1. Conventions

Item	**Convention**	**Examples**
Top level structures	Lower case bold Greek	$\boldsymbol{\sigma}$, the world state $\boldsymbol{\mu}$, the machine state.
Functions on highly structured values	Upper case Greek	Υ, the Ethereum state transition function.
Most functions	Upper case letters, possibly subscripted	C, the general cost function C_{SSTORE}, the cost function for the SSTORE operation.
Specialised functions	Typewriter	`KEC`, the Keccak-256 hash `KEC512`, the Keccak-512 hash function.
Tuple	Upper case letter	T, a transaction.
Component of a Tuple	Subscripted upper-case letter. A capital subscript refers to a component that is a tuple.	T_n, the transaction nonce I_H, The header of the current block (a tuple).
Scalars, fixed size byte sequences/arrays	Usually a lower-case letter Sometimes Greek	n, a transaction's nonce δ, the number of stack items required.
Arbitrary length sequences	Bold lower-case	$\mathbf{o}$, output data of message call.
Sets	Double struck capitals	$\mathbb{P}_{256}$, positive integers less than 2^{256} $\mathbb{B}_{32}$, byte sequences of length 32.
Components or subsequences of sequences	Square brackets	$\boldsymbol{\mu}_{\mathbf{s}}[0]$, the first item on the stack $\boldsymbol{\mu}_{\mathbf{m}}[0..31]$ the first 32 items in memory.
Modified (and utilisable) value	Prime mark	g' gas remaining Exception: s' is just a scalar! (Transaction status code.)
Intermediate values	Asterisk superscripts	g^* gas to be refunded g^{**} available gas remaining after code execution.
Element-wise transformations	Asterisk superscript on a function	$f^*\big((x_0, x_1, ...)\big) \equiv \big(f(x_0), f(x_1), ...\big)$ for any function f.

2. Symbols

Name	Description
High level constructs	
$\boldsymbol{\sigma}$	The world-state, comprising all accounts' nonces, balances, storage and code.
$\boldsymbol{\sigma}_t$	World-state at time t.
$\boldsymbol{\mu}$	Machine-state tuple, $(g, pc, \mathbf{m}, i, \mathbf{s})$, which are gas, program counter, memory, memory size, stack.
T	An Ethereum transaction
T_0, T_1, ...	Individual transactions within a block
B	A block: $B \equiv (..., (T_0, T_1, ...))$
Υ	The Ethereum state transition function: $\boldsymbol{\sigma}_{t+1} \equiv \Upsilon(\boldsymbol{\sigma}_t, T)$
Ω	The block-finalisation state transition function (pays out the mining reward).
Π	The block-level state-accumulation function: $\Pi(\boldsymbol{\sigma}, B) \equiv \Omega(B, \Upsilon(\Upsilon(\boldsymbol{\sigma}, T_0), T_1)...)$
World state	
$\boldsymbol{\sigma}[a]$	The account state of account a, being a tuple of (nonce, balance, storageRoot, codeHash).
$\boldsymbol{\sigma}[a]_n$	The nonce of account a.
$\boldsymbol{\sigma}[a]_b$	The balance of account a.
$\boldsymbol{\sigma}[a]_s$	A 256-bit hash of the root node of a Merkle Patricia tree that encodes the storage contents of account a. Note that $\texttt{TRIE}\big(L_I^*(\boldsymbol{\sigma}[a]_{\mathbf{s}})\big) \equiv \boldsymbol{\sigma}[a]_s$

Name	Description
$\boldsymbol{\sigma}[a]_c$	The hash of the EVM code of account a. Equal to $\texttt{KEC}(\mathbf{b})$ where $\mathbf{b}$ is the account's code.

Machine state

$\boldsymbol{\mu}_g$	The gas available.
$\boldsymbol{\mu}_{pc}$	The program counter.
$\boldsymbol{\mu}_{\mathbf{m}}$	The memory contents.
$\boldsymbol{\mu}_i$	The number of memory words allocated.
$\boldsymbol{\mu}_{\mathbf{s}}$	The stack.
$\boldsymbol{\mu}_{\mathbf{s}}[n]$	Item at stack depth n.

Substate

A	A Transaction substate during execution: $\equiv (A_{\mathbf{s}}, A_{\mathbf{l}}, A_r) \equiv A \equiv (\mathbf{s}, \mathbf{l}, r)$.
$A_{\mathbf{s}}$	The self-destruct set.
$A_{\mathbf{l}}$	The log series.
A_r	The gas refund balance. Can partially offset execution costs.
A^0	The empty substate: $A^0 \equiv (\varnothing, (), 0)$.

Execution environment

I	Tuple of the following items provided to the execution environment.
I_a	The address of the account which owns the code that is executing.
I_o	The sender address of the transaction that originated this execution.
I_p	The price of gas in the transaction that originated this execution.
$I_{\mathbf{d}}$	The byte array that is the input data to this execution; if the execution agent is a transaction, this would be the transaction data.
I_s	The address of the account which caused the code to be executing; if the execution agent is a transaction, this would be the transaction sender.
I_v	The value, in Wei, passed to this account as part of the same procedure as execution; if the execution agent is a transaction, this would be the transaction value.
$I_{\mathbf{b}}$	The byte array that is the machine code to be executed.
I_H	The block header of the present block.
I_e	The depth of the present message-call or contract-creation (i.e. the number of CALLs or CREATEs being executed at present).
I_w	Flag for permission to make modifications to the state. See EIP-214, STATICCALL

Execution

Ξ	The code execution function $(\boldsymbol{\sigma}', g', A, \mathbf{o}) \equiv \Xi(\boldsymbol{\sigma}, g, I)$.
$\mathbf{o}$	The output data of a message call, $\mathbf{o} \equiv H(\boldsymbol{\mu}, I)$. At contract creation, the contract bytecode to be deployed.
$H(\boldsymbol{\mu}, I)$	The normal halting function, usually the value provided by the RETURN or REVERT opcodes, or empty in the case of STOP.
$Z(\boldsymbol{\sigma}, \boldsymbol{\mu}, I)$	The exceptional halting function.
w	The current operation to be executed: $w \equiv I_{\mathbf{b}}[\boldsymbol{\mu}_{pc}]$ if $\boldsymbol{\mu}_{pc} < \lVert I_{\mathbf{b}} \rVert$, and STOP otherwise.

Blocks

B	A block: $B \equiv (B_H, B_{\mathbf{T}}, B_{\mathbf{U}})$.
B_H	The block's header.
$B_{\mathbf{T}}$	The block's transactions.
$B_{\mathbf{U}}$	Headers of ommer/uncle blocks of this block.
$B_{\mathbf{R}}$	Transaction receipts.
$D(H)$	The difficulty of the block with header H.
$P(H)$	The parent block of the block with header H.
$V(H)$	The block header validity function.

Name	Description
Block header	
H_p	**parentHash**: The Keccak 256-bit hash of the parent block's header, in its entirety.
H_o	**ommersHash** The Keccak 256-bit hash of the ommers list portion of this block.
H_c	**beneficiary** The 160-bit address to which all fees collected from the successful mining of this block be transferred.
H_r	**stateRoot** The Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied.
H_t	**transactionsRoot** The Keccak 256-bit hash of the root node of the trie structure populated with each transaction in the transactions list portion of the block.
H_e	**receiptsRoot** The Keccak 256-bit hash of the root node of the trie structure populated with the receipts of each transaction in the transactions list portion of the block.
H_b	**logsBloom** The Bloom filter composed from indexable information (logger address and log topics) contained in each log entry from the receipt of each transaction in the transactions list.
H_d	**difficulty** A scalar value corresponding to the difficulty level of this block.
H_i	**number** A scalar value equal to the number of ancestor blocks. The genesis block has a number of zero.
H_l	**gasLimit** A scalar value equal to the current limit of gas expenditure per block.
H_g	**gasUsed** A scalar value equal to the total gas used in transactions in this block.
H_s	**timestamp** A scalar value equal to the reasonable output of Unix's time() at this block's inception.
H_x	**extraData** An arbitrary byte array containing data relevant to this block. This must be 32 bytes or fewer.
H_m	**mixHash** A 256-bit hash which proves combined with the nonce that a sufficient amount of computation has been carried out on this block.
H_n	**nonce** A 64-bit hash which proves combined with the mix-hash that a sufficient amount of computation has been carried out on this block.

Transactions

T_n	Transaction nonce.
T_p	Gas price for the transaction.
T_g	The maximum gas for a transaction.
T_t	The "to" address for the transaction.
T_v	The value to be transferred by the transaction.
T_w, T_r, T_s	The v, r, s values of the transaction signature.
$T_{\mathbf{i}}$	EVM-code for account initialisation (i.e. contract deployment).
$T_{\mathbf{d}}$	Input data of a message call.
$S(T)$	Sender function—recovers the sender address from the transaction: $S(T) \equiv \mathcal{B}_{96..255}\big(\mathtt{KEC}\big(\mathtt{ECDSARECOVER}(h(T), T_w, T_r, T_s)\big)\big)$.

Transaction Receipt

R	A transaction receipt: $R \equiv (R_u, R_b, R_{\mathbf{l}}, R_{s'})$
R_u	The cumulative gas used so far in the block.
R_b	The bloom filter composed from the information in the transaction logs.
$R_{\mathbf{l}}$	The log entries created by the transaction, $(O_0, O_1, ...)$.
$R_{s'}$	The status code of the transaction.
O	A log entry: $O \equiv (O_a, (O_{\mathbf{t}0}, O_{\mathbf{t}1}, ...), O_{\mathbf{d}})$.
O_a	The logger's address.
$O_{\mathbf{t}}$	A 32-byte log topic.
$O_{\mathbf{d}}$	The log data for this entry.
Υ^g	The total gas used in this transaction.
$\Upsilon^{\mathbf{l}}$	The logs created by this transaction.
Υ^s	The status code of this transaction, s'.

Miscellaneous functions

$\ell(\mathbf{x})$	The last item in sequence $\mathbf{x}$: $\ell(\mathbf{x}) \equiv \mathbf{x}[\lVert\mathbf{x}\rVert - 1]$
$L(n)$	The "all but one 64th" function: $L(n) \equiv n - \lfloor n/64 \rfloor$.
$L_I\big((k,v)\big)$	Representation of key–value pairs in the trie: $L_I\big((k,v)\big) \equiv \big(\mathtt{KEC}(k), \mathtt{RLP}(v)\big)$
L_R	TODO

Name	Description
L_S	World-state collapse function. TODO: expand. Seems to have a different function in computing the message hash.
L_T	TODO
$M(s, f, l)$	Memory expansion function. s is the current top of memory; f is the start of writing; l is the number of bytes to be written.
$\mathcal{B}$	Bit reference function such that $\mathcal{B}_j(\mathbf{x})$ equals the bit of index j (indexed from 0) in the byte array $\mathbf{x}$
$\mathtt{EMPTY}(\boldsymbol{\sigma}, a)$	An account a is *empty* when it has no code, zero nonce and zero balance, $\boldsymbol{\sigma}[a]_c = \mathtt{KEC}\big(()\big) \wedge \boldsymbol{\sigma}[a]_n = 0 \wedge \boldsymbol{\sigma}[a]_b = 0$.
$\mathtt{DEAD}(\boldsymbol{\sigma}, a)$	An account a is *dead* when its account state is non-existent or empty: $\varnothing \vee \mathtt{EMPTY}(\boldsymbol{\sigma}, a)$.
TRIE	The root hash of the Merkle Patricia tree constructed from its arguments.
KEC	TODO
RLP	TODO
PoW	TODO

Operators and symbols

$\|...\|$, $\|...\|$	Length of a sequence. These seem to be used interchangeably, but I may have missed something.
$\wedge$	Logical “And”.
$\vee$	Logical “Or”.
$\varnothing$	The empty set.
$\cdot$	Concatenation, $(a, b, c, d) \cdot e \equiv (a, b, c, d, e)$, or scalar multiplication depending on context.

Todo

$\mathbb{B}$	The set of all sequences of bytes.
$\mathbb{B}_n$	The set of all byte sequences of length n bytes: $\mathbb{B}_n = \{B : B \in \mathbb{B} \wedge \Vert B \Vert = n\}$
$\mathbb{P}$	The set of positive integers [what’s wrong with $\mathbb{N}$??? Grrr...].
$\mathbb{P}_n$	The set of all positive integers smaller than 2^n: $\mathbb{P}_n = \{P : P \in \mathbb{P} \wedge P < 2^n\}$
$M_{3:2048}$	Specialised Bloom filter.
$\Lambda(...)$	Contract creation function.
$\Theta(...)$	“Message call”/contract execution function? Not very clearly defined anywhere, but used extensively.
$\Gamma(B)$	The “initiation state” of block B. Usually $\boldsymbol{\sigma}_i : \mathtt{TRIE}(L_S(\boldsymbol{\sigma}_i)) = P(B_H)_{H_r}$.
$\Psi(B)$	A block transition function that maps an incomplete block B to a complete block B' (adds in mixHash, nonce, stateRoot).
$r(...)$	Calculates stateRoot? Used once but not defined.
etc.	