Dont Run on Fumes - Parametric Gas Bounds for Smart Contracts

NOTA IMPORTANTE: El artículo se encuentra aceptado pendiente de publicación. En el momento en el artículo esté publicado, toda la documentación acerca del artículo estará disponible en la URL:

`http://costa.ls.fi.upm.es/groman/aneca/dontrunonfumes.pdf`

- Journal of Systems and Software (JSS).
  `https://www.journals.elsevier.com/journal-of-systems-and-software`
- Artículo:
- Justificación Información Artículo:
  - DOI: `https://doi.org/10.1016/j.jss.2021.110923`
  - Primera y última página del artículo
- Justificación Índice Impacto:
  - Copia de la información Índice JCR 2019

Submissions with an Editorial Office Decision for Author Guillermo Román-Díez

Page: 1 of 1 (2 total completed submissions) Display 10 results per page.

Action	Manuscript Number	Title	Authorship	Initial Date Submitted	Status Date	Current Status	Date Final Disposition Set	Final Disposition
View Submission View Decision Letter Send E-mail	JSS-D-18-00296	Peak Resource Analysis of Concurrent Distributed Systems	Corresponding Author	Apr 04, 2018	Nov 13, 2018	Completed - Accept	Nov 13, 2018	Accept
View Submission	JSS-D-20-00396	Don't Run on Fumes --- Parametric Gas Bounds for Smart Contracts	Other Author	May 19, 2020	Feb 04, 2021	Completed - Accept	Feb 04, 2021	Accept

Page: 1 of 1 (2 total completed submissions) Display 10 results per page.

<< Author Main Menu

The Journal of Systems & Software 176 (2021) 110923

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Don't run on fumes—Parametric gas bounds for smart contracts☆

Elvira Albert [a,b], Jesús Correas [b], Pablo Gordillo [b,*], Guillermo Román-Díez [c], Albert Rubio [a,b]

[a] *Instituto de Tecnología del Conocimiento, Spain*
[b] *Complutense University of Madrid, Spain*
[c] *Universidad Politécnica de Madrid, Spain*

ARTICLE INFO

Article history:
Received 19 May 2020
Received in revised form 7 October 2020
Accepted 1 February 2021
Available online 26 February 2021

Keywords:
Smart contracts
Resource analysis
Static analysis
Decompilation

ABSTRACT

Gas is a measurement unit of the computational effort that it will take to execute every single replicated operation that takes part in the Ethereum blockchain platform. If a transaction exceeds the amount of gas allotted by the user (known as gas limit), an *out-of-gas* exception is raised and its execution is interrupted. One of the main open problems in the analysis of Ethereum smart contracts is the inference of *sound* bounds on their gas consumption.

We present, to the best of our knowledge, the first static analysis that is able to infer sound *parametric* (i.e., non-constant) gas bounds for smart contracts. The inferred bounds can be parametric on the sizes of the input parameters for the functions, but also they can be parametric on the contract state, or blockchain data. Our gas analysis is developed at EVM bytecode level, in which Ethereum gas model is defined.

Our analysis is implemented in a tool named GASTAP, Gas-Aware Smart contracT Analysis Platform, which takes as input a smart contract and automatically infers *sound* gas upper-bounds for its public functions. GASTAP has been applied over 318,093 functions fetched from the Ethereum blockchain, and succeeded to obtain gas bounds for 90.24% of them.

1. Introduction

In the Ethereum consensus protocol, every operation on a replicated blockchain state, which can be performed in a transactional manner by executing a *smart contract* code, costs a certain amount of *gas* (Wood, 2014). Gas has a monetary value in *Ether*, Ethereum's currency, and it is paid by a transaction-proposing party. Computations (initiated by a protocol client invoking a smart contract) that require *more computational or storage resources*, cost more gas than those that require fewer resources. As regards storage, the Ethereum Virtual Machine (EVM) has three areas where it can store items: (a) the *storage* is where all *contract state* variables reside, every contract has its own storage and it is persistent between external function calls (transactions) and quite expensive to use; (b) the *memory* is used to hold temporary values, and it is erased between transactions and thus is cheaper to use; (c) the *stack* is used to carry out operations and it is free to use, but can only hold a limited amount of values.

The rationale behind the resource-aware smart contract semantics, instrumented with gas consumption, is three-fold. First, paying for gas at the moment of proposing the transaction does not allow the emitter to waste other parties' (aka *miners*) computational power by requiring them to perform a lot of worthless intensive work. Second, gas fees disincentivize users to consume too much of replicated *storage*, which is a valuable resource in a blockchain-based consensus system. Finally, such a semantics puts a cap on the number of computations that a transaction can execute, hence prevents attacks based on non-terminating executions (which could otherwise, *e.g.*, make all miners loop forever).

The gas-instrumented operational semantics of EVM has introduced novel challenges *wrt*. sound static reasoning about resource consumption, correctness, and security of replicated computations: *(i)* While the EVM specification (Wood, 2014) provides the precise gas consumption of the low-level operations, most of the smart contracts are written in high-level languages, such as Solidity (Ethereum, 2018b) or Vyper (Ethereum, 2018c).

The translation of the high-level language constructs to the low-level ones makes static estimation of runtime gas bounds challenging (as we will see throughout this paper), and is implemented in an *ad-hoc* way by state-of-the art compilers, which are only able to give constant gas bounds, or return ∞ otherwise. *(ii)* As noted in the recent study by Grech et al. (2018) and Foundation (2018), in general it is dangerous for a smart contract to make its gas consumption dependent on the size of the data it stores (*i.e.*,

☆ Editor: A. Bertolino.

* Corresponding author.
E-mail addresses: elvira@sip.ucm.es (E. Albert), jcorreas@ucm.es (J. Correas), pabgordi@ucm.es (P. Gordillo), guillermo.roman@upm.es (G. Román-Díez), alberu04@ucm.es (A. Rubio).

https://doi.org/10.1016/j.jss.2021.110923
0164-1212/

directions. First, we are studying a more precise abstraction for the memory-allocated data (*i.e.*, for the abstraction explained in Section 3). Also, we aim at handling bit-wise operations by including in our tool an abstraction for them.

CRediT authorship contribution statement

Elvira Albert: Conceptualization, Methodology, Software, Validation, Investigation, Formal analysis, Writing - original draft, Writing - review & editing. **Jesús Correas:** Conceptualization, Methodology, Software, Validation, Investigation, Formal analysis, Writing - original draft, Writing - review & editing. **Pablo Gordillo:** Conceptualization, Methodology, Software, Validation, Investigation, Formal analysis, Writing - original draft, Writing - review & editing. **Guillermo Román-Díez:** Conceptualization, Methodology, Software, Validation, Investigation, Formal analysis, Writing - original draft, Writing - review & editing. **Albert Rubio:** Conceptualization, Methodology, Software, Validation, Investigation, Formal analysis, Writing - original draft, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was funded partially by the Spanish MCIU, AEI and FEDER (EU) projects RTI2018-094403-B-C31 and RTI2018-094403-B-C33, by the CM project S2018/TCS-4314 co-funded by EIE Funds of the European Union and by the UCM CT27/16-CT28/16 grant.

References

Aho, A.V., Lam, M.S., Sethi, R., Ullman, J.D., 2006. Compilers: Principles, Techniques, and Tools, second ed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Albert, E., Arenas, P., Correas, J., Genaim, S., Gómez-Zamalloa, M., Puebla, G., Román-Díez, G., 2015a. Object-sensitive cost analysis for concurrent objects. STVR 25 (3), 218–271.

Albert, E., Arenas, P., Genaim, S., Puebla, G., 2008. Automatic inference of upper bounds for recurrence relations in cost analysis. In: Alpuente, M., Vidal, G. (Eds.), Static Analysis, 15th International Symposium, SAS 2008, Valencia, Spain, July 16-18, 2008. Proceedings. In: Lecture Notes in Computer Science, vol. 5079, Springer, pp. 221–237.

Albert, E., Arenas, P., Genaim, S., Puebla, G., 2011. Closed-form upper bounds in static cost analysis. J. Automat. Reason. 46 (2), 161–203.

Albert, E., Arenas, P., Genaim, S., Puebla, G., Román-Díez, G., 2014. Conditional termination of loops over heap-allocated data. Sci. Comput. Program. 92, 2–24.

Albert, E., Arenas, P., Genaim, S., Puebla, G., Zanardini, D., 2012. Cost analysis of object-oriented bytecode programs. Theor. Comput. Sci. 413 (1), 142–159 (Special Issue on Quantitative Aspects of Programming Languages).

Albert, E., Correas, J., Gordillo, P., Hernández-Cerezo, A., Román-Díez, G., Rubio, A., 2020a. Analyzing Smart Contracts: From EVM to a Sound Control-Flow Graph. Tech. Rep.

Albert, E., Correas, J., Gordillo, P., Román-Díez, G., Rubio, A., 2020b. GASOL: Gas analysis and optimization for ethereum smart contracts. In: Biere, A., Parker, D. (Eds.), Proceedings of 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2020. In: Lecture Notes in Computer Science, vol. 12079, pp. 118–125.

Albert, E., Correas, J., Román-Díez, G., 2015b. Non-cumulative resource analysis. In: Proceedings of 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2015. In: Lecture Notes in Computer Science, vol. 9035, Springer, pp. 85–100.

Albert, E., Gordillo, P., Livshits, B., Rubio, A., Sergey, I., 2018. Ethir: A framework for high-level analysis of ethereum bytecode. In: Lahiri, S., Wang, C. (Eds.), 16th International Symposium on Automated Technology for Verification and Analysis, ATVA 2018. Proceedings. In: Lecture Notes in Computer Science, vol. 11138, Springer, pp. 513–520.

Albert, E., Gordillo, P., Rubio, A., Sergey, I., 2019. Running on fumes: Preventing out-of-gas vulnerabilities in ethereum smart contracts using static resource analysis. In: Ganty, P., Kaâniche, M. (Eds.), 13th International Conference on Verification and Evaluation of Computer and Communication Systems, VECoS 2019. Proceedings. In: Lecture Notes in Computer Science, vol. 11847, pp. 63–78.

Amani, S., Bégel, M., Bortin, M., Staples, M., 2018. Towards verifying ethereum smart contract bytecode in isabelle/HOL. In: Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2018, Los Angeles, CA, USA, 2018. ACM, pp. 66–77.

Ambroladze, N., 2018. Fast and Scalable Analysis of Smart Contracts (Master's thesis). Swiss Federal Institute of Technology Zurich, Switzerland.

Andersen, L.O., 1994. Program Analysis and Specialization for the C Programming Language (Ph.D. thesis). University of Cophenhagen.

Bernani, T., 2016. Oraclize. http://www.oraclize.it.

Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Gollamudi, A., Gonthier, G., Kobeissi, N., Kulatova, N., Rastogi, A., Sibut-Pinote, T., Swamy, N., Zanella-Béguelin, S., 2016. Formal verification of smart contracts: Short paper. In: Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security, PLAS@CCS 2016, Vienna, Austria, October 2016. ACM, pp. 91–96.

Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y., 1999. Symbolic model checking without BDDs. In: Proceedings of 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 1999. In: Lecture Notes in Computer Science, vol. 1579, Springer, pp. 193–207.

Bloxy, 2018a. Bloxy. https://bloxy.info/.

Brent, L., Jurisevic, A., Kong, M., Liu, E., Gauthier, F., Gramoli, V., Holz, R., Scholz, B., 2018. Vandal: A scalable security analysis framework for smart contracts. arXiv:1809.03981.

Brockschmidt, M., Emmes, F., Falke, S., Fuhs, C., Giesl, J., 2016. Analyzing runtime and size complexity of integer programs. ACM Trans. Program. Lang. Syst. 38 (4), 13:1–13:50.

Chen, T., Feng, Y., Li, Z., Zhou, H., Luo, X., Li, X., Xiao, X., Chen, J., Zhang, X., 2020. Gaschecker: Scalable analysis for discovering gas-inefficient smart contracts. IEEE Trans. Emerg. Top. Comput. PP(99), 1–14.

Chen, T., Li, X., Luo, X., Zhang, X., 2017. Under-optimized smart contracts devour your money. In: IEEE 24th International Conference on Software Analysis, Evolution and Reengineering, SANER. IEEE Computer Society, pp. 442–446.

Chen, T., Li, Z., Zhou, H., Chen, J., Luo, X., Li, X., Zhang, X., 2018. Towards saving money in using smart contracts. In: Zisman, A., Apel, S. (Eds.), Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results, ICSE (NIER) 2018, Gothenburg, Sweden, 2018. ACM, pp. 81–84.

Ethereum, 2018a. Etherscan. https://etherscan.io.

Ethereum, 2018b. Solidity. https://solidity.readthedocs.io.

Ethereum, 2018c. Vyper. https://vyper.readthedocs.io.

EthereumPot, 2017. The EthereumPot contract. https://etherscan.io/address/0x5a13caa82851342e14cd2ad0257707cddb8a31b7.

Foundation, E., 2018. Safety - Ethereum Wiki. https://github.com/ethereum/wiki/wiki/Safety, (Last Accessed on 14 November 2018).

Grech, N., Brent, L., Scholz, B., Smaragdakis, Y., 2019. Gigahorse: thorough, declarative decompilation of smart contracts. In: Atlee, J.M., Bultan, T., Whittle, J. (Eds.), Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25–31, 2019. IEEE / ACM, pp. 1176–1186.

Grech, N., Kong, M., Jurisevic, A., Brent, L., Scholz, B., Smaragdakis, Y., 2018. MadMax: surviving out-of-gas conditions in Ethereum smart contracts. PACMPL 2 (OOPSLA), 116:1–116:27.

Grishchenko, I., Maffei, M., Schneidewind, C., 2018. A semantic framework for the security analysis of ethereum smart contracts. In: Principles of Security and Trust - 7th International Conference, POST 2018, Thessaloniki, Greece. Proceedings. In: Lecture Notes in Computer Science, vol. 10804, Springer, pp. 243–269.

Grossman, S., Abraham, I., Golan-Gueta, G., Michalevsky, Y., Rinetzky, N., Sagiv, M., Zohar, Y., 2018. Online detection of effectively callback free objects with applications to smart contracts. PACMPL 2 (POPL), 48:1–48:28.

Hajdu, A., Jovanovic, D., 2020. SMT-friendly formalization of the solidity memory model. In: Müller, P. (Ed.), Programming Languages and Systems - 29th European Symposium on Programming, ESOP 2020, Dublin, Ireland, Proceedings. In: Lecture Notes in Computer Science, vol. 12075, Springer, pp. 224–250.

He, J., Balunovic, M., Ambroladze, N., Tsankov, P., Vechev, M.T., 2019. Learning to fuzz from symbolic execution with application to smart contracts. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, 2019. ACM, pp. 531–548. http://dx.doi.org/10.1145/3319535.3363230.

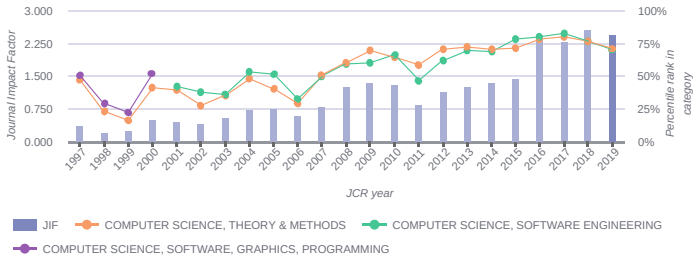
Hildenbrandt, E., Saxena, M., Rodrigues, N., Zhu, X., Daian, P., Guth, D., Park, D., Zhang, Y., Moore, B., Rosu, G., 2018. KEVM: A complete semantics of the ethereum virtual machine. In: 31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, 2018. IEEE Computer Society, pp. 204–217.

InCites Journal Citation Reports

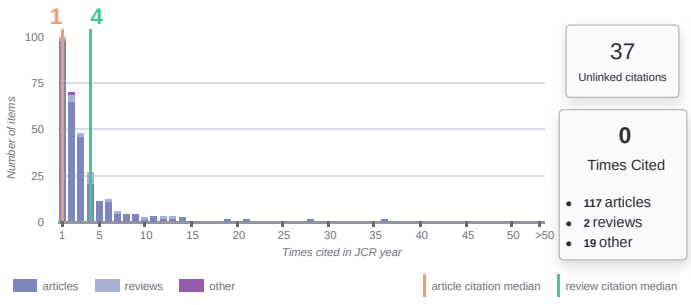
The data in the two graphs below and in the Journal Impact Factor calculation panels represent citation activity in 2019 to items published in the journal in the prior two years. They detail the components of the Journal Impact Factor. Use the "All Years" tab to access key metrics and additional data for the current year and all prior years for this journal.

2019 Journal Impact Factor & percentile rank in category for: JOURNAL OF SYSTEMS AND SOFTWARE

2.450
2019 Journal Impact Factor



2019 JIF Citation Distribution for: JOURNAL OF SYSTEMS AND SOFTWARE



Rank

Rank

JCR Impact Factor

JCR Year	COMPUTER SCIENCE, SOFTWARE ENGINEERING			COMPUTER SCIENCE, THEORY & METHODS			COM
	Rank	Quartile	JIF Percentile	Rank	Quartile	JIF Percentile	
2019	33/108	Q2	69.907	32/108	Q2	70.833	
2018	26/107	Q1	76.168	25/105	Q1	76.667	
2017	19/104	Q1	82.212	21/103	Q1	80.097	
2016	22/106	Q1	79.717	23/104	Q1	78.365	
2015	24/106	Q1	77.830	31/105	Q2	70.952	
2014	33/104	Q2	68.750	31/102	Q2	70.098	
2013	33/105	Q2	69.048	29/102	Q2	72.059	
2012	41/105	Q2	61.429	30/100	Q2	70.500	
2011	57/104	Q3	45.673	42/99	Q2	58.081	
2010	34/99	Q2	66.162	35/97	Q2	64.433	
2009	38/93	Q2	59.677	29/92	Q2	69.022	
2008	36/86	Q2	58.721	34/84	Q2	60.119	
2007	43/84	Q3	49.405	40/79	Q3	50.000	
2006	56/82	Q3	32.317	54/75	Q3	28.667	
2005	39/79	Q2	51.266	43/71	Q3	40.141	
2004	36/76	Q2	53.289	37/70	Q3	47.857	
2003	51/78	Q3	35.256	46/70	Q3	35.000	
2002	49/77	Q3	37.013	51/69	Q3	26.812	
2001	44/75	Q3	42.000	44/71	Q3	38.732	
2000	n/a	n/a	n/a	40/67	Q3	41.045	
1999	n/a	n/a	n/a	52/61	Q4	15.574	
1998	n/a	n/a	n/a	50/64	Q4	22.656	
1997	n/a	n/a	n/a	32/59	Q3	46.610	