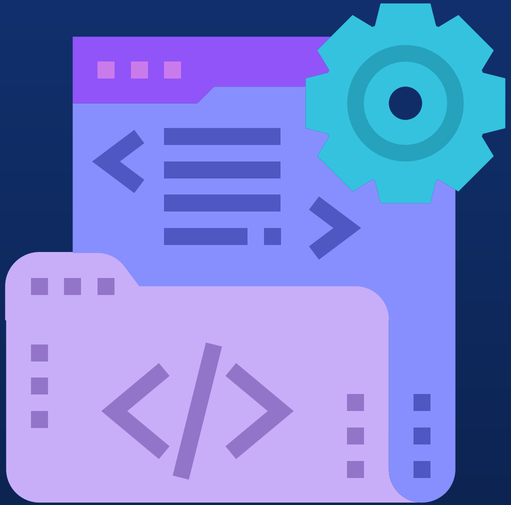
Formal approach in Smart Contracts

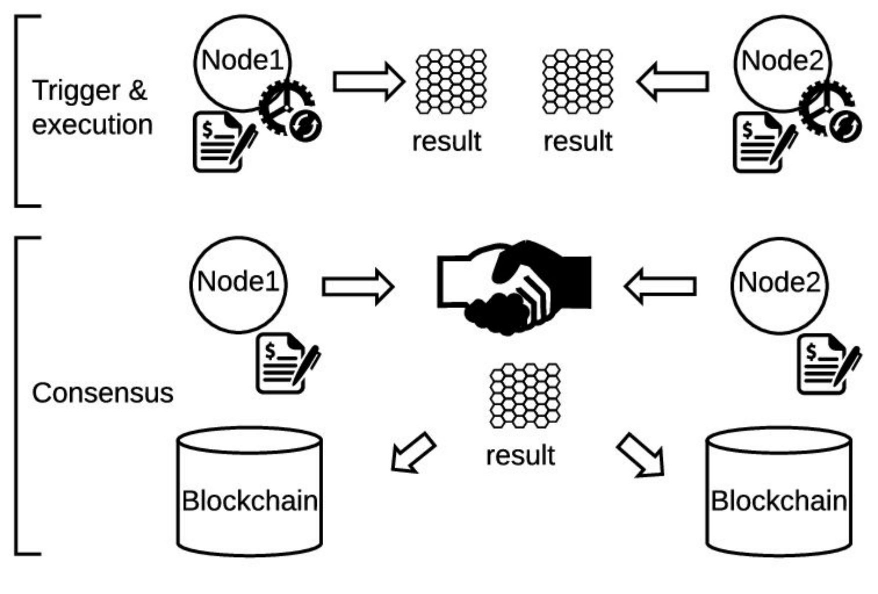
LANMR 2024
PCIC - UNAM
René Adrián Dávila Pérez

01

Smart Contracts vs Traditional Software



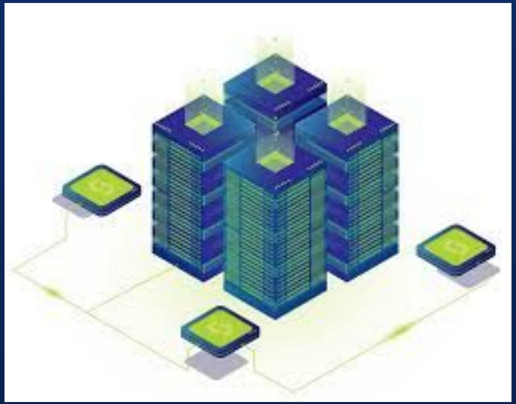
Execution



Source: eGov-DAO: a Better Government using Blockchain based Decentralized Autonomous Organization. 166-171. 10.1109/ICEDEG.2018.8372356.

Storage

- **Smart Contracts:** Blockchain Storage, but limited.
- **Traditional Software:** Server storage, but with possibilities for manipulation.



Source: https://storpool.com/overview

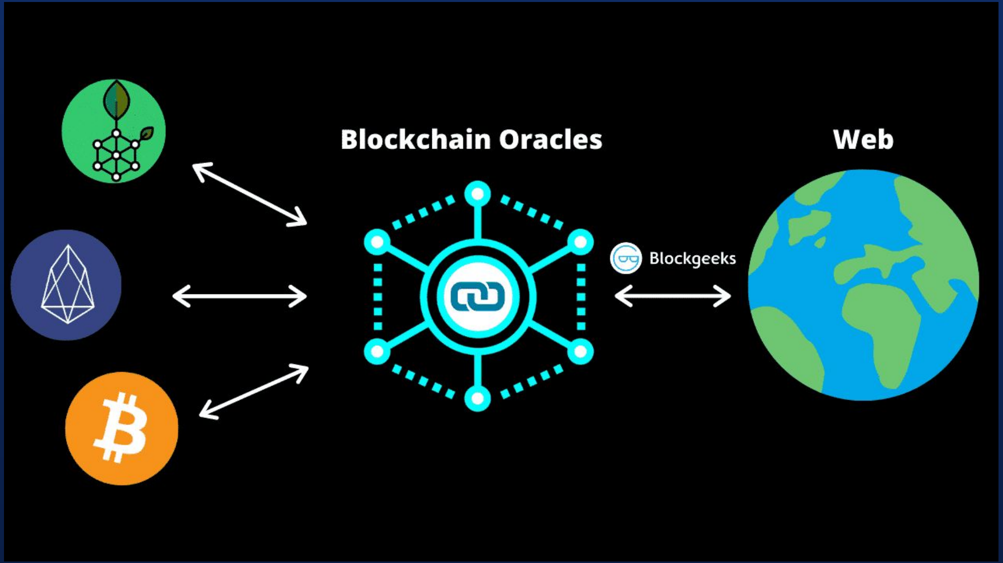
Security

- **Smart Contracts:** Design errors may cause serious issues.
- **Traditional Software:** Errors can be fixed with updates or patches.

Source: https://www.dotmagazine.online/issues/blockchain-e-government/blockchain-security/smart-contract-security-expect-and-deal-with-attacks

Interoperability

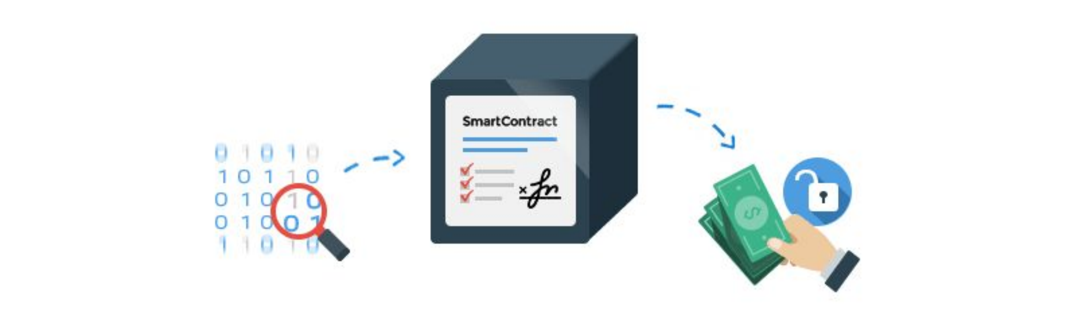
- **Smart Contracts:** Bridges or interoperability protocols.
- **Traditional Software:** API's, web services, or integration protocols.



Source: https://blockgeeks.com/guides/blockchain-oracles/

Transparency

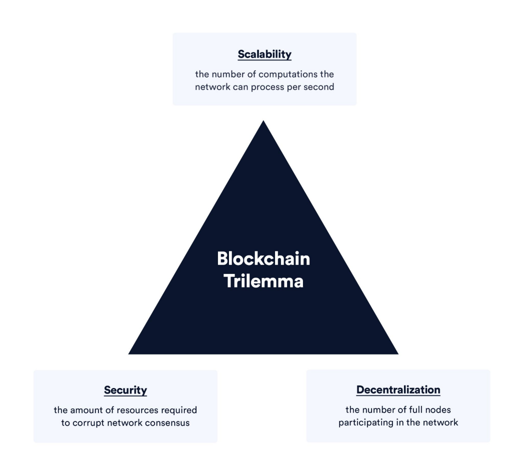
- **Smart Contracts:** Eliminating third parties increases trust through compliance with terms.
- **Traditional Software:** They depend on the owner of the software; usually, the code is not freely accessible.



Source: https://sloboda-studio.com/blog/the-guide-to-smart-contracts-for-business-owners/

Scalability

- **Smart Contracts:** It is a challenge due to the limitations of Blockchain in terms of transaction speed and capacity.
- **SW:** It is more feasible to add hardware resources, optimize code, or use efficient architectures, such as microservices or the cloud.


Scalability
the number of computations the network can process per second
Blockchain Trilemma
Security
the amount of resources required to corrupt network consensus
Decentralization
the number of full nodes participating in the network

Source: https://chain.link/education-hub/smart-contract-platforms

02

Finite State Machines

What is the Smart Contract Formalization process in FSM?

Identification of States

Identify the different states in which the contract may be found.

Events and Conditions

Determine the events that caused transitions between states and define the conditions under which transitions can be made.

Transitions

It specifies how and when one can move from one state to another through events and conditions.

Example: Auction System

1. States
   a. *Open.*
   b. *Close.*
   c. *End.*
2. Events
   a. *Receive offer.*
   b. *Close auction.*
   c. *End auction.*
3. Transitions
   a. *Open to Closed when event happens Close auction.*
   b. *Closed to Ended when event happens End Auction.*
   c. *Open to Ended if the auction time expires without being closed.*

How do FSMs compare to other formalisms?

Abstraction

They operate at a lower level of abstraction, focusing on discrete states and transitions, while logical systems, especially first-order logic, operate at a higher, more general level, allowing the expression of abstract concepts and relationships between objects.

Expressiveness

FSMs are limited compared to logical systems. For example, they cannot capture concepts such as "for all" or "exists," which are fundamental in first-order logic.

Modeling

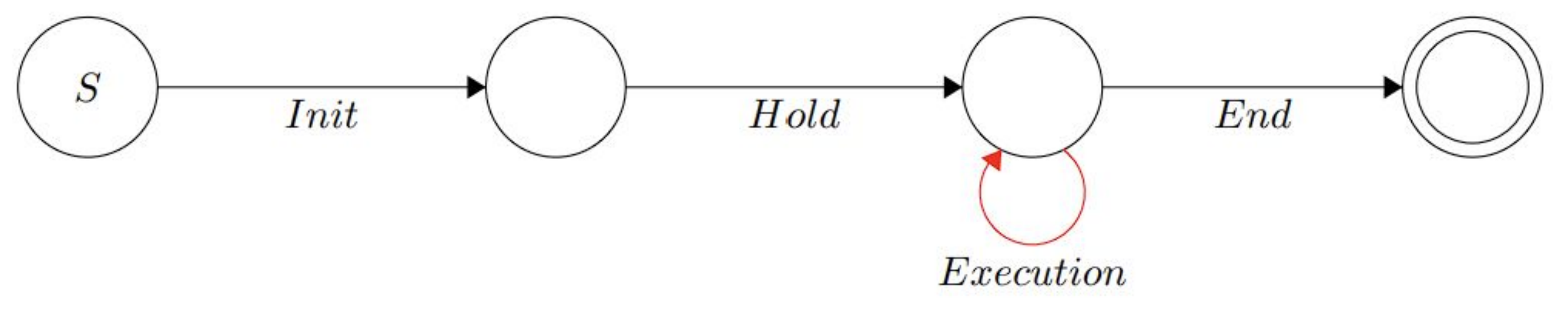
FSMs are useful in systems where behavior can be modeled as a finite set of states and transitions. At the same time, logic is used in applications where reasoning about abstract properties and complex relationships is necessary.

03

SC's Errors through Finite State Machines

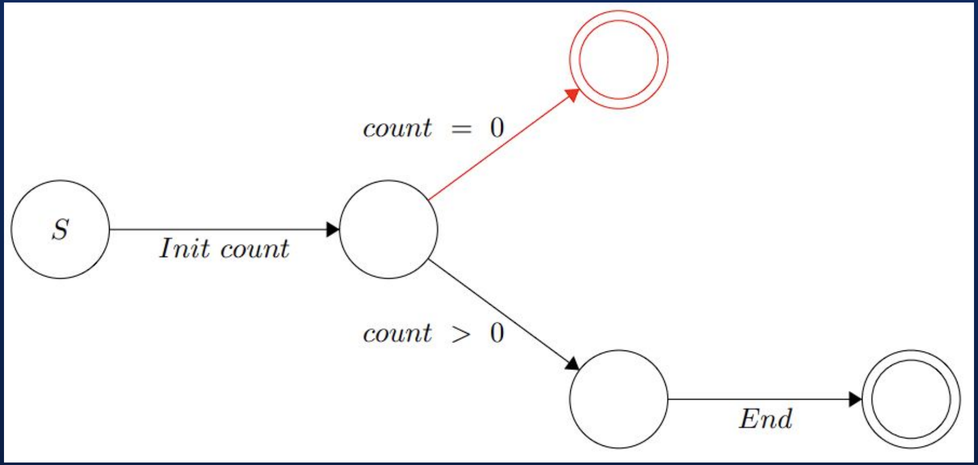
Reentrancy

- An attacker can use a contract's ability to make external calls before all its transactions are finalized. This allows the vulnerable function to be called repeatedly before the original execution is complete.
- **Example:** This was the problem in the DAO hack on Ethereum, where attackers drained millions of dollars.



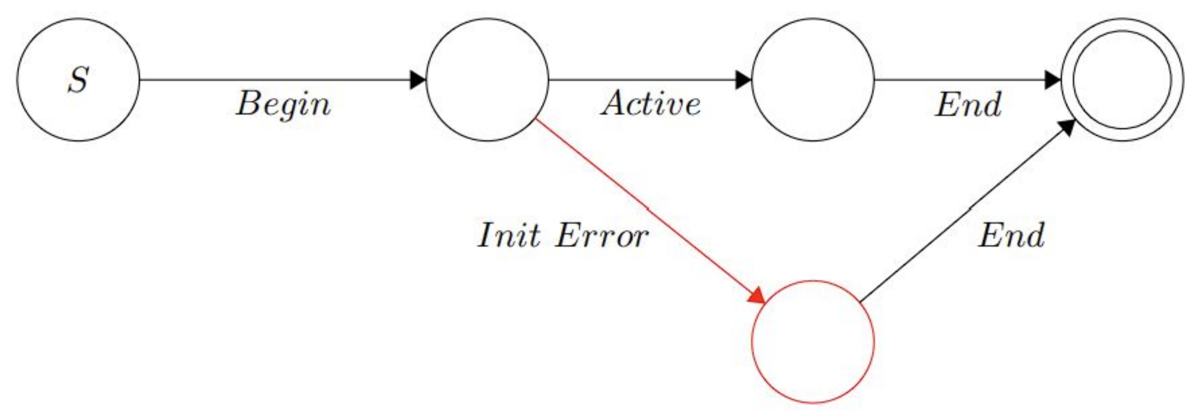
Integer Overflow/Underflow

- They occur when an arithmetic operation results in a value outside the range allowed by the data type (for example, exceeding the limit of a uint256).
- **Example:** Variable with a value that causes overflow.



Initialization

- Contracts can be deployed without properly initializing their variables, which could allow an attacker to take control of the contract.
- **Example:** A misconfigured contract with no restrictions on its initialize() function could allow any user to execute it and modify the state of the contract.



FSM's relationship with DLs

- Finite State Machines (FSM) and description logics (DL) are related in the context of systems verification and specification, especially in model-checking theory.

- The connection between both is mainly given by using description logic to describe and reason about the properties of finite state machines and the systems they can model.

Relationship of FSM with families of DLs

ALC

A basic descriptive logic that allows describing concepts and relationships with logical operators.

ALCO

It is a descriptive logic that allows for working basic concepts, complements, and nominals.

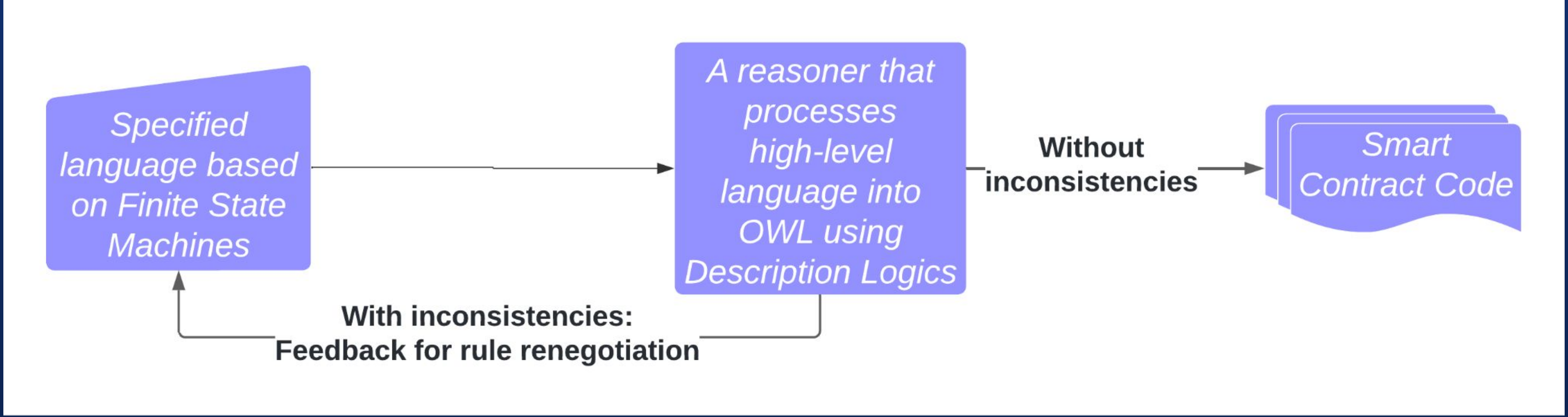
SHOIQ

It is a more advanced descriptive logic that supports transitivity, role hierarchies, cardinalities, and nominals, being more suitable for representing complex and detailed knowledge structures.

04

Future Work

Figure. Verification tool template.


Specified language based on Finite State Machines
A reasoner that processes high-level language into OWL using Description Logics
Without inconsistencies
Smart Contract Code
With inconsistencies: Feedback for rule renegotiation

Actual term

- Investigate identified errors in Smart Contracts.
- Classify the errors that can be modeled using FSMs (Finite State Machines).
- Define consistency in Smart Contracts based on FSMs (Finite State Machines).
- Publish the results of the definition and classification.

Thank You

photographic_ren@comunidad.unam.mx
+52 5530695535

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik**

References

[1] Tesnim Abdellatif & Kei-Léo Brousmiche. «Formal verification of smart contracts based on users and blockchain behaviors models». Available: *IFIP NTMS International Workshop on Blockchains and Smart Contracts (BSC)*. Paris, France, feb. de 2018. url: https://hal.science/hal-01760787.

[2] Xiaomin Bai et al. «Formal Modeling and Verification of Smart Contracts». Available: *Proceedings of the 2018 7th International Conference on Software and Computer Applications*. ICSCA '18. Kuantan, Malaysia: Association for Computing Machinery, 2018, págs. 322-326. isbn: 9781450354141. doi: 10.1145/3185089.3185138. url: https://doi.org/10.1145/3185089.3185138.

[3] Anastasia Mavridou et al. «VeriSolid: Correct-by-Design Smart Contracts for Ethereum». Available: *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers*. St. Kitts, Saint Kitts y Nevis: Springer-Verlag, 2019, págs. 446-465. isbn: 978-3-030-32100-0. doi: 10 . 1007 / 978 - 3 - 030 - 32101 - 7_27. url: https://doi.org/10.1007/978-3-030-32101-7_27.

References

[4] Zeinab Nehai, Pierre-Yves Piriou & Frédéric Daumas. «Model-Checking of Smart Contracts». Available: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. 2018, págs. 980-987. doi: 10.1109/Cybermatics_2018.2018. 00185.

[5] Franz Baader et al. *The Description Logic Handbook: Theory, Implementation and Applications*. 2.a ed. Cambridge University Press, 2007. doi: 10 . 1017 / CBO9780511711787.

[6] Christel Baier & Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008. isbn: 026202649.

[7] Yongfeng Huang et al. «Smart Contract Security: A Software Lifecycle Perspective». Available: *IEEE Access 7 (2019)*, págs. 150184-150202. doi: 10.1109/ACCESS.2019.2946988.

References

[8] Anastasia Mavridou & Aron Laszka. «Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach». Available: *Financial Cryptography and Data Security*. Ed. por Sarah Meiklejohn y Kazue Sako. Berlin, Heidelberg:Springer Berlin Heidelberg, 2018, págs. 523-540. isbn: 978-3-662-58387-6.

[9] Sebastian Rudolph. «Foundations of description logics». Available: *Reasoning Web International Summer School*. Springer, 2011, págs. 76-136.