Coalition Logic for Modelling and Verification of Smart Contract Upgrades

Rustam Galimullin[1] and Thomas Ågotnes[1,2]
{rustam.galimullin, thomas.agotnes}@uib.no
1: University of Bergen, Norway
2: Southwest University, China

Atomic swap

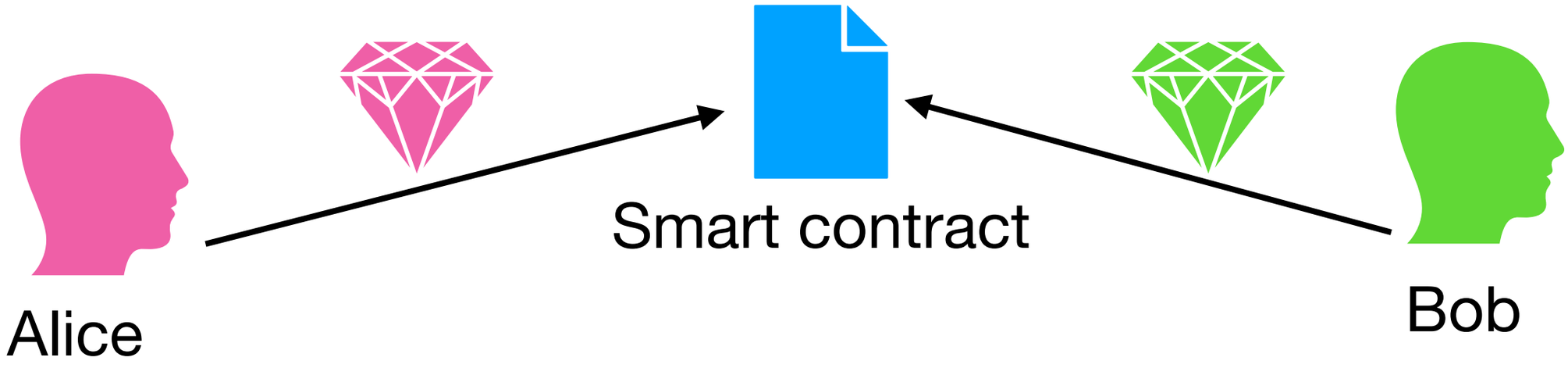
- A smart contract is a program deployed on a blockchain
- Atomic swap smart contract allows two agents to swap their assets without trusting each other

At the start, Alice and Bob hold their assets: *has(A,a)* and *has(B,b)*

Adapted from [van der Meyden, 2018]

Atomic swap

- A smart contract is a program deployed on a blockchain
- Atomic swap smart contract allows two agents to swap their assets without trusting each other



Then Alice and Bob deposit, simultaneously or one after another, their assets at the smart contract: *dep(A,a)* and *dep(B,b)*

Adapted from [van der Meyden, 2018]

Atomic swap

- A smart contract is a program deployed on a blockchain
- Atomic swap smart contract allows two agents to swap their assets without trusting each other

Then Alice and Bob deposit, simultaneously or one after another, their assets at the smart contract: *dep(A,a)* and *dep(B,b)*

Adapted from [van der Meyden, 2018]

Atomic swap

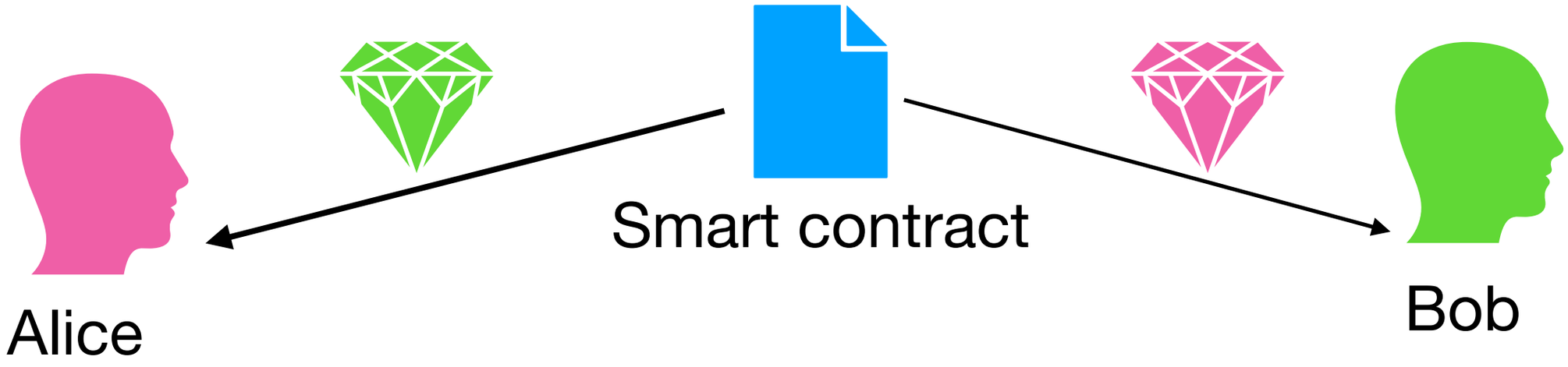
- A smart contract is a program deployed on a blockchain
- Atomic swap smart contract allows two agents to swap their assets without trusting each other

Finally, if (at least) one of the agents wants to finalise the swap, each one receives the asset of their partner

Adapted from [van der Meyden, 2018]

Atomic swap

- A smart contract is a program deployed on a blockchain
- Atomic swap smart contract allows two agents to swap their assets without trusting each other



Finally, if (at least) one of the agents wants to finalise the swap, each one receives the asset of their partner

Adapted from [van der Meyden, 2018]

Coalition logic

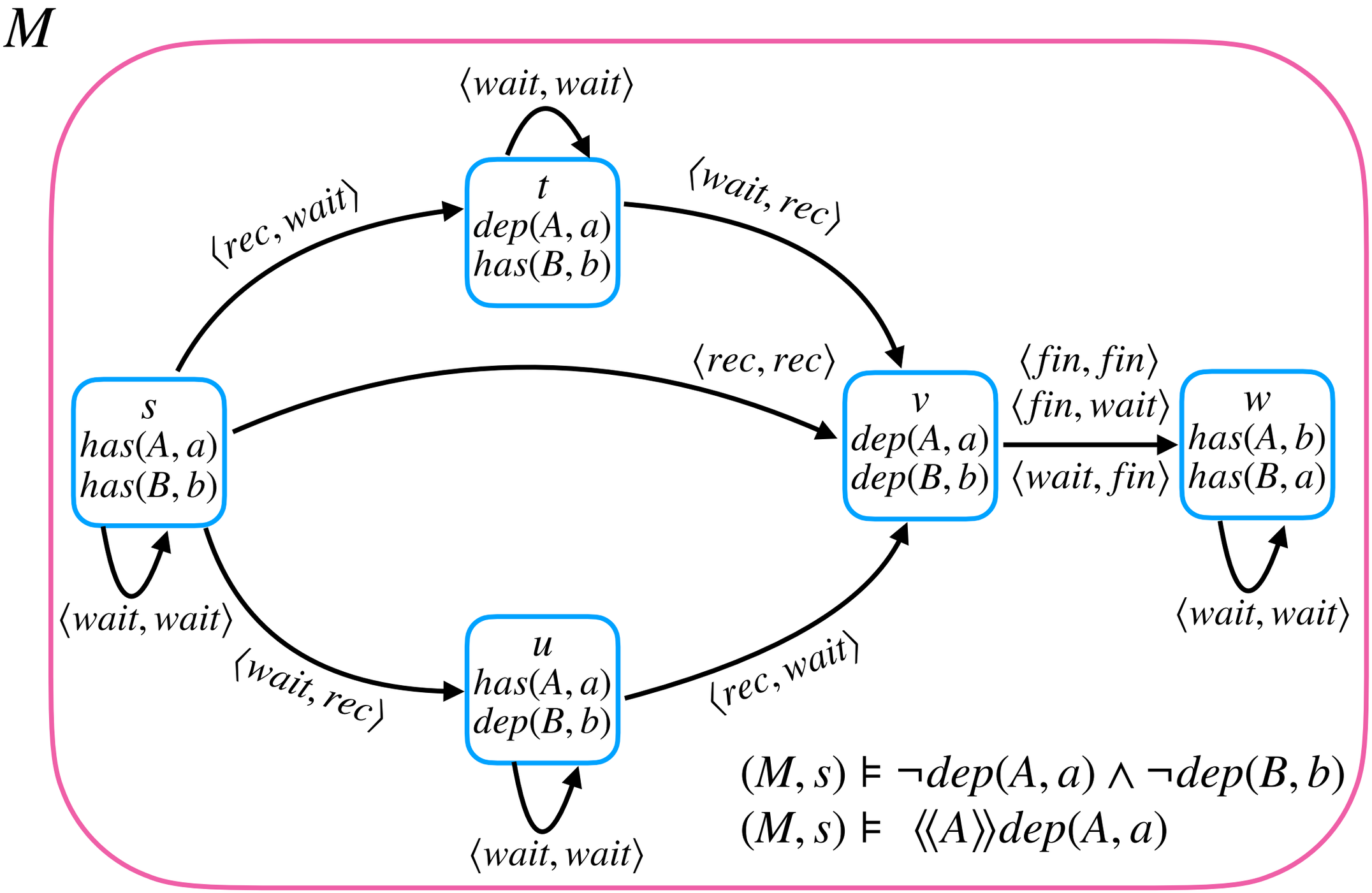
- Coalition logic (CL) [Pauly, 2002] is used to reason about abilities of groups of agents in the presence of opponents

- Language of CL: $\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle\langle C \rangle\rangle \varphi$

- $\langle\langle C \rangle\rangle \varphi$ is read as ‘coalition C can bring about φ by a joint action no matter what agents outside of the coalition do.’

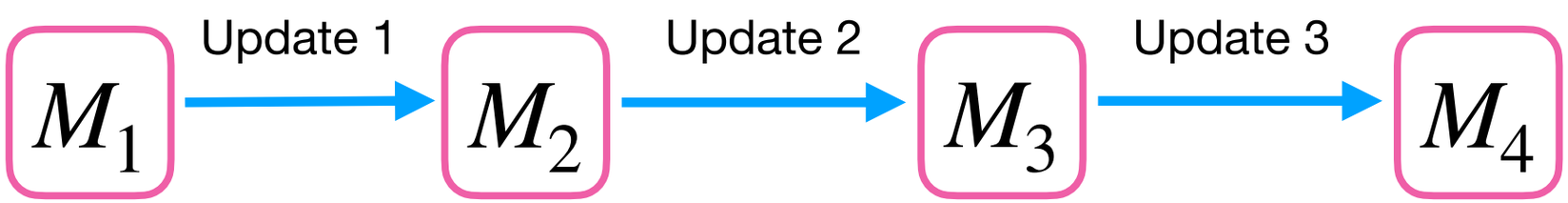
- Dual $[\![C]\!] \varphi$ is read as ‘coalition C cannot avoid φ’

Atomic swap


M
⟨wait, wait⟩
t
dep(A, a)
has(B, b)
⟨rec, wait⟩
⟨wait, rec⟩
⟨rec, rec⟩
s
has(A, a)
has(B, b)
v
dep(A, a)
dep(B, b)
⟨fin, fin⟩
⟨fin, wait⟩
⟨wait, fin⟩
w
has(A, b)
has(B, a)
⟨wait, wait⟩
⟨wait, wait⟩
⟨wait, rec⟩
u
has(A, a)
dep(B, b)
⟨rec, wait⟩
⟨wait, wait⟩
(M, s) ⊨ ¬dep(A, a) ∧ ¬dep(B, b)
(M, s) ⊨ ⟨⟨A⟩⟩dep(A, a)

Upgrades of smart contracts

- Models of CL can capture some properties of smart contracts
- CL, however, cannot capture upgrades of such contracts
- Moreover, once an upgrade is deployed on a blockchain (an old version of) smart contract may still be available
- Thus, we propose using dynamic coalition logic (DCL)



We use a recently introduced dictatorial DCL (DDCL) that allows granting agents dictatorial powers in certain states

Atomic swap upgraded

- Assume that we want to upgrade atomic swap in the following way

- Any agent can cancel the swap before it has happened, i.e. before finalisation

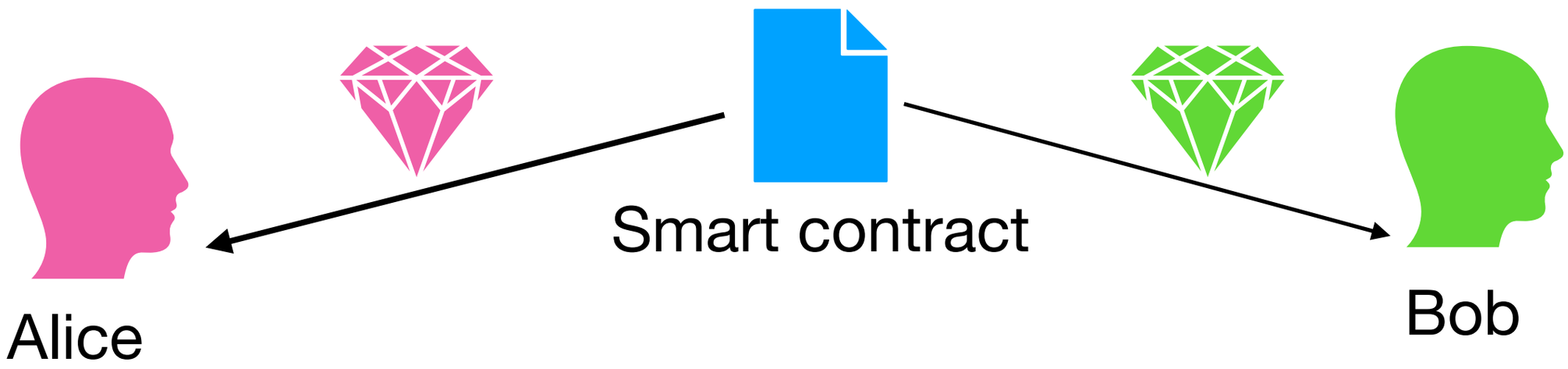
If Alice cancels the swap, the assets are returned to their owners

Adapted from [van der Meyden, 2018]

Atomic swap upgraded

- Assume that we want to upgrade atomic swap in the following way

- Any agent can cancel the swap before it has happened, i.e. before finalisation

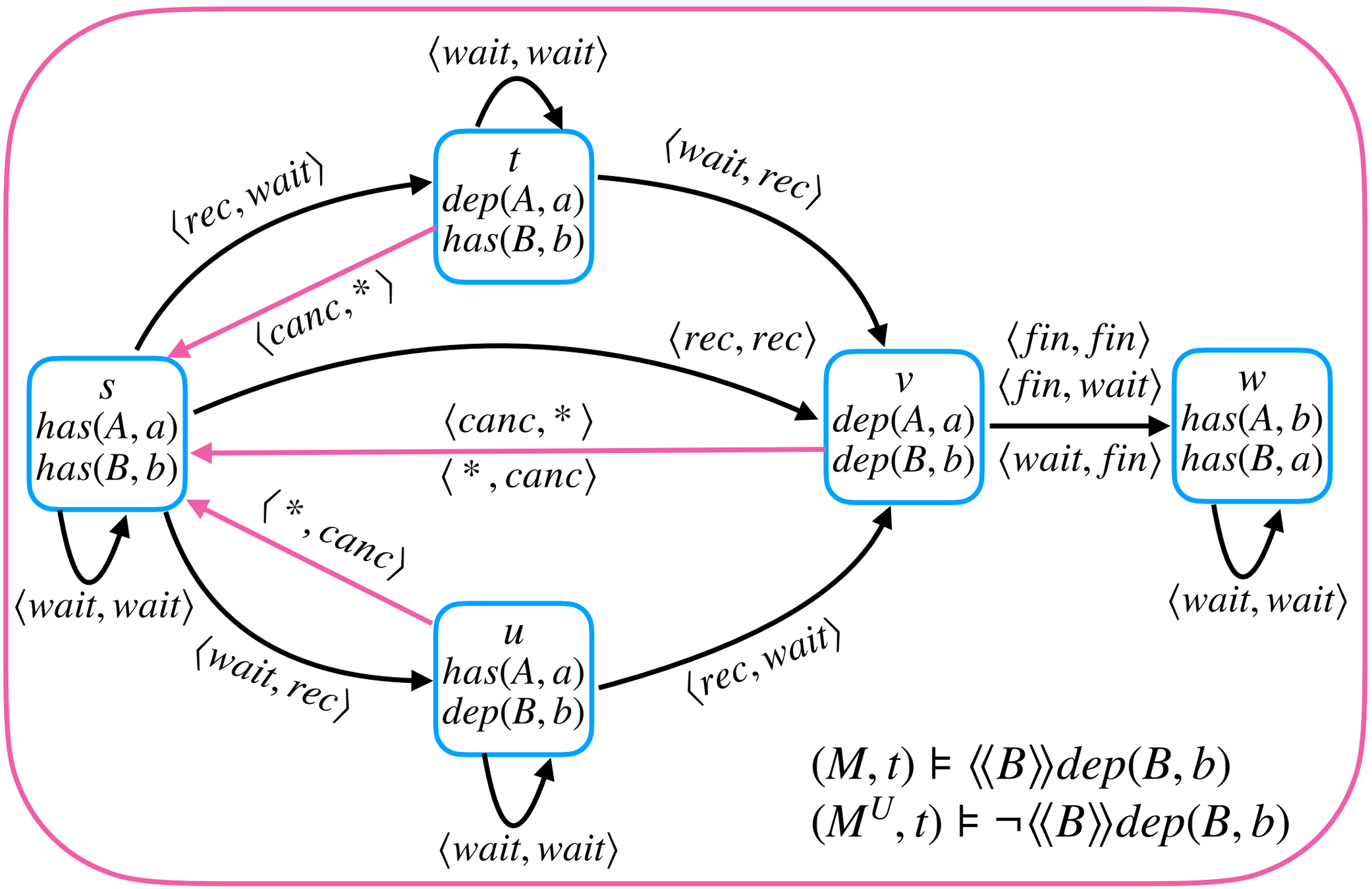


If Alice cancels the swap, the assets are returned to their owners

Adapted from [van der Meyden, 2018]

Atomic swap upgraded

M^U



Temporal DDCL

- Finally, to capture the fact that older versions of smart contracts may remain on a blockchain we add a temporal backwards-looking relation
- This relation also allows for introspection: reasoning about current version of a smart contract based on its previous versions

$$(\mathscr{C}, M^U, s) \models \varphi \wedge \Diamond \neg \varphi$$

Results and open questions

We proposed a use of Temporal DDCL for smart contract upgrades

We studied some properties of models for Temporal DDCL

We argued that the complexity of the model checking problem for Temporal DDCL is *P-complete*

?More expressive base logic than CL (e.g. ATL or SL)?

?More expressive model updates?

?Proof system?