$\zeta \in \mathbb{B}_{32} \cup \mathbb{B}_0 \tag{83}$$

If the creation was caused by CREATE2, then $\zeta \neq \varnothing$.

We define the creation function formally as the function Λ, which evaluates from these values, together with the state $\boldsymbol{\sigma}$ and the accrued substate A, to the tuple containing the new state, remaining gas, new accrued substate, status code and output $(\boldsymbol{\sigma}', g', A', z, \mathbf{o})$:

$$(\boldsymbol{\sigma}', g', A', z, \mathbf{o}) \equiv \Lambda(\boldsymbol{\sigma}, A, s, o, g, p, v, \mathbf{i}, e, \zeta, w) \tag{84}$$

The address of the new account is defined as being the rightmost 160 bits of the Keccak-256 hash of the RLP encoding of the structure containing only the sender and the account nonce. For CREATE2 the rule is different and is described in EIP-1014 by Buterin [2018]. Combining the two cases, we define the resultant address for the new account a:

$$a \equiv \mathtt{ADDR}(s, \boldsymbol{\sigma}[s]_{\mathrm{n}} - 1, \zeta, \mathbf{i}) \tag{85}$$

$$\mathtt{ADDR}(s, n, \zeta, \mathbf{i}) \equiv \mathcal{B}_{96..255}\Big(\mathtt{KEC}\big(L_{\mathrm{A}}(s, n, \zeta, \mathbf{i})\big)\Big) \tag{86}$$

$$L_{\mathrm{A}}(s, n, \zeta, \mathbf{i}) \equiv \begin{cases} \mathtt{RLP}\big(\ (s, n)\ \big) & \text{if } \zeta = \varnothing \\ (255) \cdot s \cdot \zeta \cdot \mathtt{KEC}(\mathbf{i}) & \text{otherwise} \end{cases} \tag{87}$$

where $\cdot$ is the concatenation of byte arrays, $\mathcal{B}_{a..b}(X)$ evaluates to a binary value containing the bits of indices in the range $[a, b]$ of the binary data X, and $\boldsymbol{\sigma}[x]$ is the address state of x, or $\varnothing$ if none exists. Note we use one fewer than the sender's nonce value; we assert that we have incremented the sender account's nonce prior to this call, and so the value used is the sender's nonce at the beginning of the responsible transaction or VM operation.

The address of the new account is added to the set of accessed accounts:

$$A^* \equiv A \quad \text{except} \quad A^*_{\mathbf{a}} \equiv A_{\mathbf{a}} \cup \{a\} \tag{88}$$

The account's nonce is initially defined as one, the balance as the value passed, the storage as empty and the code hash as the Keccak 256-bit hash of the empty string; the sender's balance is also reduced by the value passed. Thus the mutated state becomes $\boldsymbol{\sigma}^*$:

$$\boldsymbol{\sigma}^* \equiv \boldsymbol{\sigma} \quad \text{except:} \tag{89}$$

$$\boldsymbol{\sigma}^*[a] = \big(1, v + v', \mathtt{TRIE}(\varnothing), \mathtt{KEC}(())\big) \tag{90}$$

$$\boldsymbol{\sigma}^*[s] = \begin{cases} \varnothing & \text{if } \boldsymbol{\sigma}[s] = \varnothing \ \wedge \ v = 0 \\ \mathbf{a}^* & \text{otherwise} \end{cases} \tag{91}$$

$$\mathbf{a}^* \equiv (\boldsymbol{\sigma}[s]_{\mathrm{n}}, \boldsymbol{\sigma}[s]_{\mathrm{b}} - v, \boldsymbol{\sigma}[s]_{\mathbf{s}}, \boldsymbol{\sigma}[s]_{\mathrm{c}}) \tag{92}$$

where v' is the account's pre-existing value, in the event it was previously in existence:

$$v' \equiv \begin{cases} 0 & \text{if } \ \boldsymbol{\sigma}[a] = \varnothing \\ \boldsymbol{\sigma}[a]_{\mathrm{b}} & \text{otherwise} \end{cases} \tag{93}$$

Finally, the account is initialised through the execution of the initialising EVM code $\mathbf{i}$ according to the execution model (see section 9). Code execution can effect several events that are not internal to the execution state: the account's storage can be altered, further accounts can be created and further message calls can be made. As such, the code execution function Ξ evaluates to a tuple of the resultant state $\boldsymbol{\sigma}^{**}$, available gas remaining g^{**}, the resultant accrued substate A^{**} and the body code of the account $\mathbf{o}$.

$$(\boldsymbol{\sigma}^{**}, g^{**}, A^{**}, \mathbf{o}) \equiv \Xi(\boldsymbol{\sigma}^*, g, A^*, I) \tag{94}$$

where I contains the parameters of the execution environment, that is:

$$I_{\mathrm{a}} \equiv a \tag{95}$$

$$I_{\mathrm{o}} \equiv o \tag{96}$$

$$I_{\mathrm{p}} \equiv p \tag{97}$$

$$I_{\mathbf{d}} \equiv () \tag{98}$$

$$I_{\mathrm{s}} \equiv s \tag{99}$$

$$I_{\mathrm{v}} \equiv v \tag{100}$$

$$I_{\mathbf{b}} \equiv \mathbf{i} \tag{101}$$

$$I_{\mathrm{e}} \equiv e \tag{102}$$

$$I_{\mathrm{w}} \equiv w \tag{103}$$

$I_{\mathbf{d}}$ evaluates to the empty tuple as there is no input data to this call. I_{H} has no special treatment and is determined from the blockchain.

Code execution depletes gas, and gas may not go below zero, thus execution may exit before the code has come to a natural halting state. In this (and several other) exceptional cases we say an out-of-gas (OOG) exception has occurred: The evaluated state is defined as being the empty set, $\varnothing$, and the entire create operation should have no effect on the state, effectively leaving it as it was immediately prior to attempting the creation.

If the initialization code completes successfully, a final contract-creation cost is paid, the code-deposit cost, c, proportional to the size of the created contract's code:

$$c \equiv G_{\mathrm{codedeposit}} \times \|\mathbf{o}\| \tag{104}$$

If there is not enough gas remaining to pay this, i.e. $g^{**} < c$, then we also declare an out-of-gas exception.

The gas remaining will be zero in any such exceptional condition, i.e. if the creation was conducted as the reception of a transaction, then this doesn't affect payment of the intrinsic cost of contract creation; it is paid regardless. However, the value of the transaction is not transferred

[4] which can differ from the sender in the case of a message call or contract creation not directly triggered by a transaction but coming from the execution of EVM-code