

1. CONVENTIONS

Item	Convention	Examples
Top level structures	Lower case bold Greek	σ , the world state μ , 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	$\mu_{\mathbf{s}}[0]$, the first item on the stack $\mu_{\mathbf{m}}[0..31]$ the first 32 items in memory.
Modified (and utilisable) value	Prime mark	g' gas remaining.
Intermediate values	Asterisk superscripts	g^* gas to be refunded g^{**} available gas remaining after code execution.
Element-wise transformations	Asterisk superscript on a function	$f^*((x_0, x_1, \dots)) \equiv (f(x_0), f(x_1), \dots)$ for any function f .

2. SYMBOLS

Name	Description
High level constructs	
σ	The world-state, comprising all accounts' nonces, balances, storage and code.
σ_t	World-state at time t .
μ	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, \dots	Individual transactions within a block
B	A block: $B \equiv (\dots, (T_0, T_1, \dots))$
Υ	The Ethereum state transition function: $\sigma_{t+1} \equiv \Upsilon(\sigma_t, T)$
Ω	The block-finalisation state transition function (pays out the mining reward).
Π	The block-level state-accumulation function: $\Pi(\sigma, B) \equiv \Omega(B, \Upsilon(\Upsilon(\sigma, T_0), T_1) \dots)$
World state	
$\sigma[a]$	The account state of account a , being a tuple of (nonce, balance, storageRoot, codeHash).
$\sigma[a]_n$	The nonce of account a .
$\sigma[a]_b$	The balance of account a .
$\sigma[a]_s$	A 256-bit hash of the root node of a Merkle Patricia tree that encodes the storage contents of account a . Note that $\text{TRIE}(L_I^*(\sigma[a]_s)) \equiv \sigma[a]_s$
$\sigma[a]_c$	The hash of the EVM code of account a . Equal to $\text{KEC}(\mathbf{b})$ where \mathbf{b} is the account's code.

Name	Description
Machine state	
μ_g	The gas available.
μ_{pc}	The program counter.
μ_m	The memory contents.
μ_i	The number of memory words allocated.
μ_s	The stack.
$\mu_s[n]$	Item at stack depth n .
Substate	
A	A Transaction substate during execution: $A \equiv (A_s, A_l, A_t, A_r) \equiv (\mathbf{s}, \mathbf{l}, \mathbf{t}, r)$.
A_s	The self-destruct set. These accounts will be discarded following the transaction's completion.
A_l	The log series.
A_t	The set of touched accounts. Empty ones are deleted at the end of the transaction.
A_r	The gas refund balance. Can partially offset execution costs.
A^0	The empty substate: $A^0 \equiv (\emptyset, (), \emptyset, 0)$.
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_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_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 $(\sigma', g', A, \mathbf{o}) \equiv \Xi(\sigma, g, I)$.
\mathbf{o}	The output data of a message call, $\mathbf{o} \equiv H(\mu, I)$. At contract creation, the contract bytecode to be deployed.
$H(\mu, I)$	The normal halting function, usually the value provided by the RETURN or REVERT opcodes, or empty in the case of STOP.
$Z(\sigma, \mu, I)$	The exceptional halting function.
w	The current operation to be executed: $w \equiv I_b[\mu_{pc}]$ if $\mu_{pc} < \ I_b\ $, and STOP otherwise.
Blocks	
B	A block: $B \equiv (B_H, B_T, B_U)$.
B_H	The block's header.
B_T	The block's transactions.
B_U	Headers of ommer/uncle blocks of this block.
B_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_i	EVM-code for account initialisation (i.e. contract deployment).
T_d	Input data of a message call.
$S(T)$	Sender function—recovers the sender address from the transaction: $S(T) \equiv \mathcal{B}_{96..255}(\text{KEC}(\text{ECDSARECOVER}(h(T), T_w, T_r, T_s)))$.

Transaction Receipt

R	A transaction receipt: $R \equiv (R_z, R_u, R_b, R_l)$
R_z	The status code of the transaction.
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_l	The log entries created by the transaction, (O_0, O_1, \dots) .
O	A log entry: $O \equiv (O_a, (O_{t0}, O_{t1}, \dots), O_d)$.
O_a	The logger's address.
O_t	A 32-byte log topic.
O_d	The log data for this entry.
Υ^g	The total gas used in this transaction.
Υ^l	The logs created by this transaction.
Υ^z	The status code of this transaction, z .

Miscellaneous functions

$\ell(\mathbf{x})$	The last item in sequence \mathbf{x} : $\ell(\mathbf{x}) \equiv \mathbf{x}[\ \mathbf{x}\ - 1]$
$L(n)$	The "all but one 64th" function: $L(n) \equiv n - \lfloor n/64 \rfloor$.
$L_I((k, v))$	Representation of key-value pairs in the trie: $L_I((k, v)) \equiv (\text{KEC}(k), \text{RLP}(v))$
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}
$\text{EMPTY}(\sigma, a)$	An account a is <i>empty</i> when it has no code, zero nonce and zero balance, $\sigma[a]_c = \text{KEC}()$ $\wedge \sigma[a]_n = 0 \wedge \sigma[a]_b = 0$.
$\text{DEAD}(\sigma, a)$	An account a is <i>dead</i> when its account state is non-existent or empty: $\emptyset \vee \text{EMPTY}(\sigma, a)$.
TRIE	The root hash of the Merkle Patricia tree constructed from its arguments.
KEC	TODO
RLP	TODO
PoW	TODO

Operators and symbols

$\ ...\ , \dots $	Length of a sequence. These seem to be used interchangeably, but I may have missed something.
\wedge	Logical “And”.
\vee	Logical “Or”.
\emptyset	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: $\mathbb{B}_n = \{B : B \in \mathbb{B} \wedge \ B\ = n\}$
\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.
$\Lambda(\dots)$	Contract creation function.
$\Theta(\dots)$	“Message call”/contract execution function? Not very clearly defined anywhere, but used extensively.
$\Gamma(B)$	The “initiation state” of block B . Usually $\sigma_i : \text{TRIE}(L_S(\sigma_i)) = P(B_H)_{H_r}$.
$\Psi(B)$	A block transition function that maps an incomplete block B to a complete block B' (adds in mixHash, nonce, stateRoot).
$r(\dots)$	Calculates stateRoot? Used once but not defined.
<i>etc.</i>	
