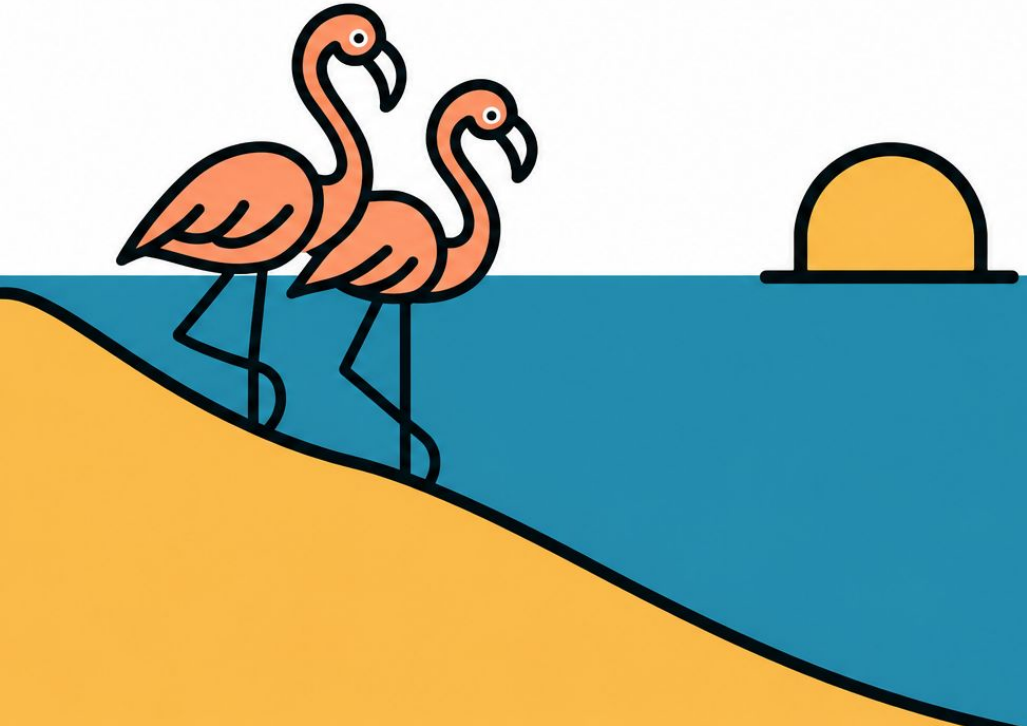
DLT2026

JUNE 1-6
PULA

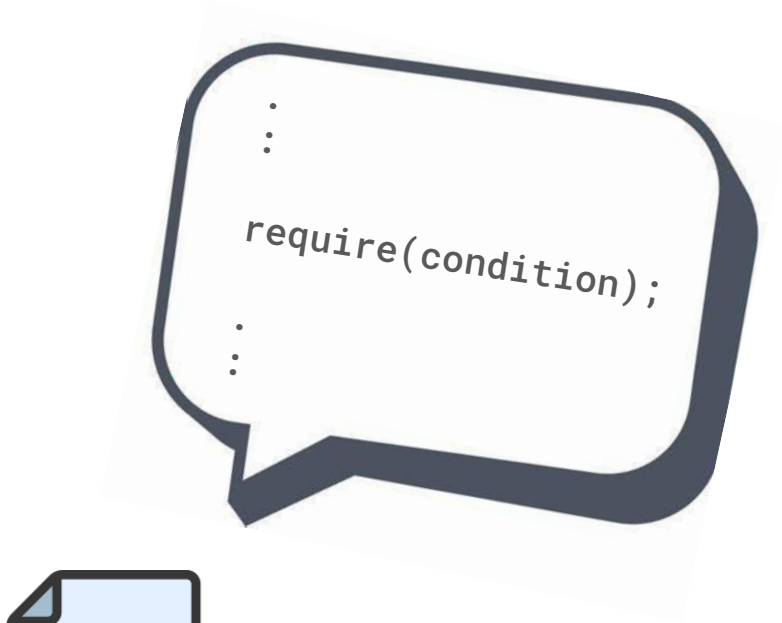
UNICA
UNIVERSITÀ DEGLI STUDI
DI CAGLIARI

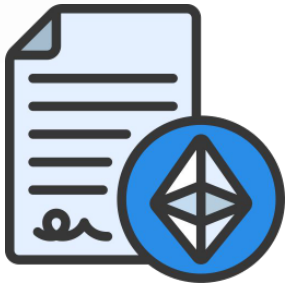


Emily Priyadarshini

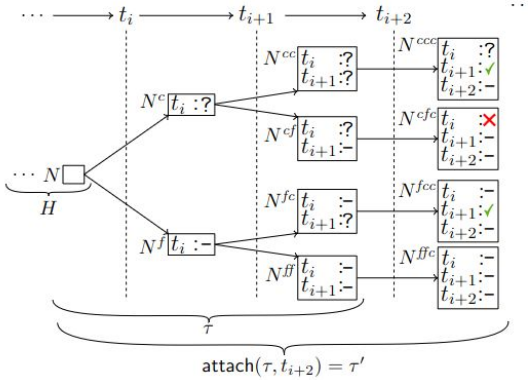
Massimo Bartoletti

Smart contracts let you perform an action only if the requirements are satisfied.





WHAT IF smart contracts let you perform an action provided some requirement is satisfied in the FUTURE?



(a)

Monitoring the Future of Smart Contracts*

Margarita Capretto[1,2], Martin Ceresa[1], and César Sánchez[1]

[1] IMDEA Software Institute, Spain
[2] Universidad Politécnica de Madrid (UPM), Madrid, Spain

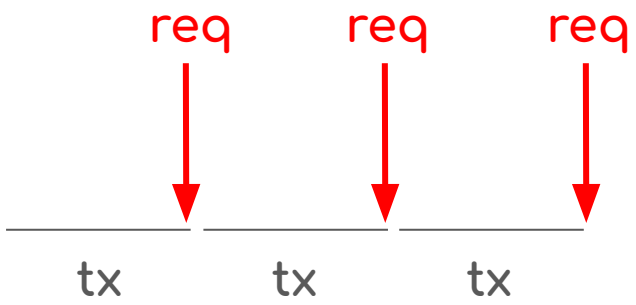
Abstract. Blockchains are decentralized systems that provide trustable execution guarantees. Smart contracts are programs written in specialized programming languages running on blockchains that govern how tokens and cryptocurrency are sent and received. Smart contracts can invoke other smart contracts during the execution of transactions always initiated by external users.

Once deployed, smart contracts cannot be modified, so techniques like runtime verification are very appealing for improving their reliability. However, the conventional model of computation of smart contracts is transactional: once operations commit, their effects are permanent and cannot be undone.

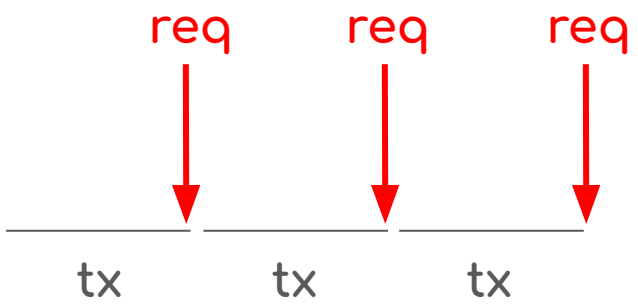
In this paper, we proposed the concept of *future monitors* which allows monitors to remain waiting for future transactions to occur before committing or aborting. This is inspired by optimistic rollups, which are modern blockchain implementations that increase efficiency (and reduce cost) by delaying transaction effects. We exploit this delay to propose a model of computation that allows (bounded) future monitors. We show our monitors correct respect of legacy transactions, how they implement future bounded monitors and how they guarantee progress. We illustrate the use of future bounded monitors to implement correctly multi-transaction flash loans.

https://link.springer.com/chapter/10.1007/978-3-031-57259-3_6

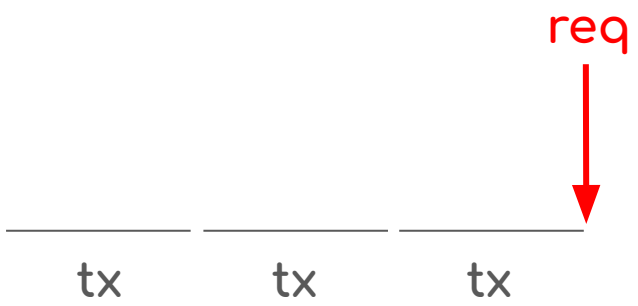
Now

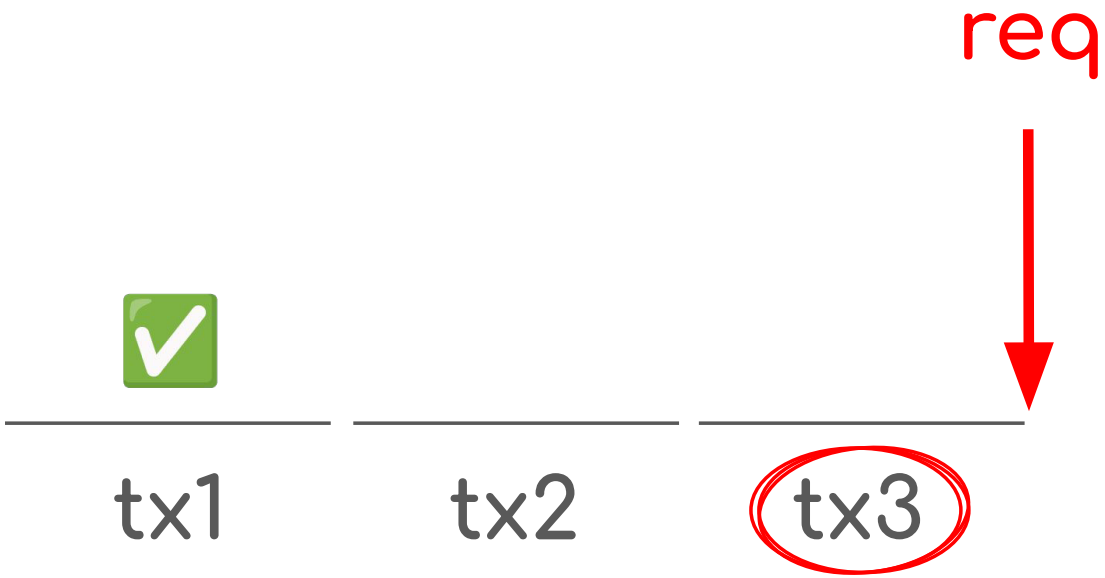


Now



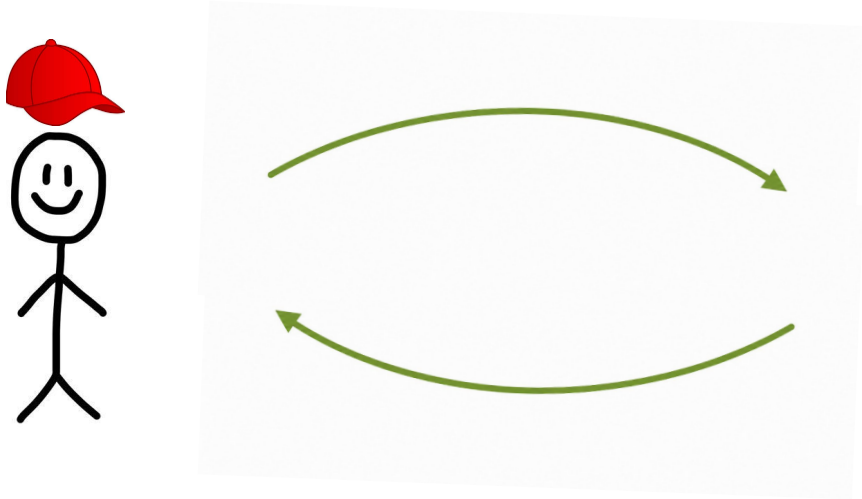
Future




req
✅
tx1
tx2
tx3

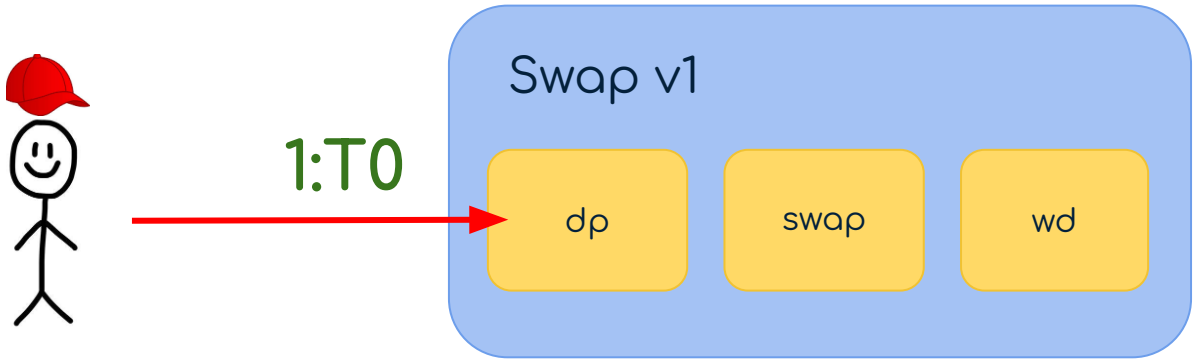
Player P0

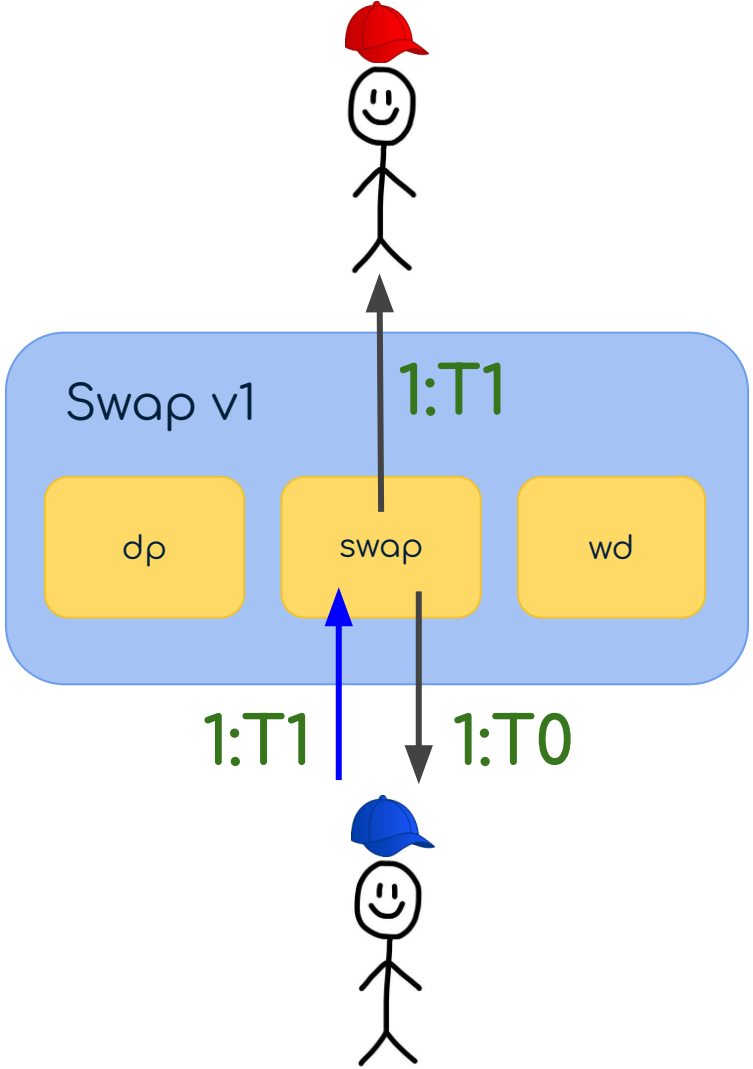
Player P1

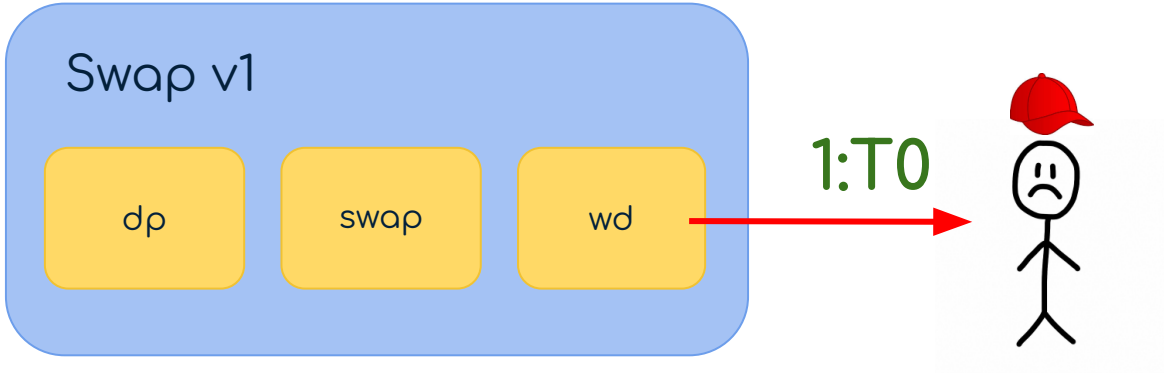


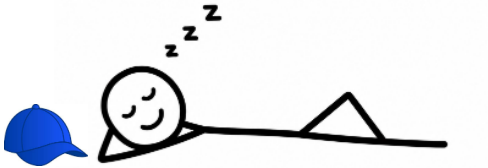
1 Token T0

1 Token T1

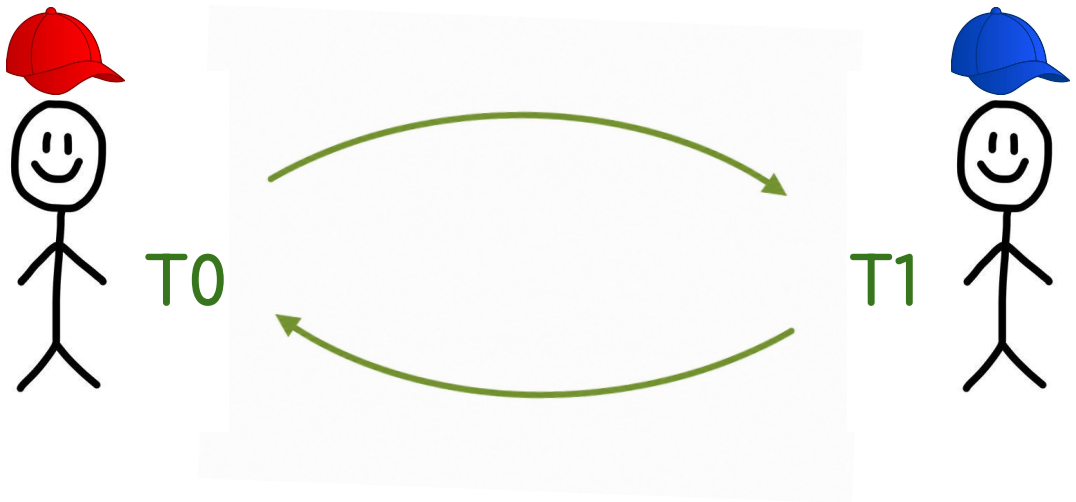

Swap v1
1:T0
dp
swap
wd


Swap v1
1:T1
dp
swap
wd
1:T1
1:T0

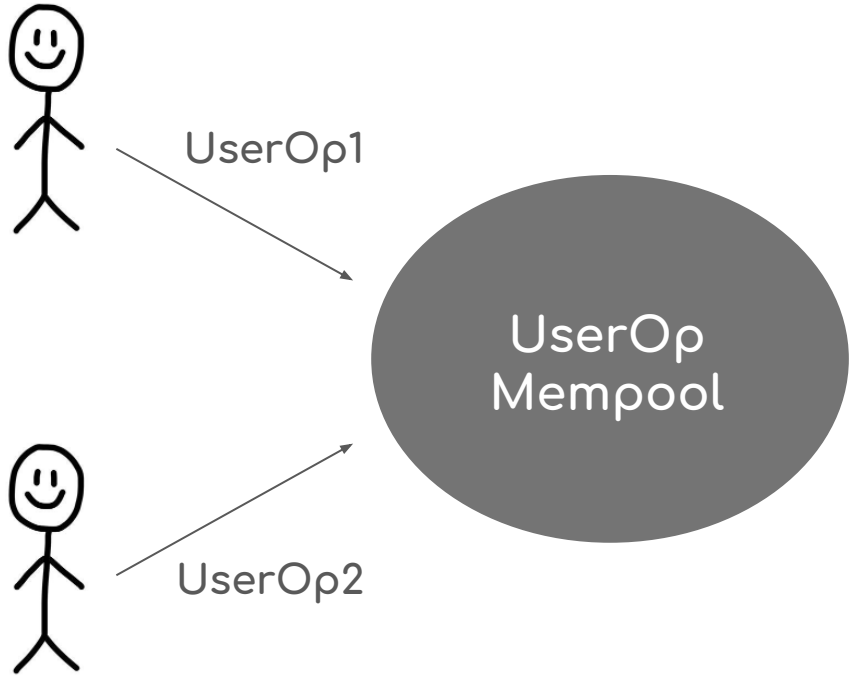

Swap v1
dp
swap
wd
1:T0



ATOMIC SWAP

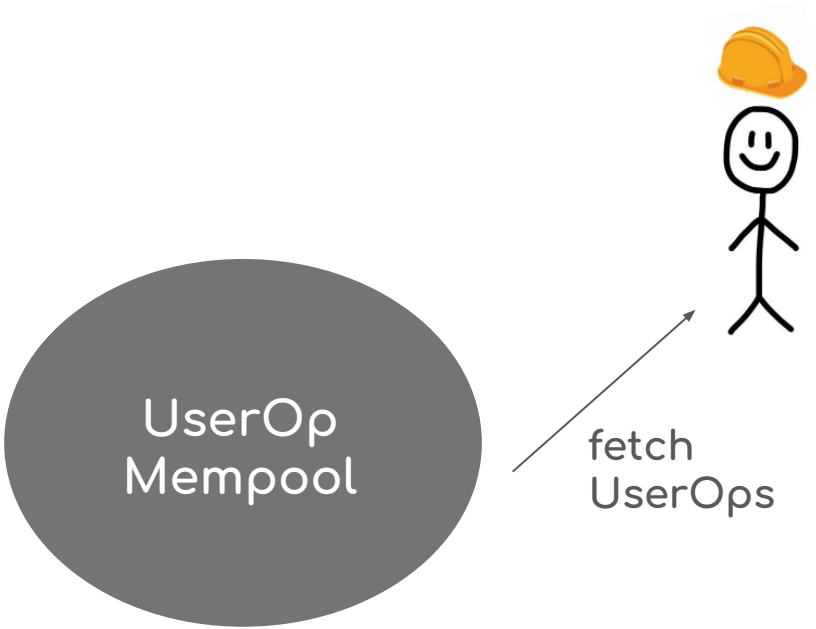


ACCOUNT ABSTRACTION

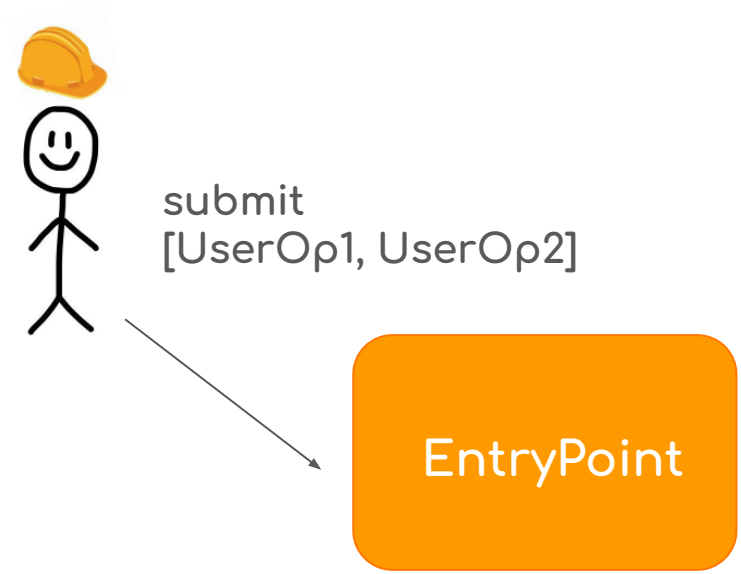


EIP-4337, 2023

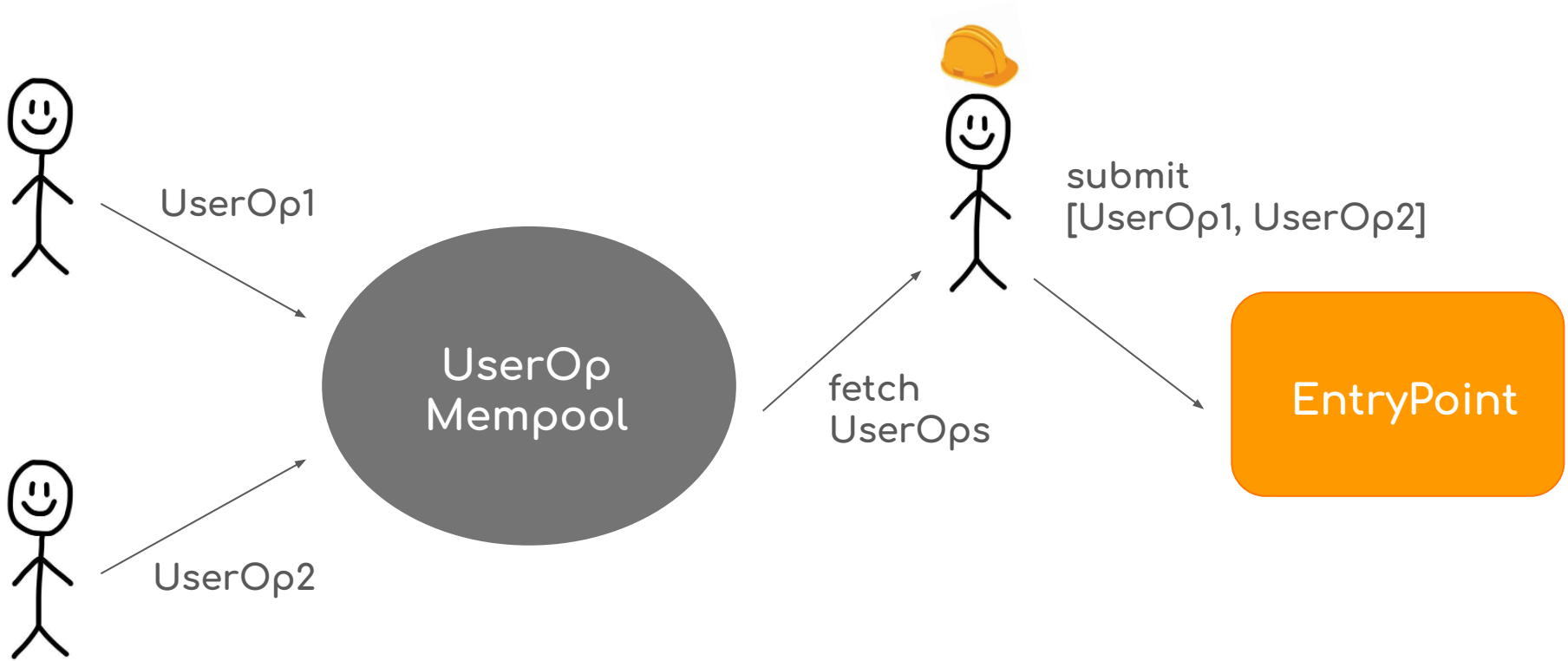
ACCOUNT ABSTRACTION

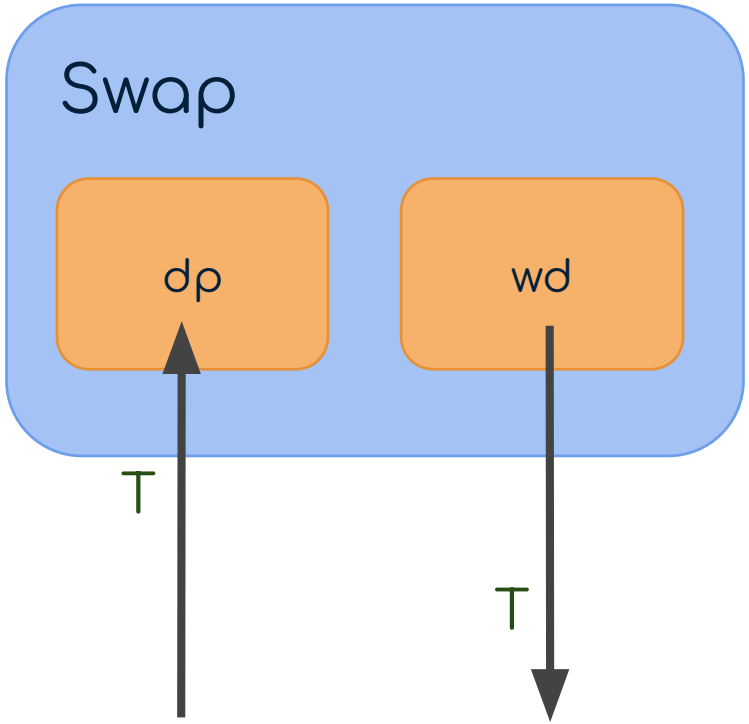


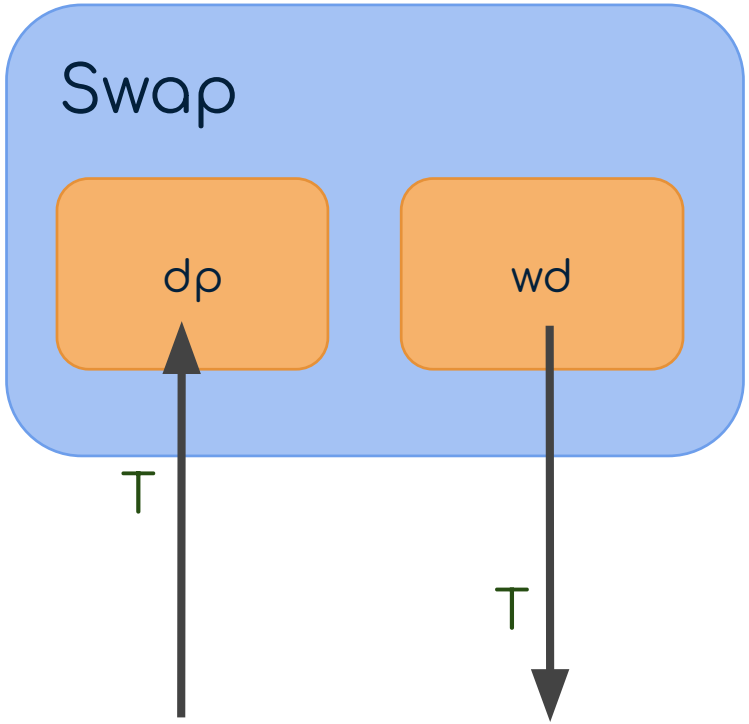
ACCOUNT ABSTRACTION



ACCOUNT ABSTRACTION



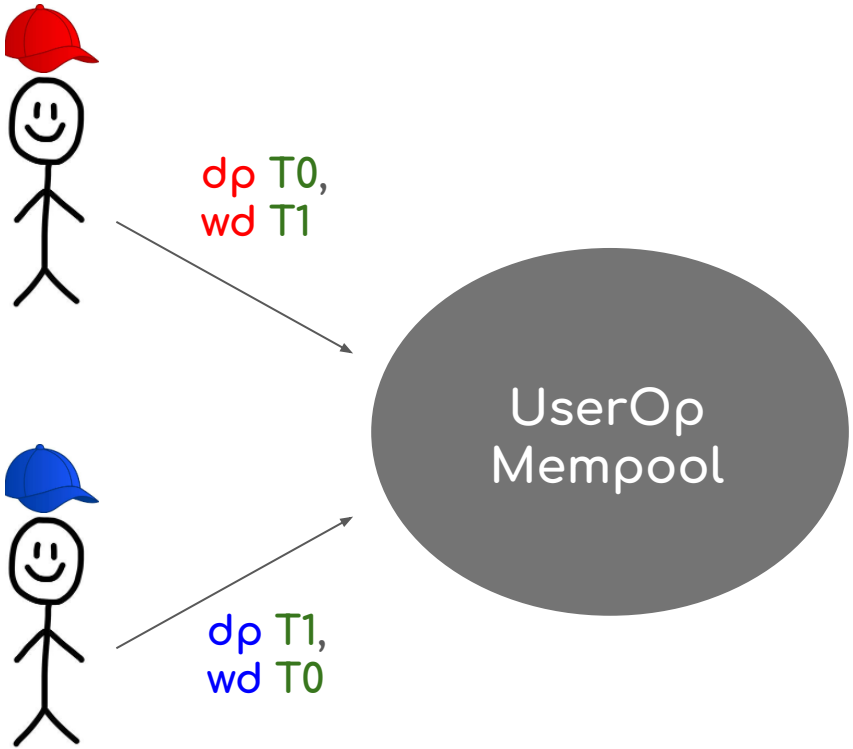

Swap
dp
wd
T
T



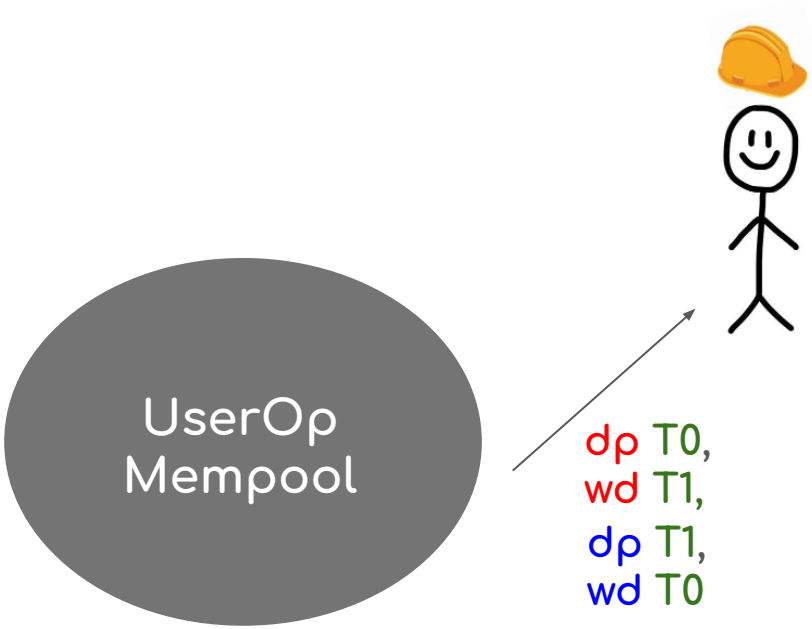
EntryPoint*

* only allow bundles that perform a valid swap

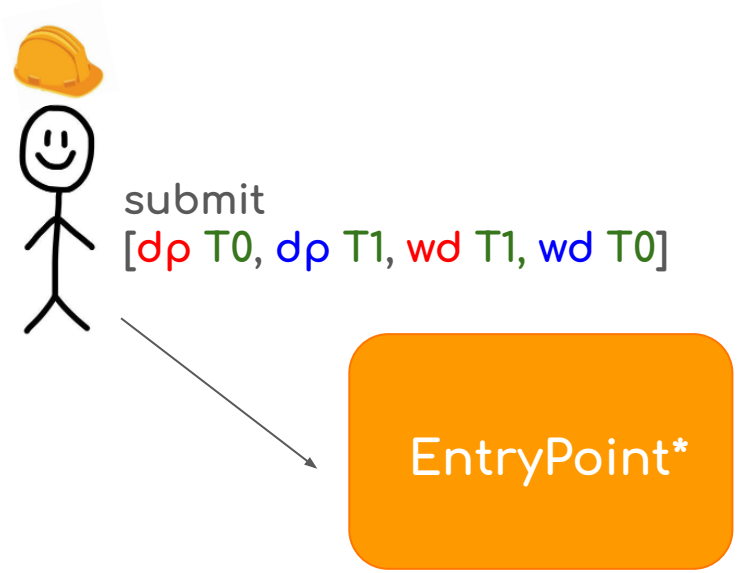
ATOMIC SWAP



ATOMIC SWAP



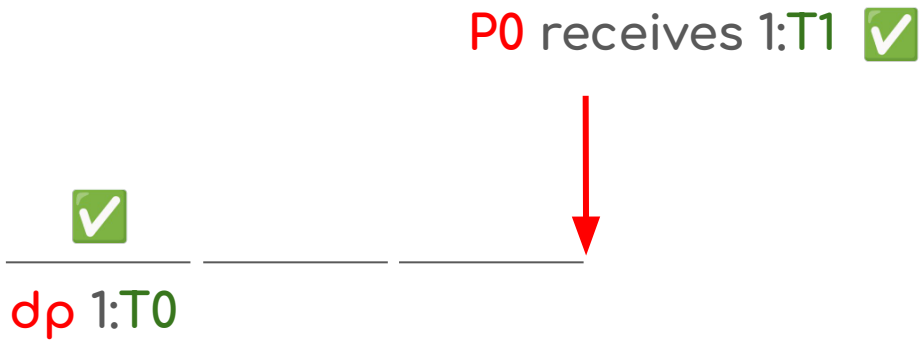
ATOMIC SWAP



tx: $X_1, X_2, \ldots, X_n$

tx bundle: $[X_1, X_2, \ldots, X_n]$

Programmable bundles: `commit(`E`)`



TO DO LIST

- [x] formal semantics
- [ ] use cases
- [ ] algorithms
- [ ] costs
- [ ] composability & attacks

MAIN IDEAS

- Enable smart contracts to predicate on the *future*
- This new capability facilitates *atomicity*
- Studying a general model of *bundles* permits us to freely explore additional capabilities
- *Bundles* reshape user-builder dynamics & unlock novel forms of MEV extraction