

Lightning Network Analysis and Recursive Virtual Channels

Orfeas Stefanos Thyfronitis Litos
University of Edinburgh
8/2/2021

VISA

20,000 tx/s

 **bitcoin**

7 tx/s

Problem

All txs validated by all wallets

Solution

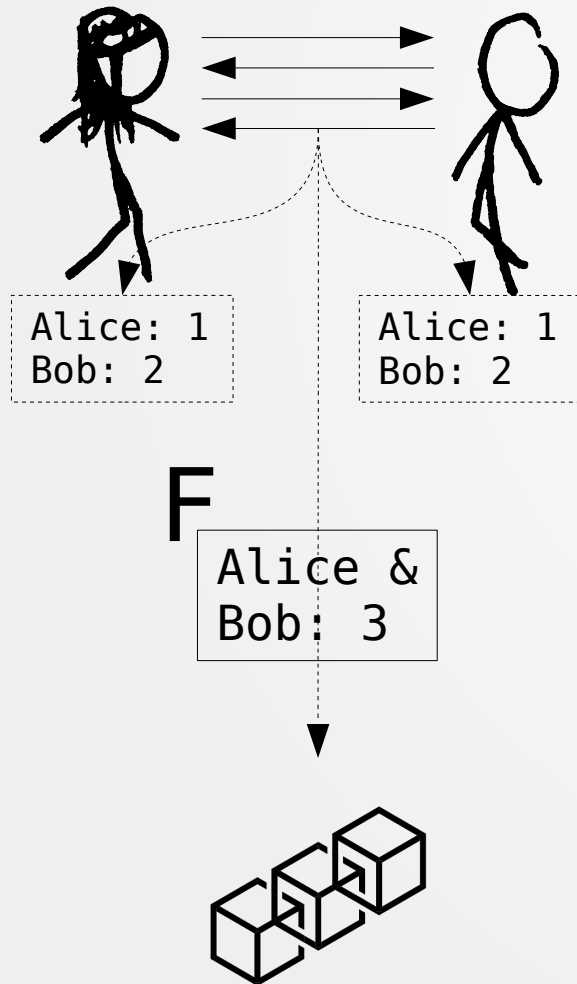
- Move most txs off-chain
- Resolve disputes on-chain

Part 1

A Composable Security
Treatment of the
Lightning Network
[CSF 2020]

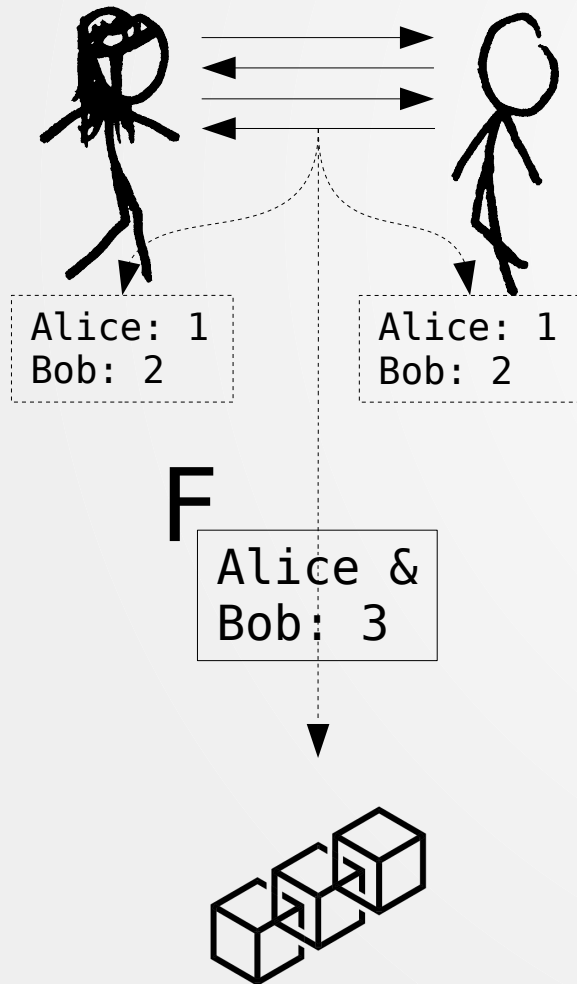
Lightning Channels

Open

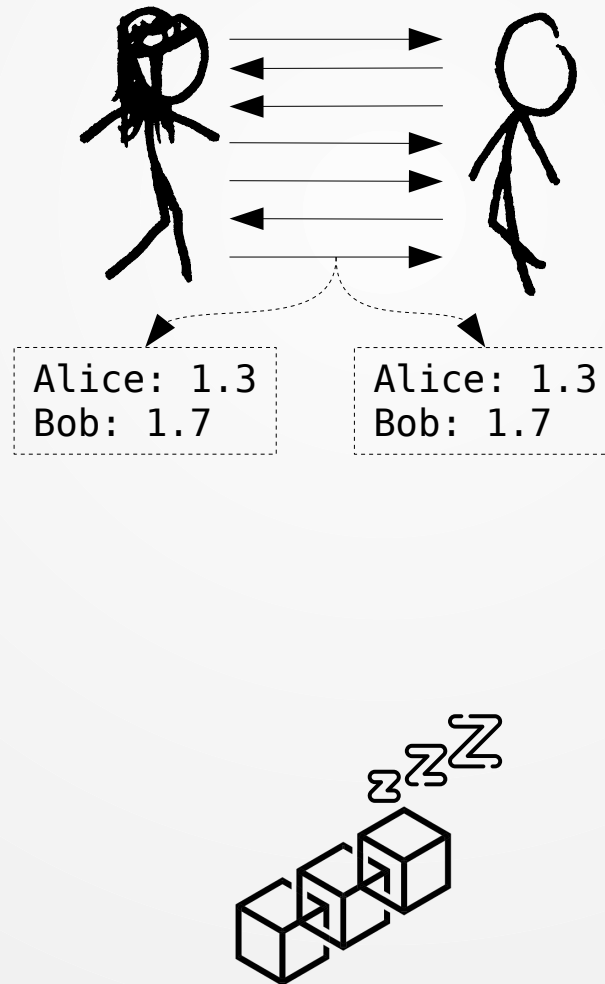


Lightning Channels

Open

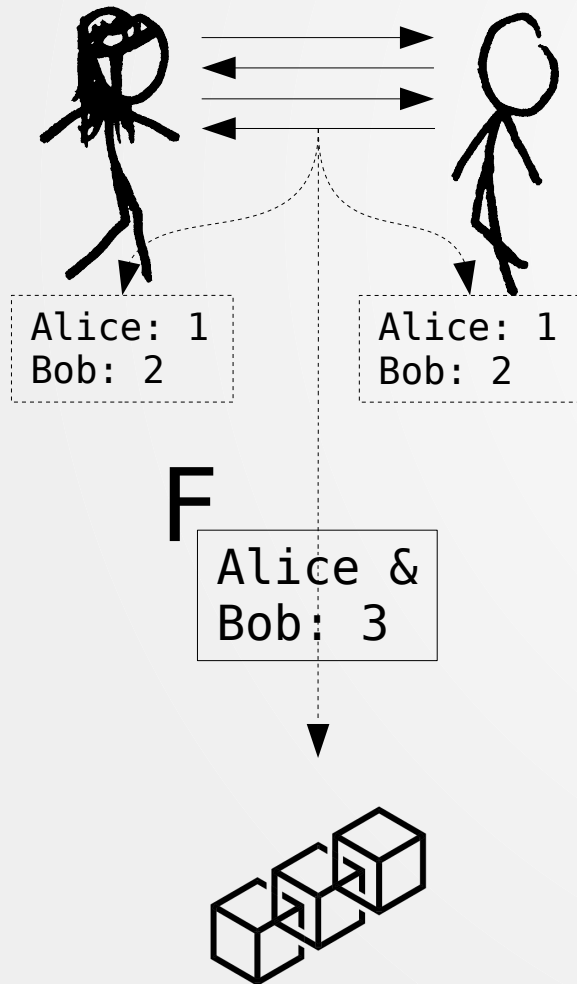


Pay

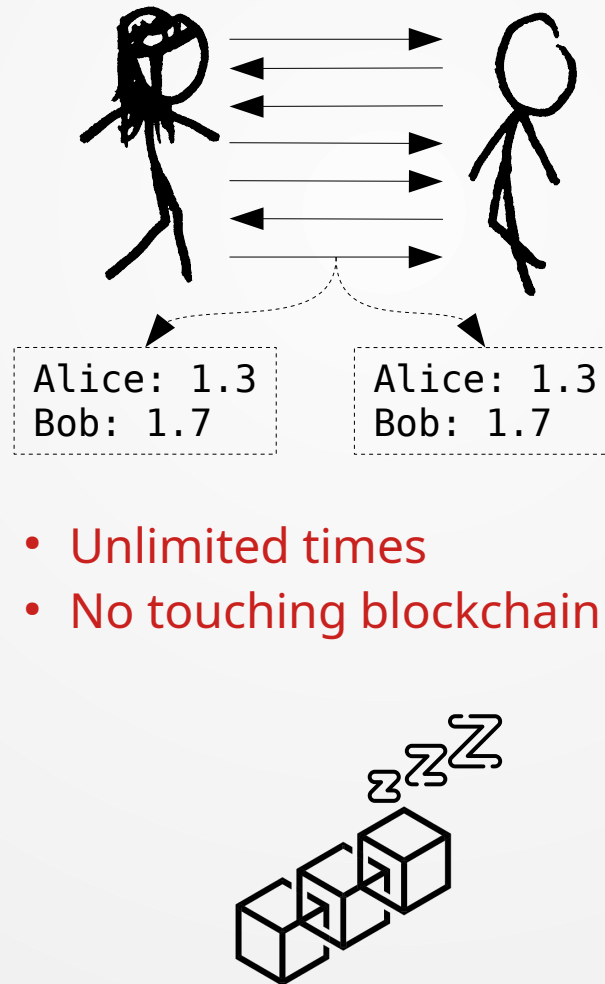


Lightning Channels

Open

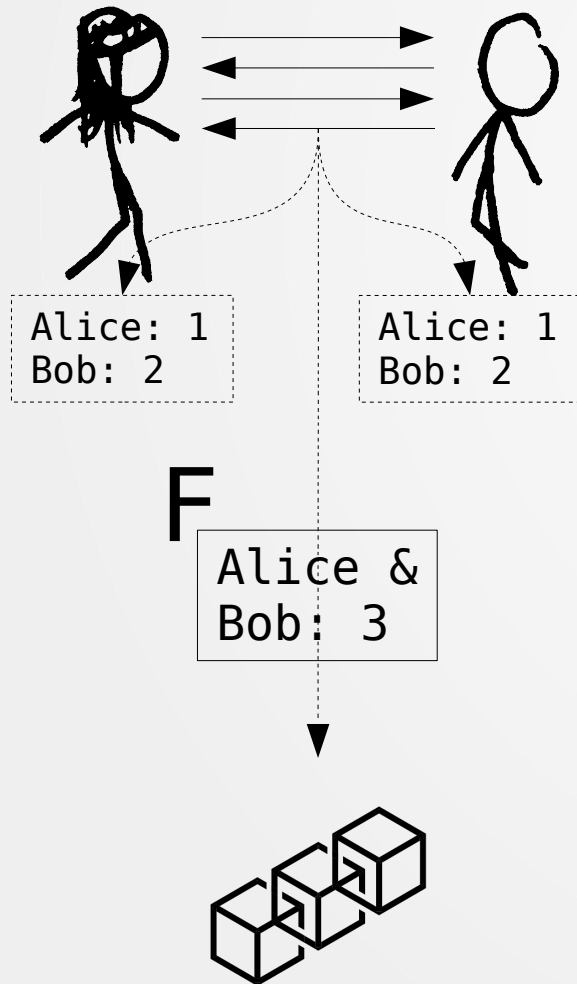


Pay

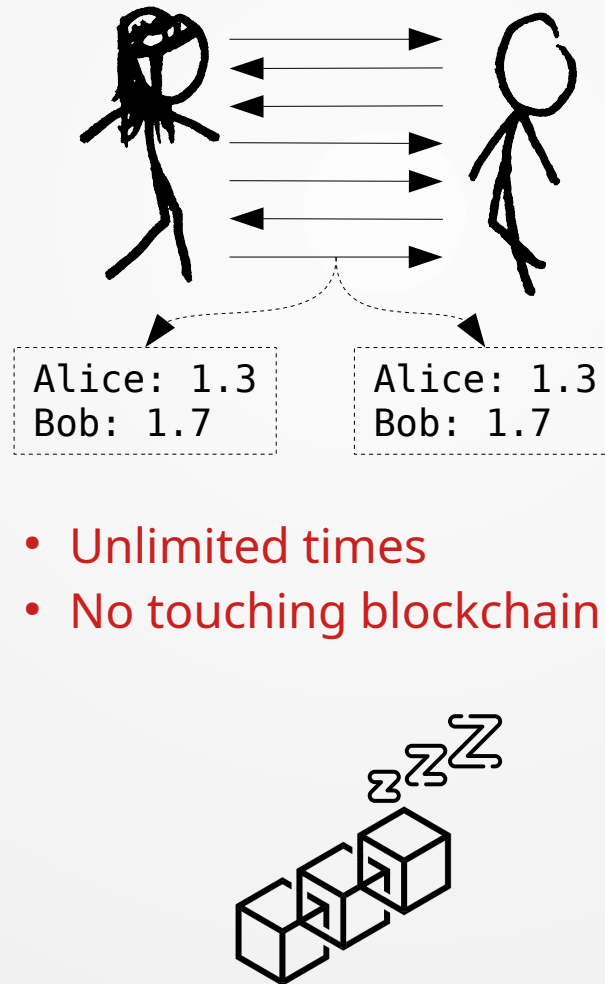


Lightning Channels

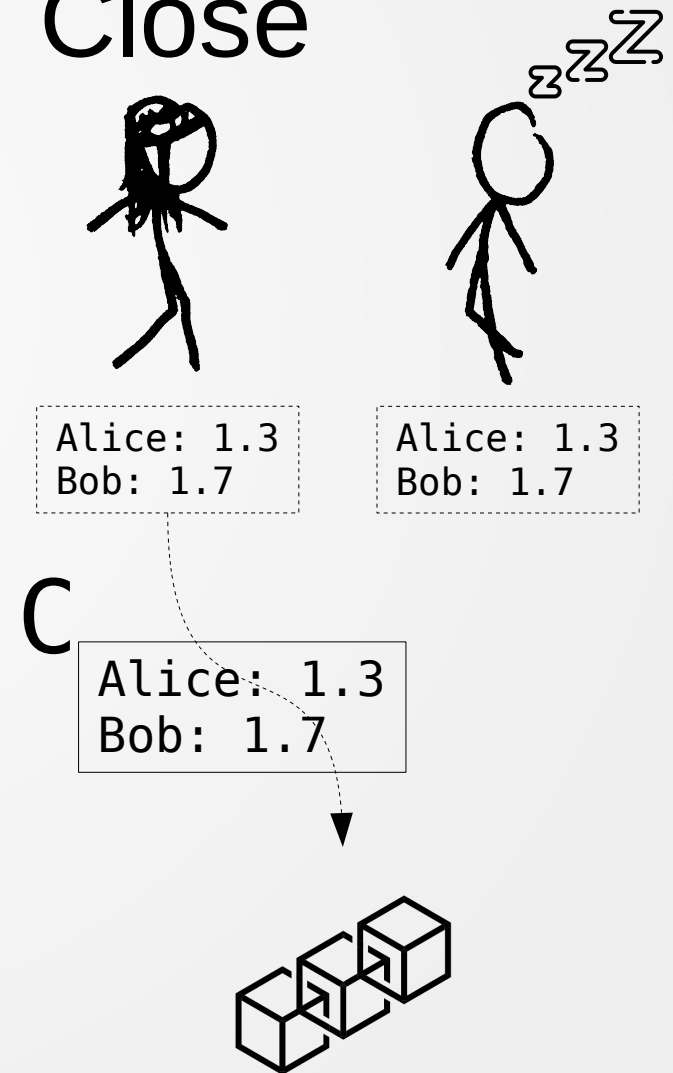
Open



Pay

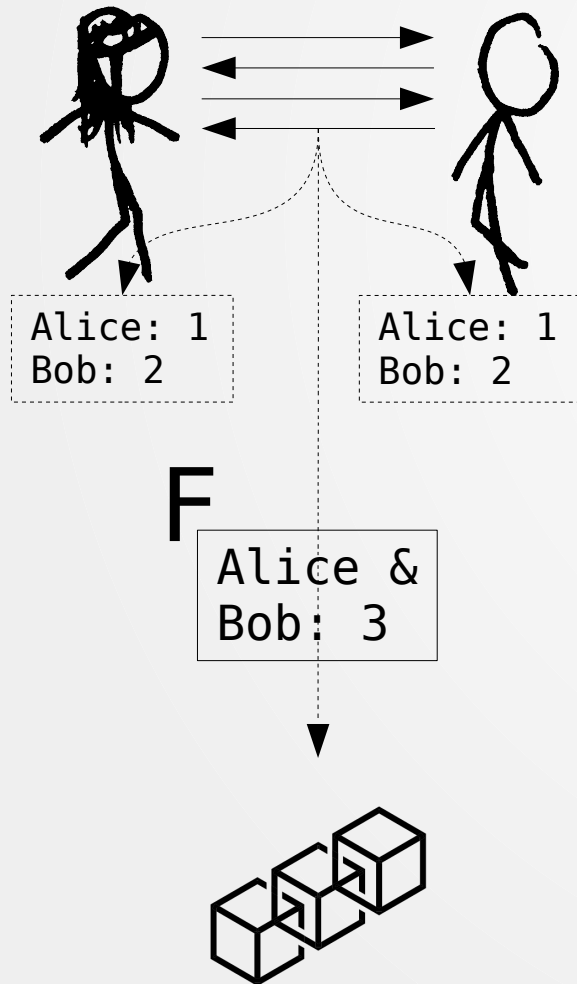


Close

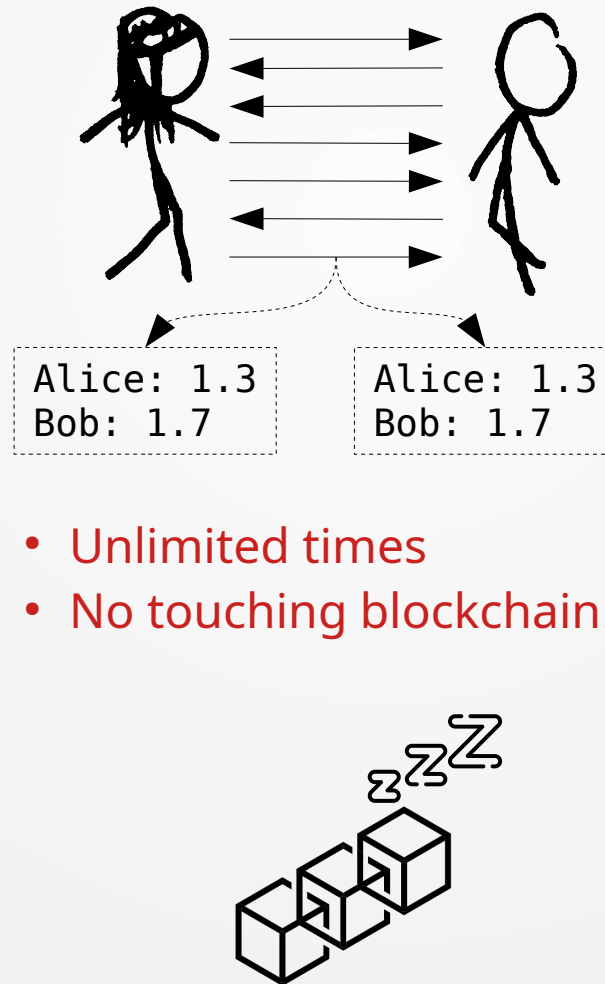


Lightning Channels

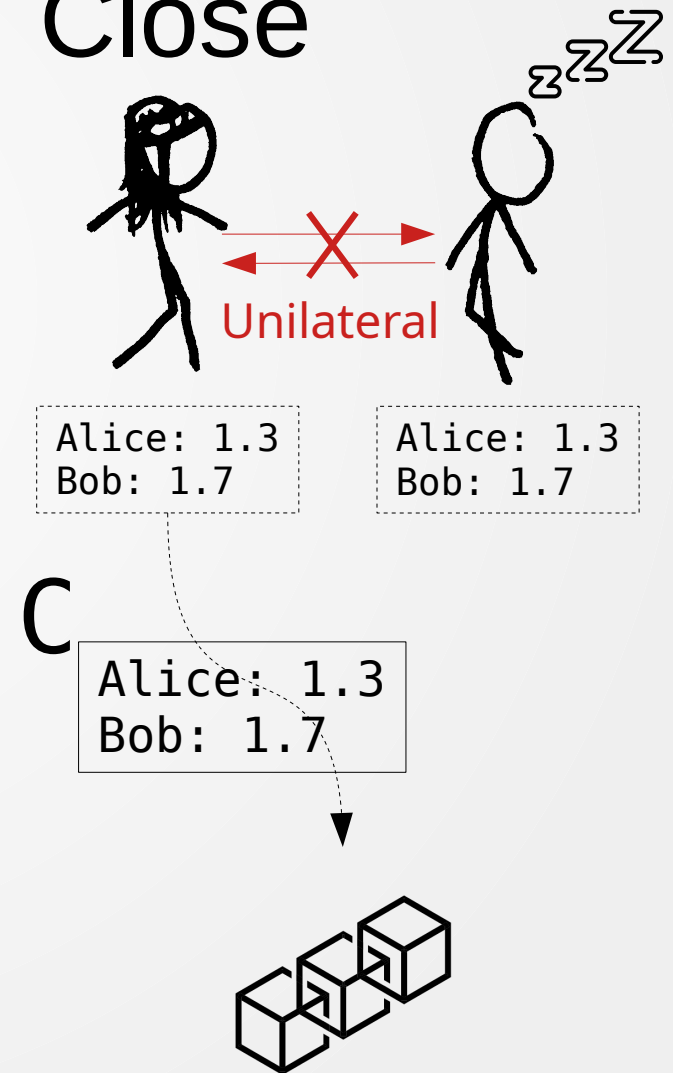
Open



Pay



Close



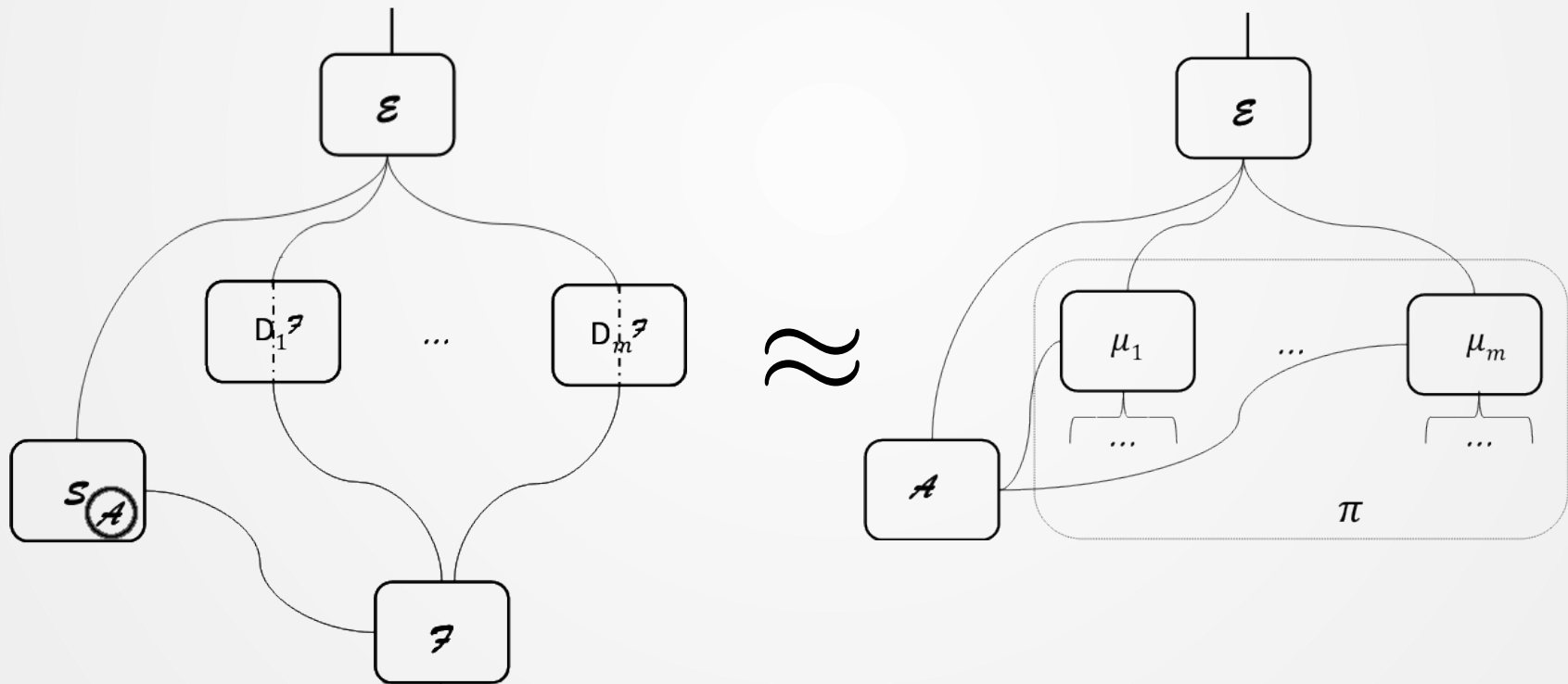
Multi-hop payments



From channels
to network!

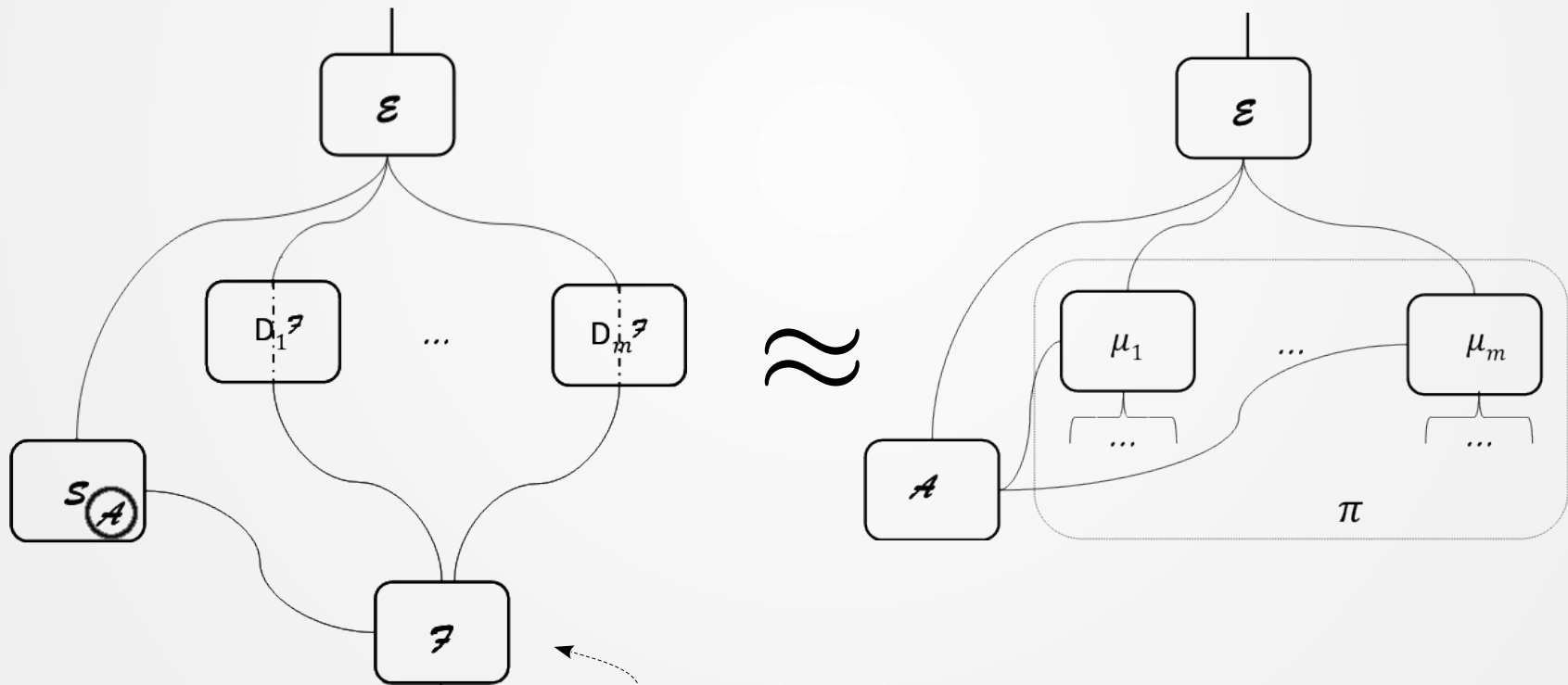
Simulation-based Security

$$\forall \mathcal{A} \exists \mathcal{S} : \forall \text{PPT } \mathcal{E}$$



Our paper

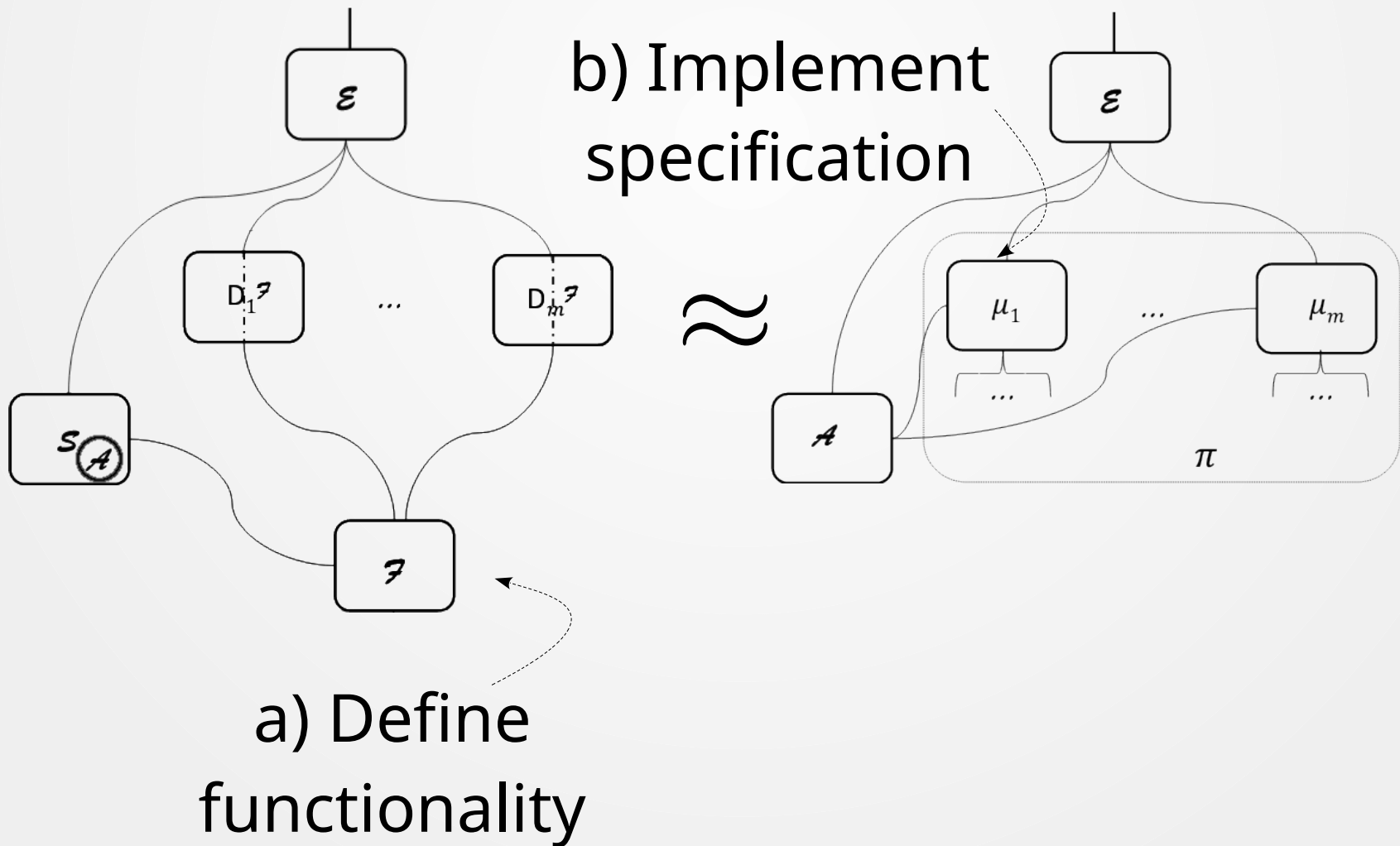
$$\forall \mathcal{A} \in \mathcal{S} : \forall \text{PPT } \mathcal{E}$$



a) Define
functionality

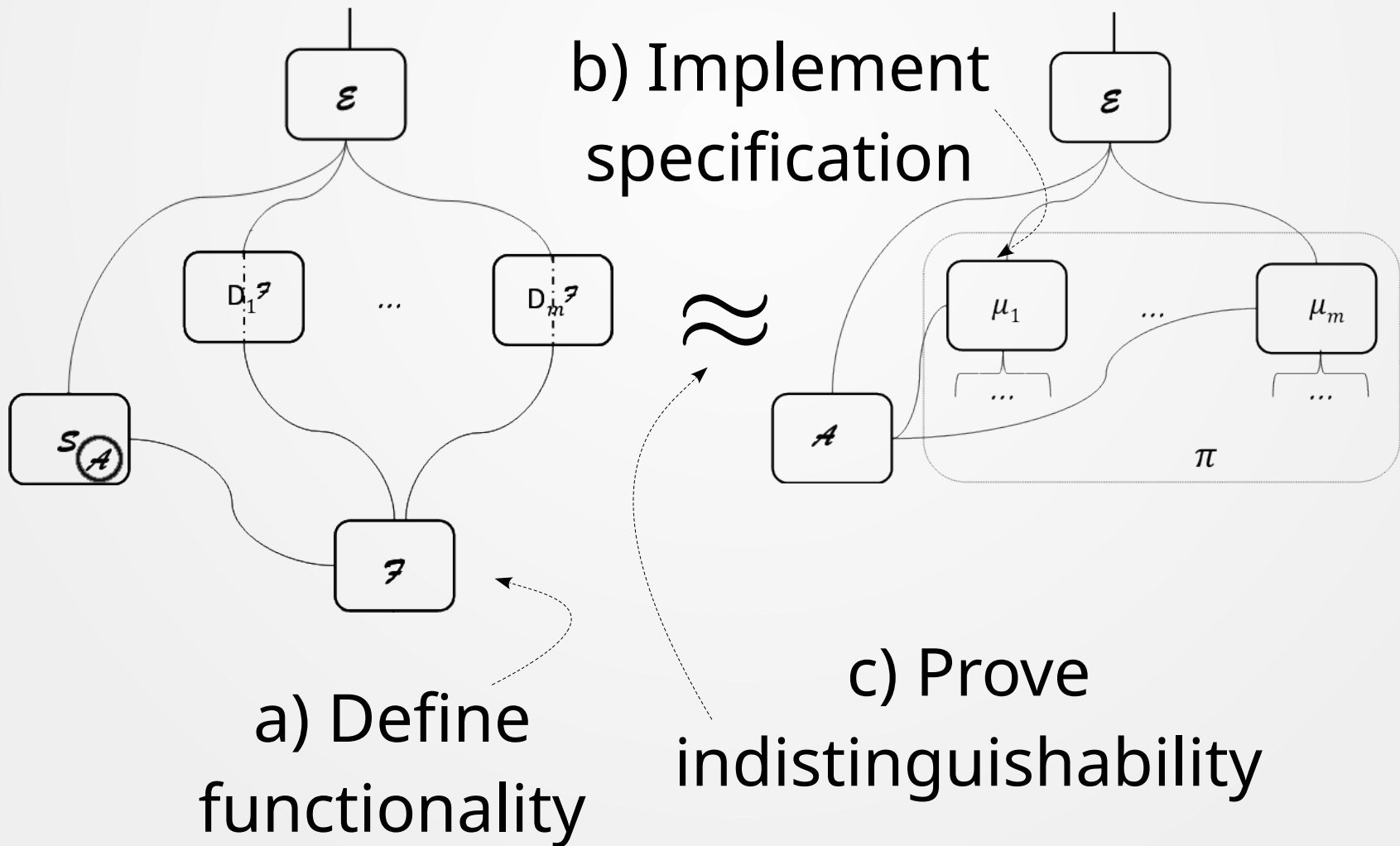
Our paper

$$\forall \mathcal{A} \exists \mathcal{S} : \forall \text{PPT } \mathcal{E}$$



Our paper

$$\forall \mathcal{A} \exists \mathcal{S} : \forall \text{PPT } \mathcal{E}$$



Blockchain Functionality

$\mathcal{G}_{\text{ledger}}$ [BMTZ'17, BGKRZ'18]

Functionality

Functionality $\mathcal{F}_{\text{PayNet}}$ – interface

- from \mathcal{E} :
 - (REGISTER, delay, relayDelay)
 - (TOPPEDUP)
 - (OPENCHANNEL, *Alice*, *Bob*, *x*, *tid*)
 - (CHECKFORNEW, *Alice*, *Bob*, *tid*)
 - (PAY, *Bob*, *x*, $\overrightarrow{\text{path}}$, receipt)
 - (CLOSECHANNEL, receipt, *pchid*)
 - (FORCECLOSECHANNEL, receipt, *pchid*)
 - (POLL)
 - (PUSHFULFILL, *pchid*)
 - (PUSHADD, *pchid*)
 - (COMMIT, *pchid*)
 - (FULFILLONCHAIN)
 - (GETNEWS)
- to \mathcal{E} :
 - (REGISTER, *Alice*, delay(*Alice*), relayDelay(*Alice*), pubKey)
 - (REGISTERED)
 - (NEWS, newChannels, closedChannels, updatesToReport)
- from \mathcal{S} :
 - (REGISTERDONE, *Alice*, pubKey)
 - (CORRUPTED, *Alice*)
 - (CHANNELANNOUNCED, *Alice*, $p_{\text{Alice},F}$, $p_{\text{Bob},F}$, *fchid*, *pchid*, *tid*)
 - (UPDATE, receipt, *Alice*)
 - (CLOSECHANNEL, channel, *Alice*)
 - (RESOLVEPAYS, *payid*, charged)
- to \mathcal{S} :
 - (REGISTER, *Alice*, delay, relayDelay)
 - (OPENCHANNEL, *Alice*, *Bob*, *x*, *fchid*, *tid*)
 - (CHANNELOPENED, *Alice*, *fchid*)
 - (PAY, *Alice*, *Bob*, *x*, $\overrightarrow{\text{path}}$, receipt, *payid*)
 - (CONTINUE)
 - (CLOSECHANNEL, *fchid*, *Alice*)
 - (FORCECLOSECHANNEL, *fchid*, *Alice*)
 - (POLL, Σ_{Alice} , *Alice*)
 - (PUSHFULFILL, *pchid*, *Alice*)
 - (PUSHADD, *pchid*, *Alice*)
 - (COMMIT, *pchid*, *Alice*)
 - (FULFILLONCHAIN, *t*, *Alice*)

How often online?

- No in-flight payments
 - sync at least every `to_self_delay` blocks
- In-flight HTLC – intermediary
 - $a = \text{“max new blocks from tx bcast till settled”}$
 - Sync during `[out_cltv_exp, in_cltv_exp - 2a]`
 - try to publish HTLC-timeout
 - Sync again after a
 - If HTLC-success found, update or fulfill on-chain
- In-flight HTLC – payee
 - Fulfill on-chain until `min_final_cltv_expiry - a`

Our contributions

- Use a realistic ledger functionality
 - Prove naive ledger unrealizable
- Prove Lightning Network security in UC framework
- Derive exact time bounds for how often parties need to check the chain

Part 2

Recursive Virtual Payment Channels for Bitcoin

Features

- Enables long-lived “virtual” channels w/o an on-chain funding TX
- Virtual channels built on top of a path of preexisting “base” channels – left endpoint funds the channel
- Base channels may themselves be virtual!
- Virtual channels can leverage a path of many base channels

Construction

Intermediary i has 3 classes of TXs:

- “Initiator” TX
 - Spends left & right funding outputs
 - Has virtual output with interval $[i]$

Construction

Intermediary i has 3 classes of TXs:

- “Initiator” TX
 - Spends left & right funding outputs
 - Has virtual output with interval $[i]$
- “Extend-interval” TXs
 - Spends 1 funding output and 1 virtual output with interval $[j, \dots, i-1]$ or $[i+1, \dots, j]$
 - Has virtual output w/ interval $[j, \dots, i]$ or $[i, \dots, j]$

Construction

Intermediary i has 3 classes of TXs:

- “Initiator” TX
 - Spends left & right funding outputs
 - Has virtual output with interval $[i]$
- “Extend-interval” TXs
 - Spends 1 funding output and 1 virtual output with interval $[j, \dots, i-1]$ or $[i+1, \dots, j]$
 - Has virtual output w/ interval $[j, \dots, i]$ or $[i, \dots, j]$
- “Merge-intervals” TXs
 - Spends 2 virtual outputs with intervals $[j, \dots, i-1]$ and $[i+1, \dots, k]$
 - Has virtual output with interval $[j, \dots, k]$

Example 1

Fundee wants to close

$t=0$

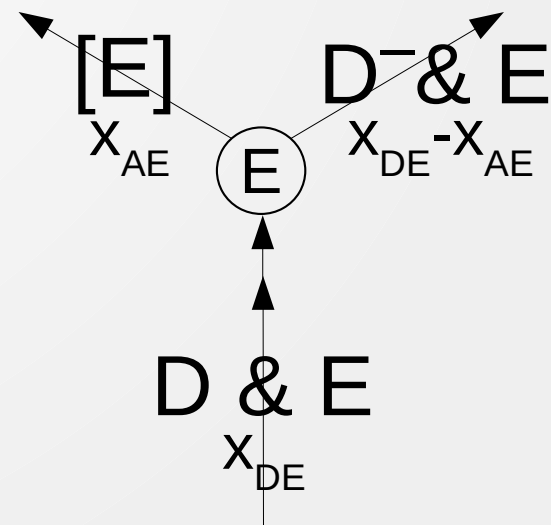
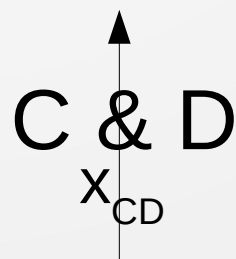
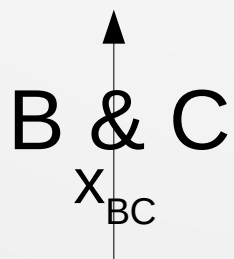
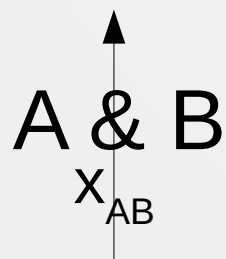
A & B
 x_{AB}

B & C
 x_{BC}

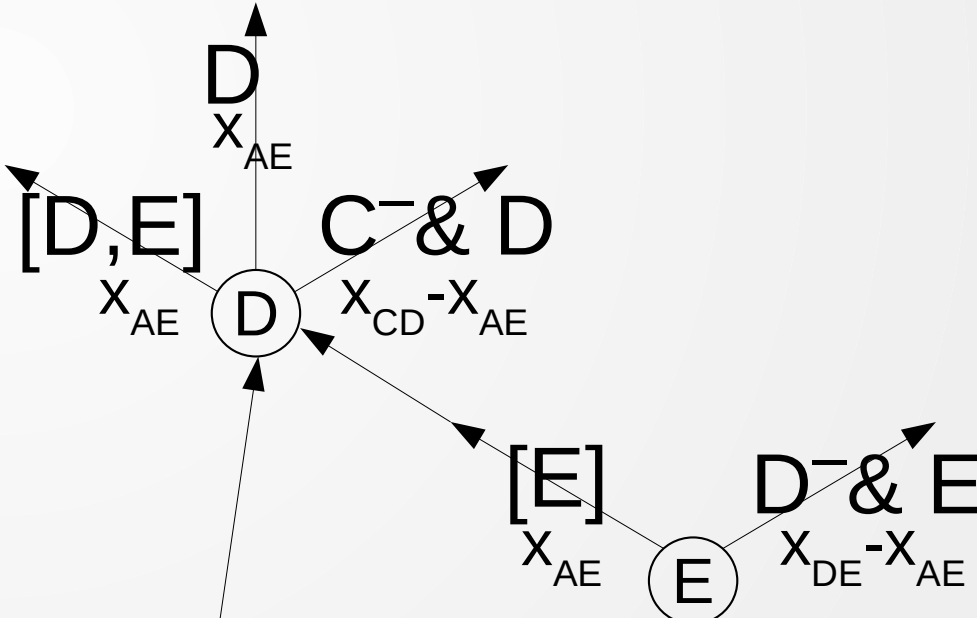
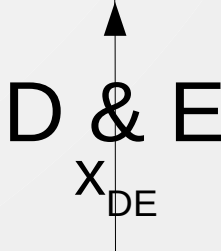
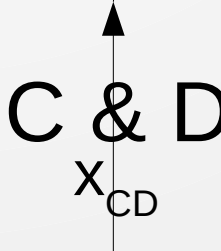
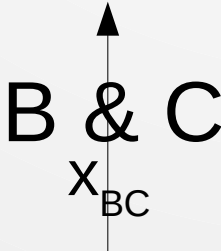
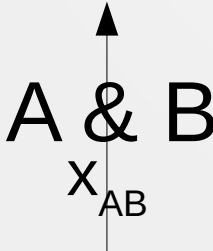
C & D
 x_{CD}

D & E
 x_{DE}

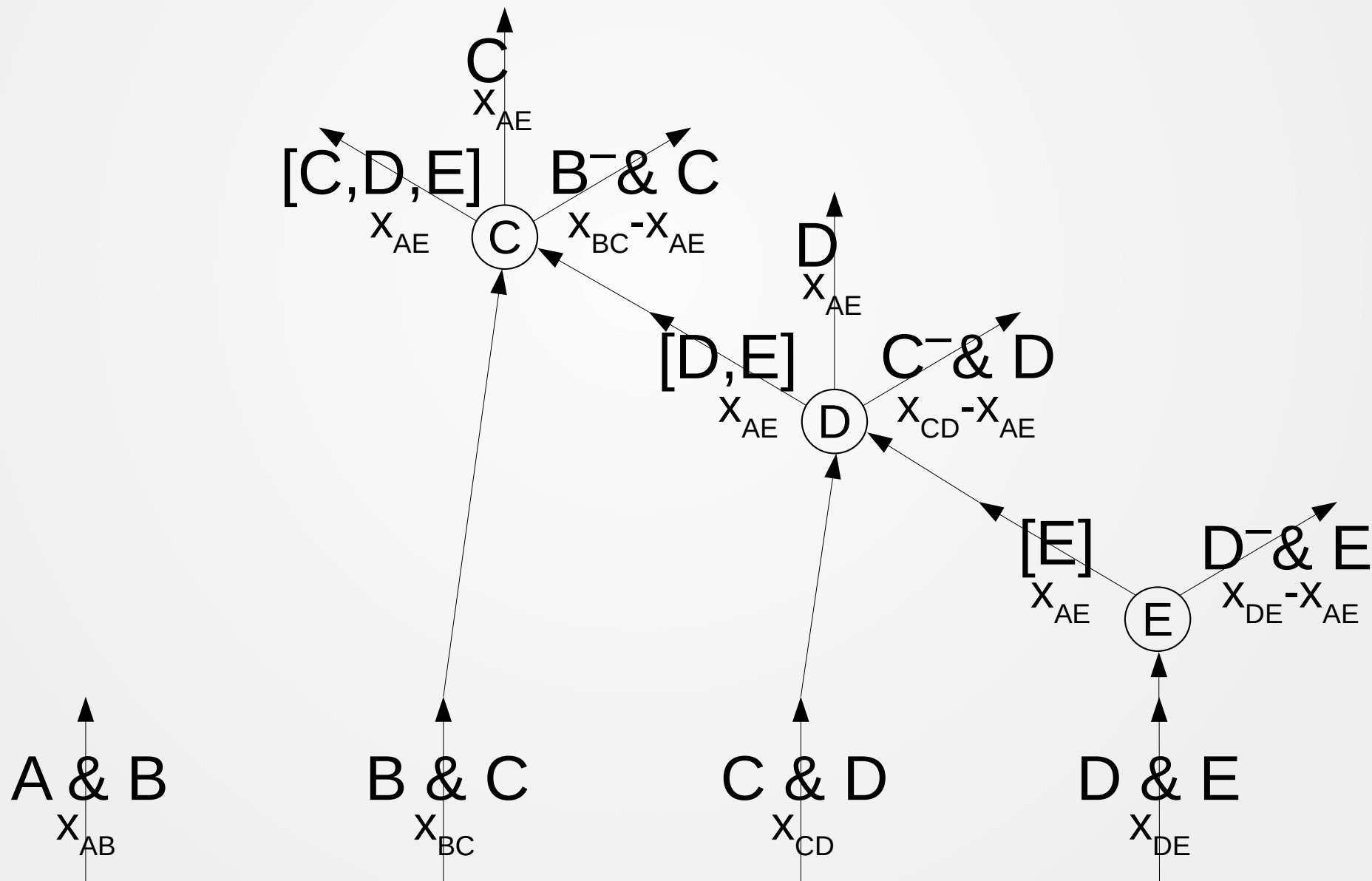
t=1



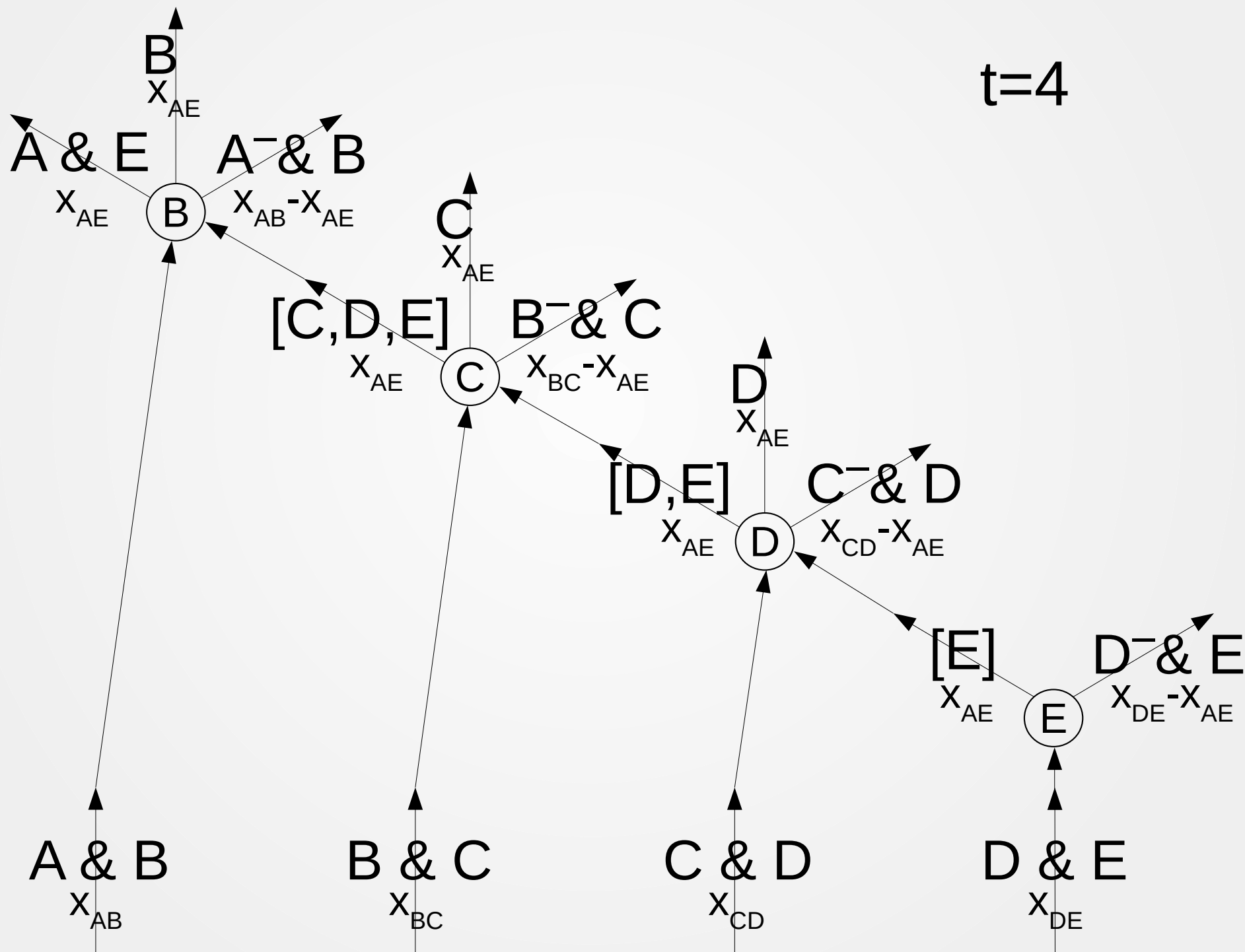
t=2



t=3



t=4



Example 2

Simultaneous initiators

t=0

A & B
x_{AB}

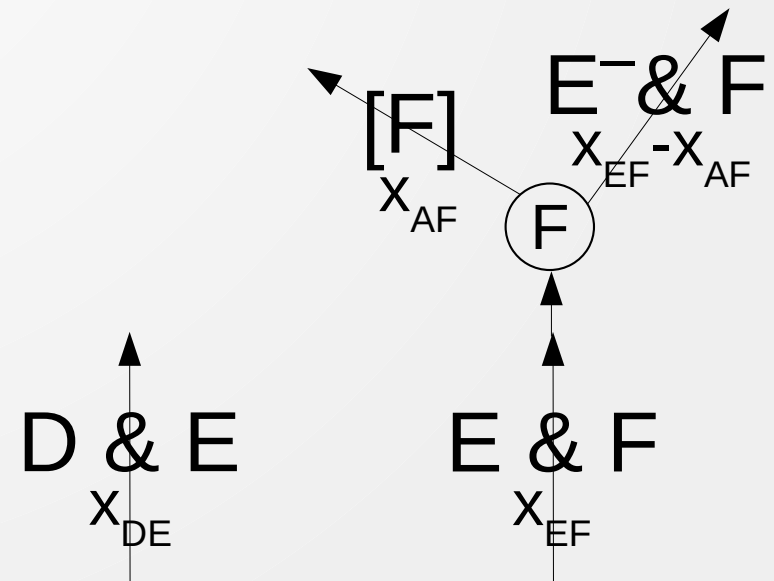
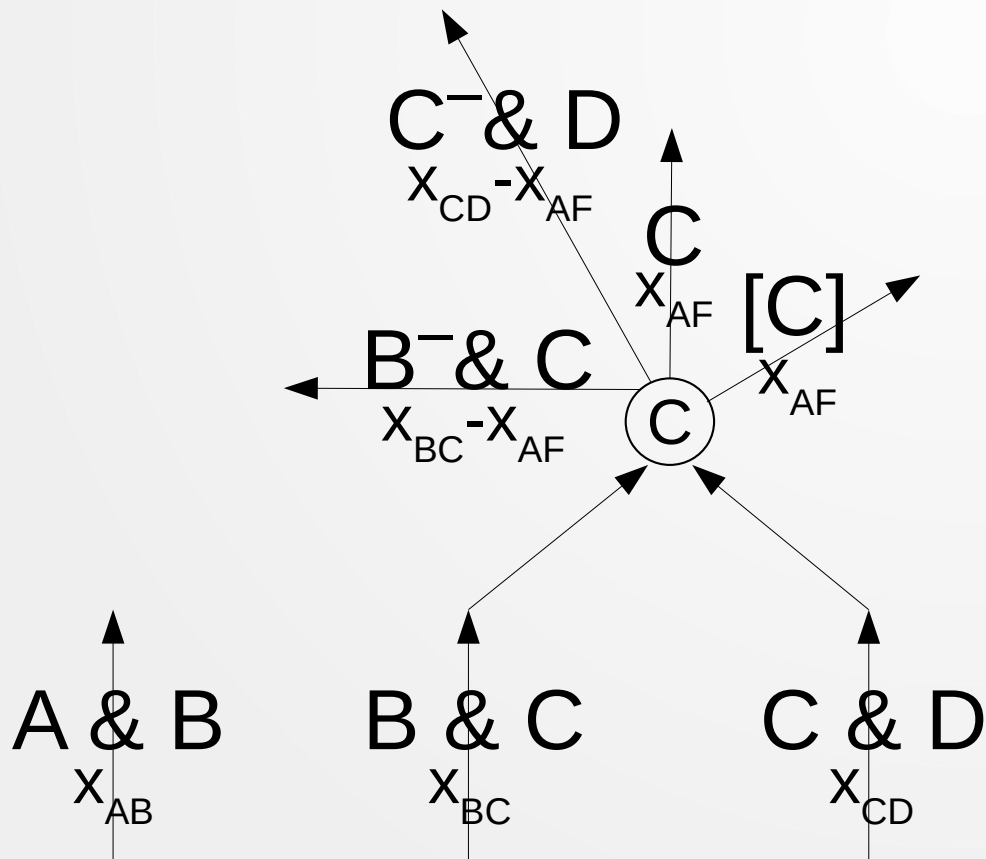
B & C
x_{BC}

C & D
x_{CD}

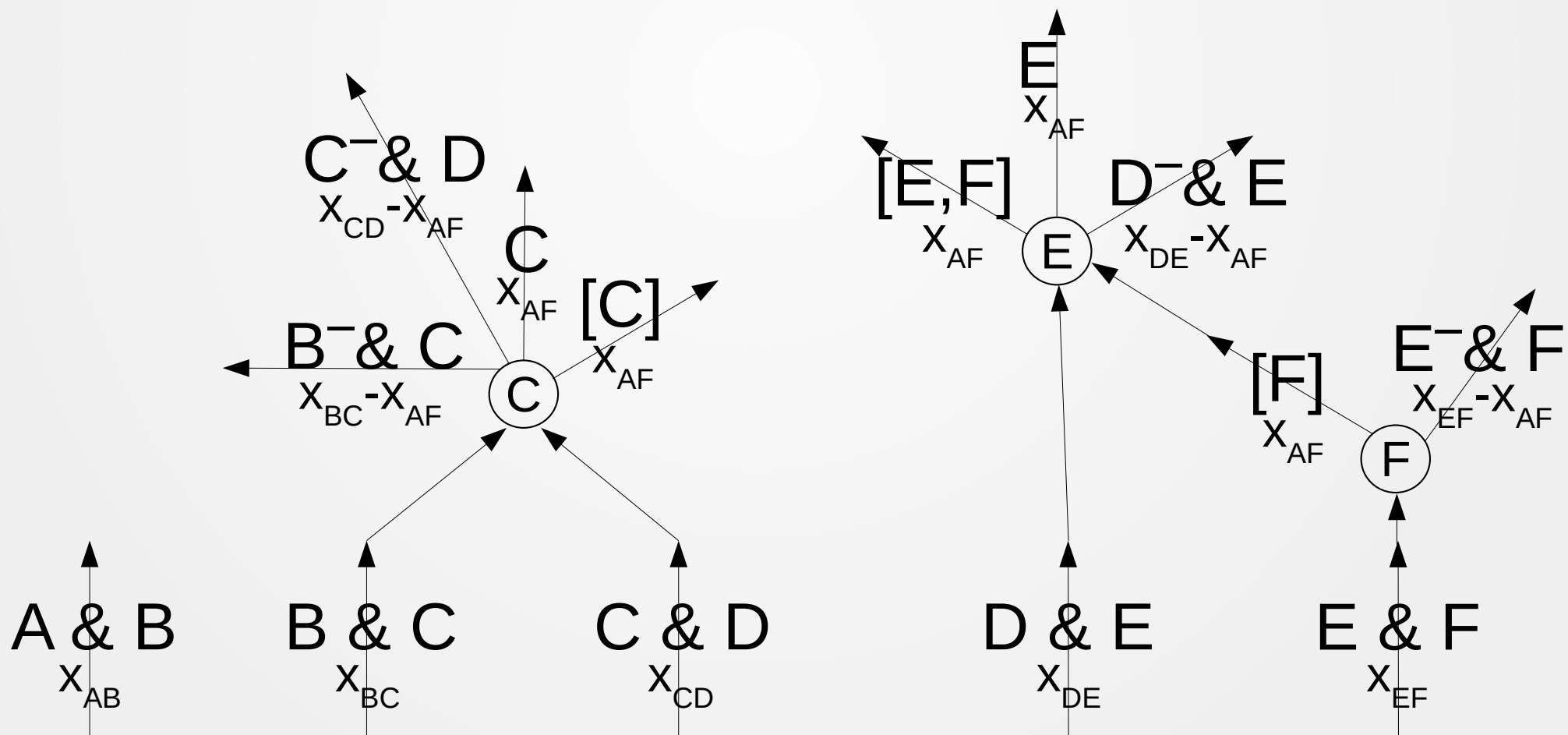
D & E
x_{DE}

E & F
x_{EF}

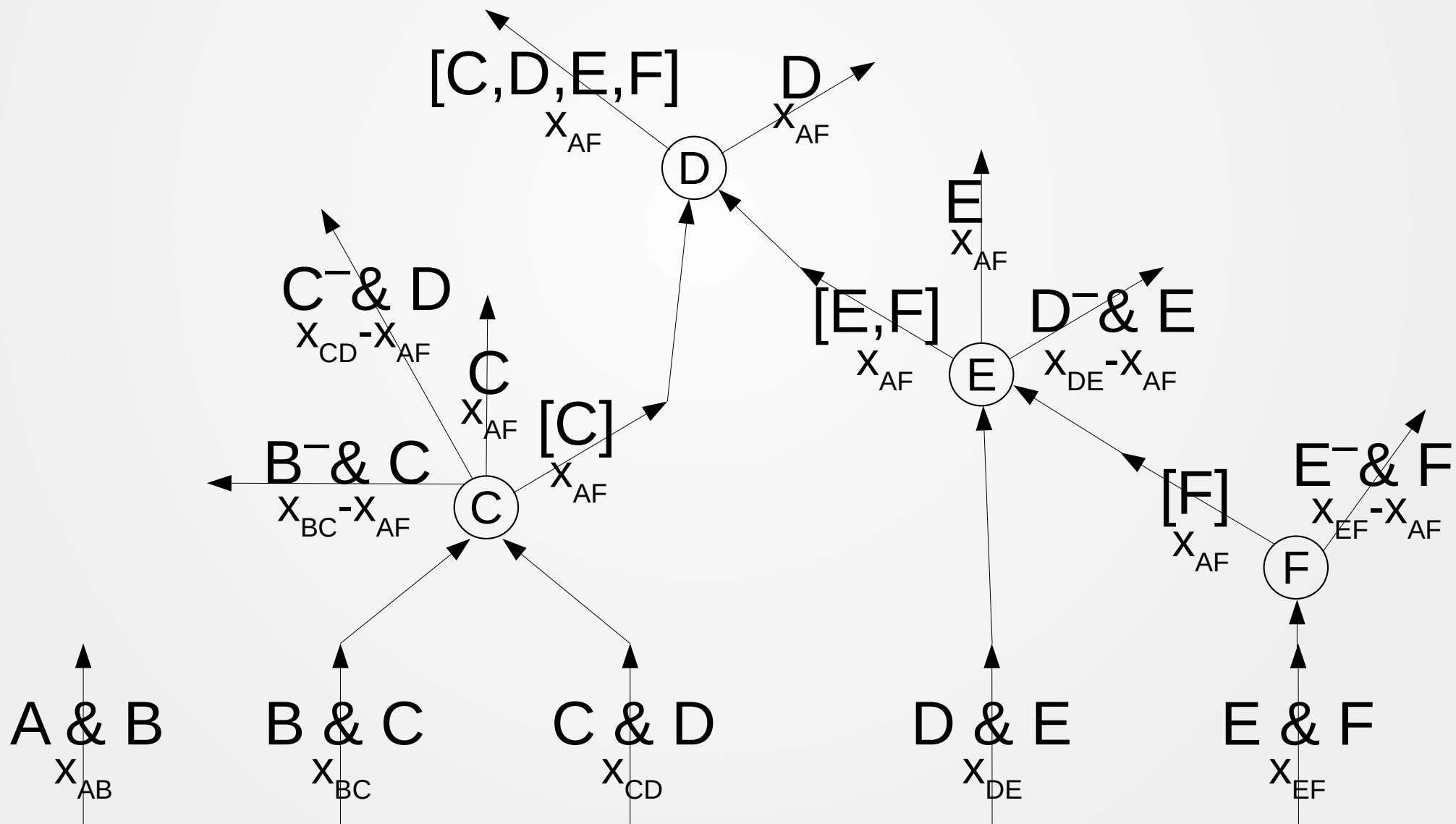
t=1



t=2



t=3



Example 3

Virtual base channel

t=0

A & C
 x_{AC}

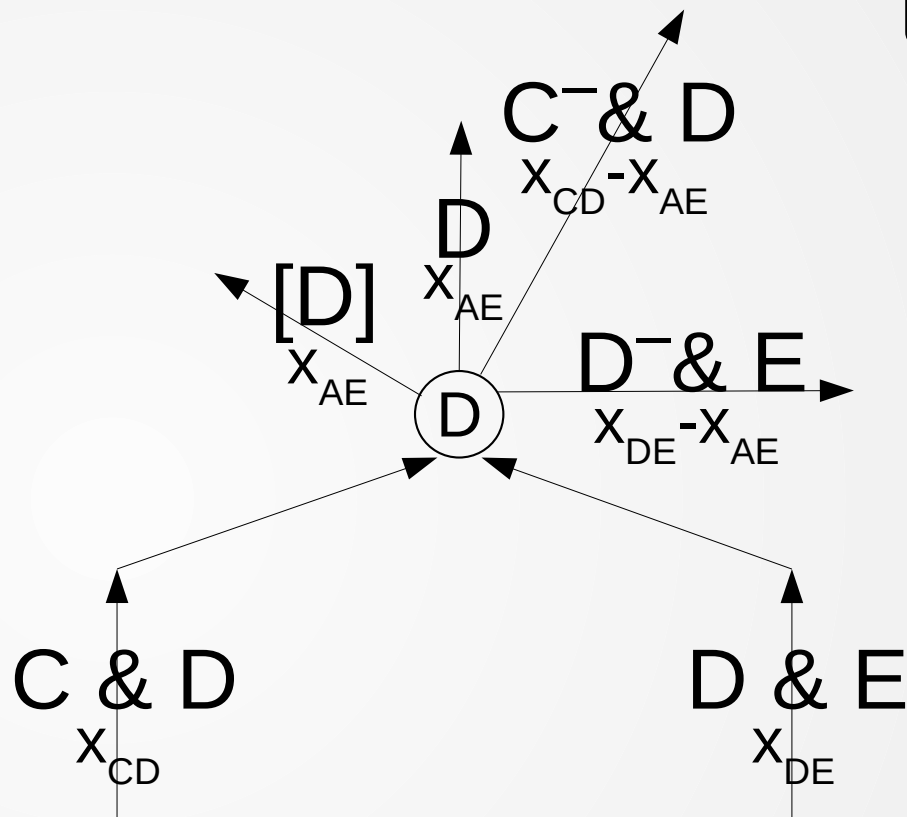
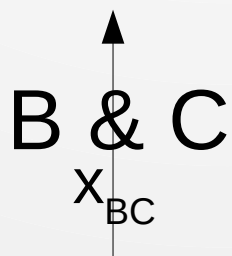
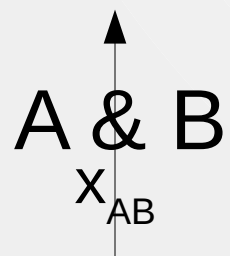
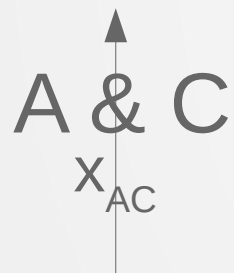
C & D
 x_{CD}

D & E
 x_{DE}

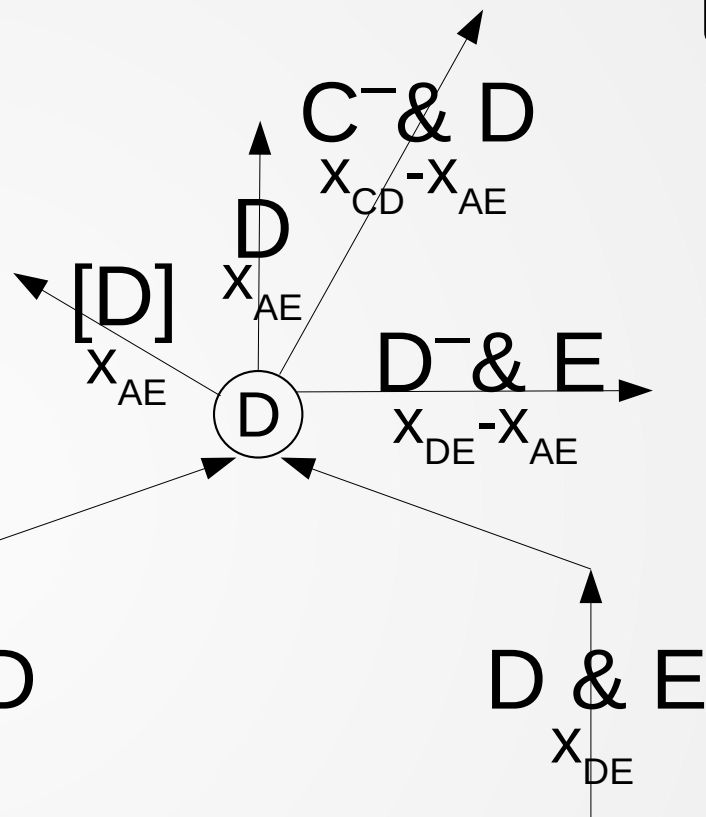
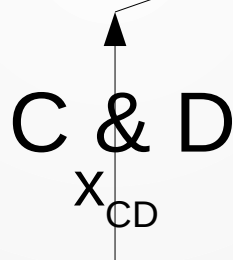
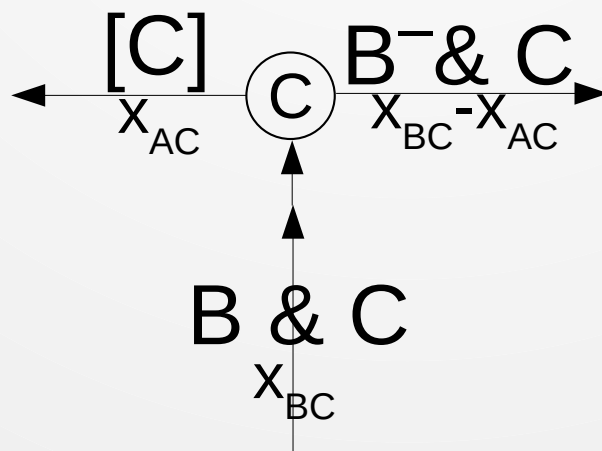
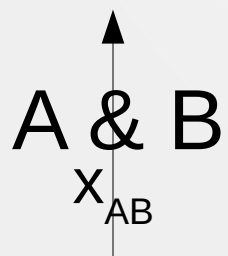
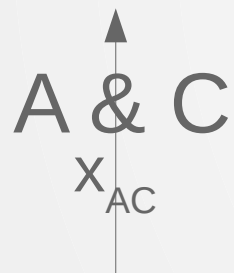
A & B
 x_{AB}

B & C
 x_{BC}

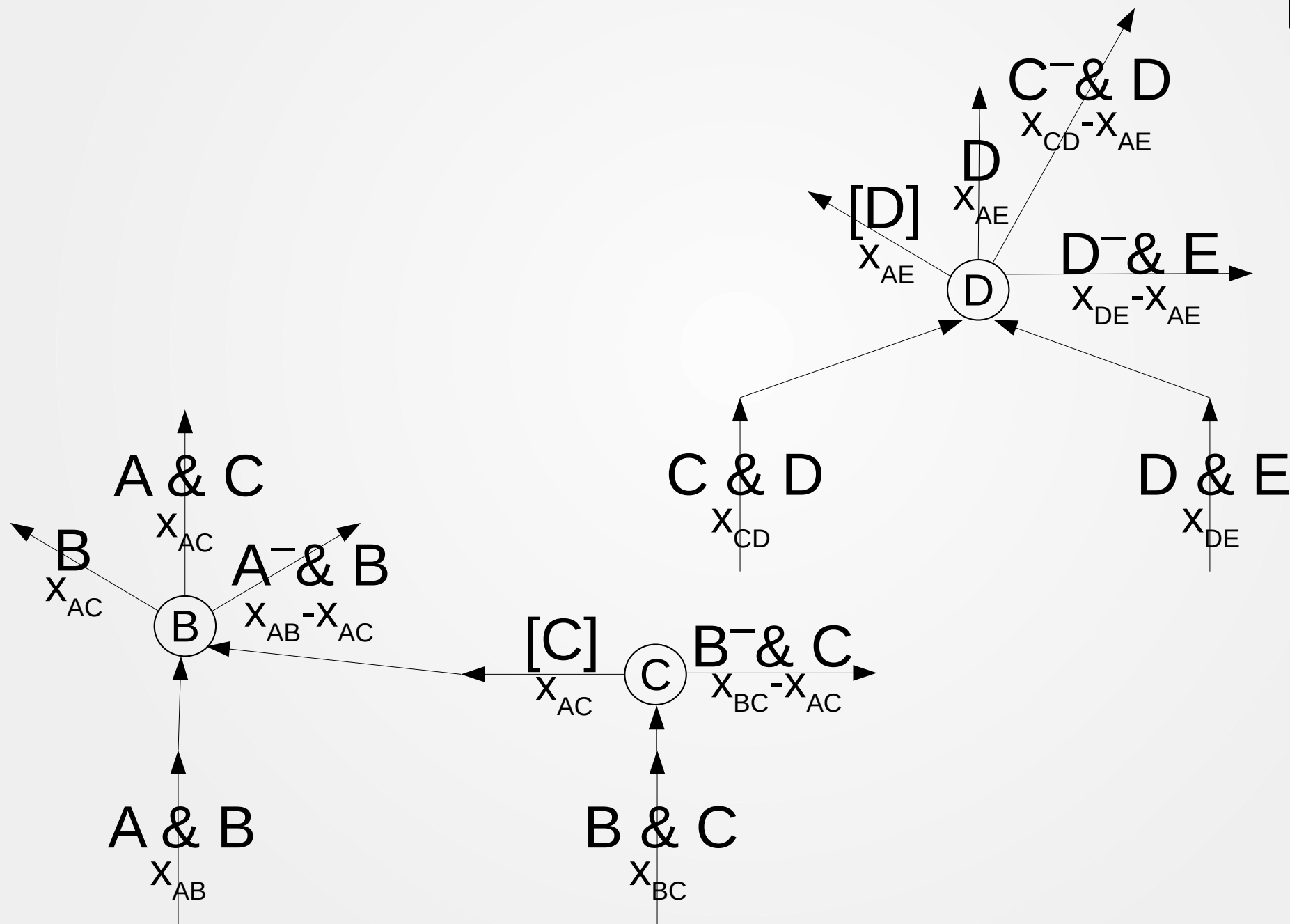
t=1



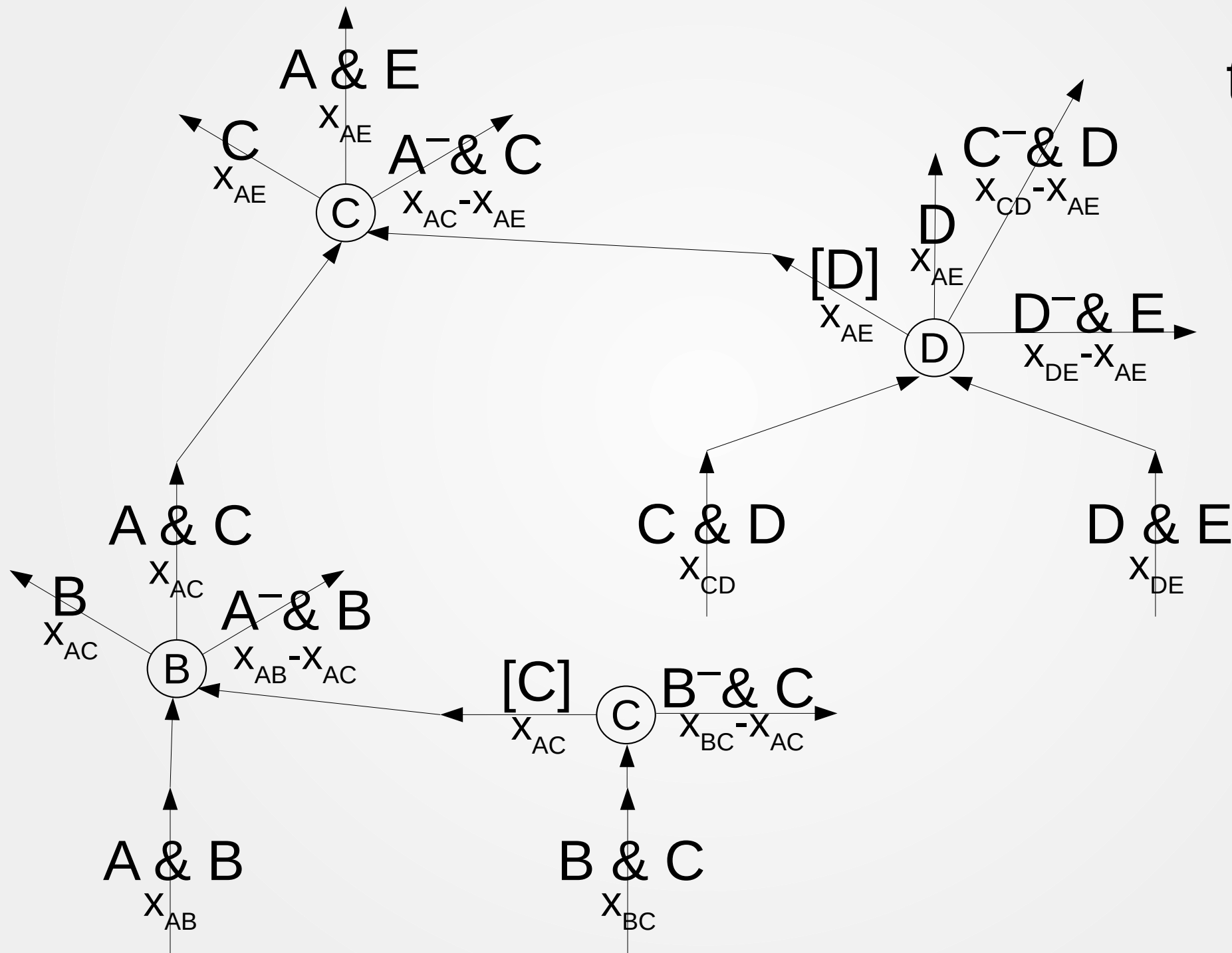
t=2



t=3



t=4



Design decisions

- UC secure
- Intuitive functionality
- State machine
- Uses \mathcal{G} ledger

Summary

- Composable analysis of the Lightning Network
- Construction and composable analysis of Recursive Virtual Channels for Bitcoin

Thank you!