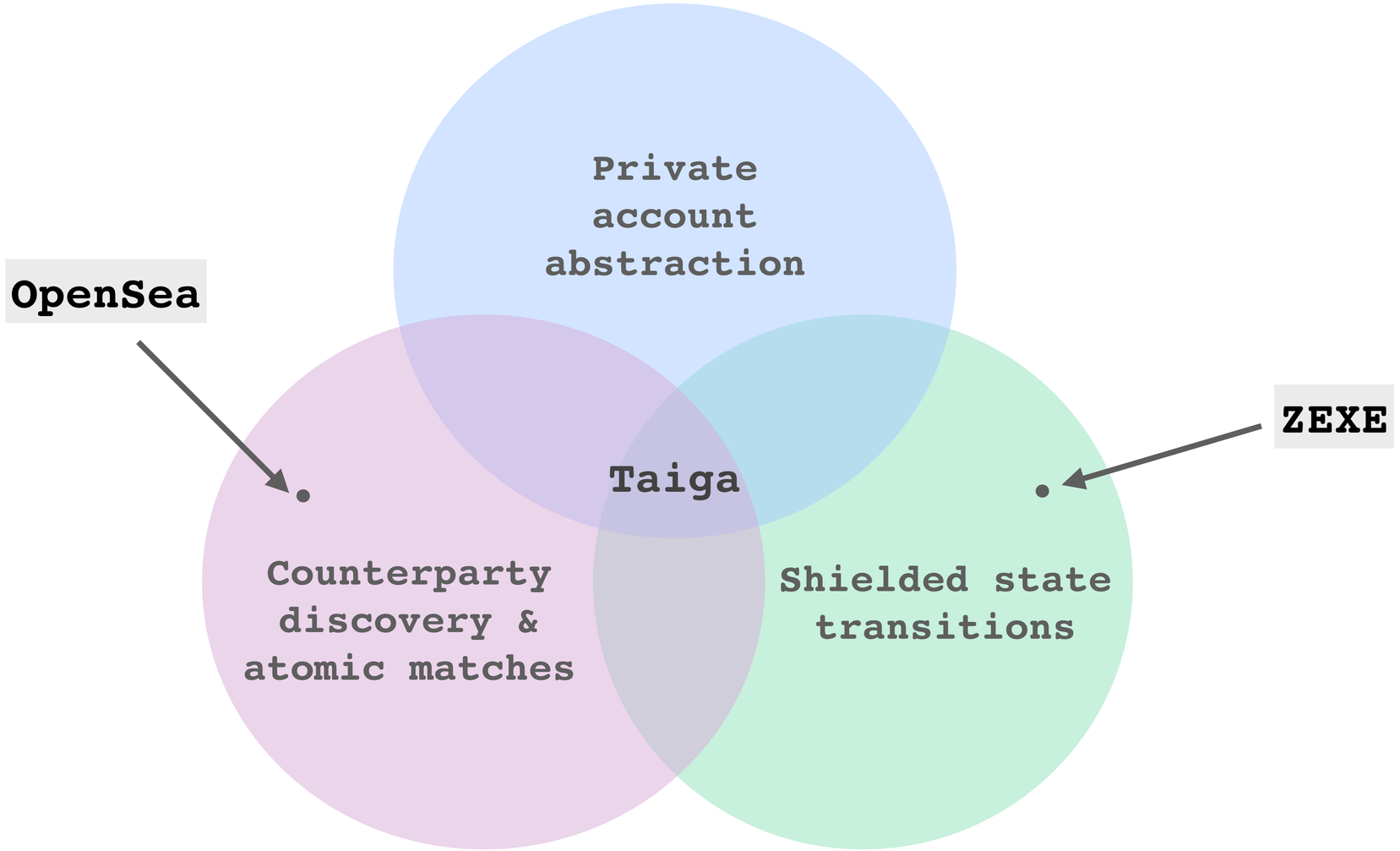
Taiga: a dark forest for composable private applications

Yulia Khalniyazova

zkSummit9, Lisbon, 2023

The Taiga team

What problems Taiga solves



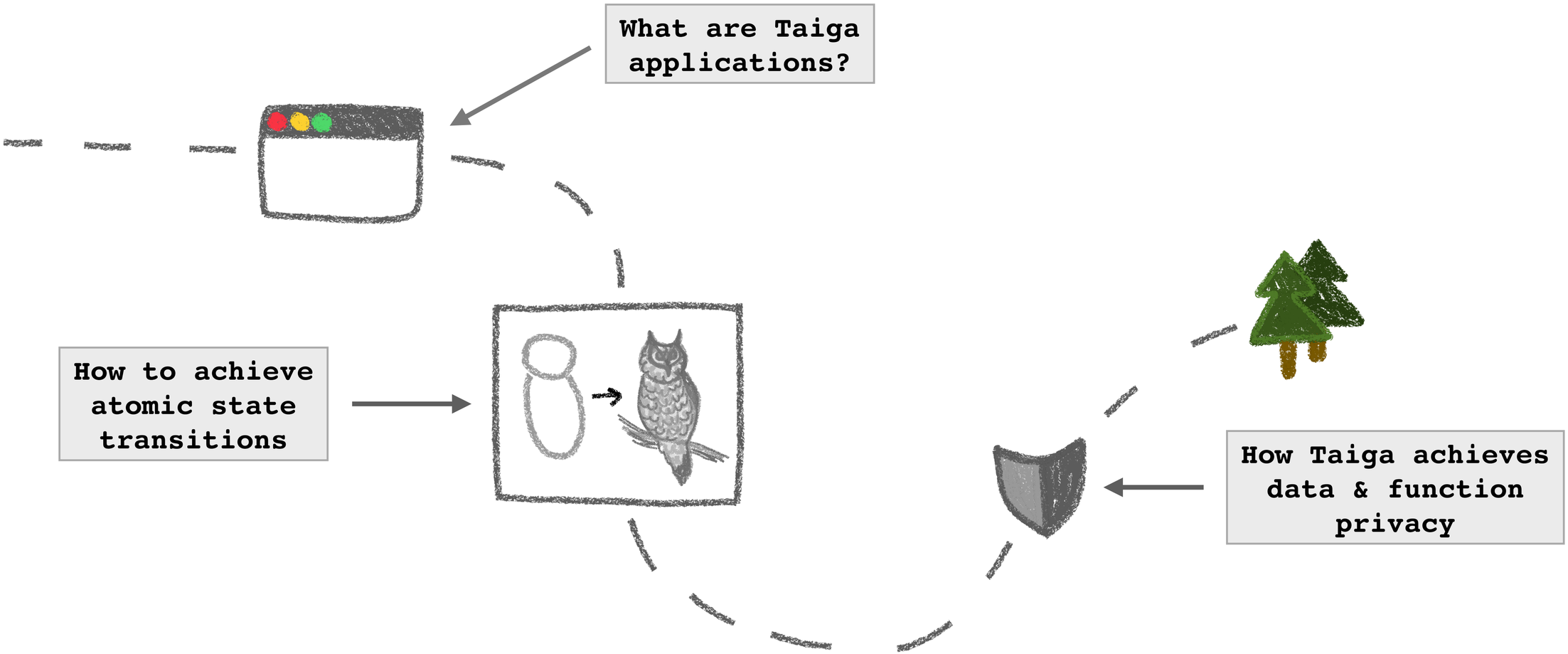
Taiga & Anoma

- A part of the Anoma protocol that takes care of shielded state transitions
- Can be used as a standalone component provided a data storage

Taiga overview

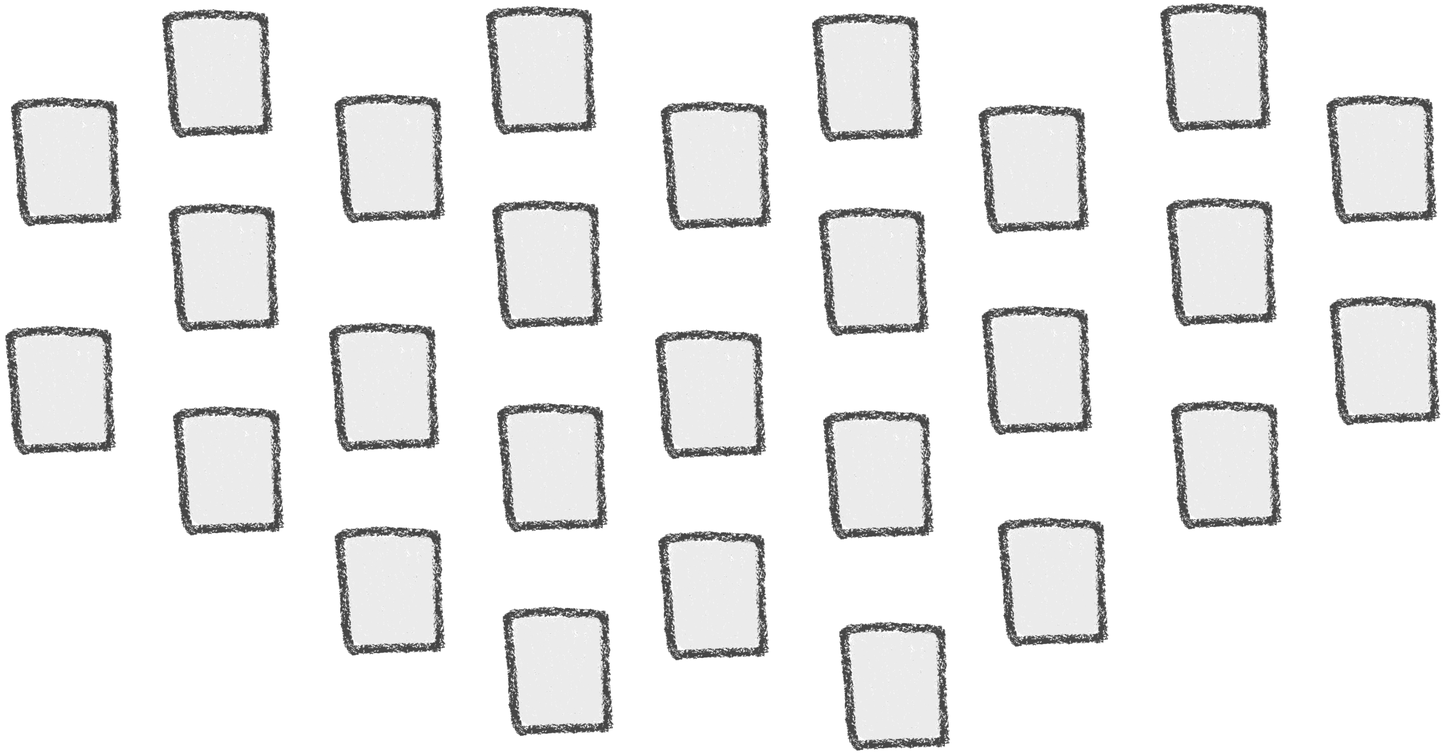
- An **OS** for distributed applications
- **Shielded** : data & function privacy
- **Private account abstraction**: users define their own authorization logic
- **UTXO-based**: each transaction is a list of input and output **notes**
- **Validity predicates** (VPs): declarative smart contracts. A valid transaction requires all relevant VPs to be satisfied
- **Atomic** state transitions of arbitrary complexity
- **Intent-centric:** built around **intents** - user preferences
- **Counterparty discovery:** matching intents is taken care of

The map of the talk



I. Taiga applications

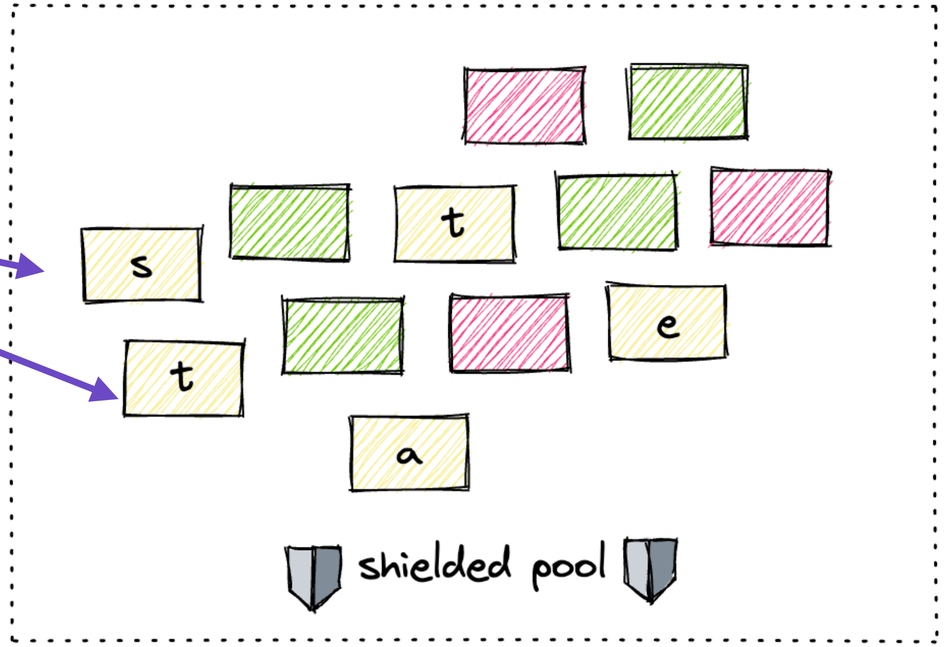
Taiga’s shielded pool



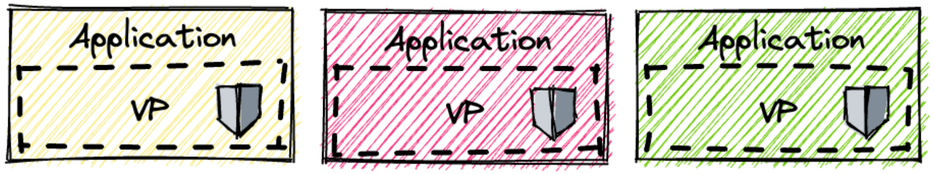
Taiga applications

Application state

Each note belongs to an application



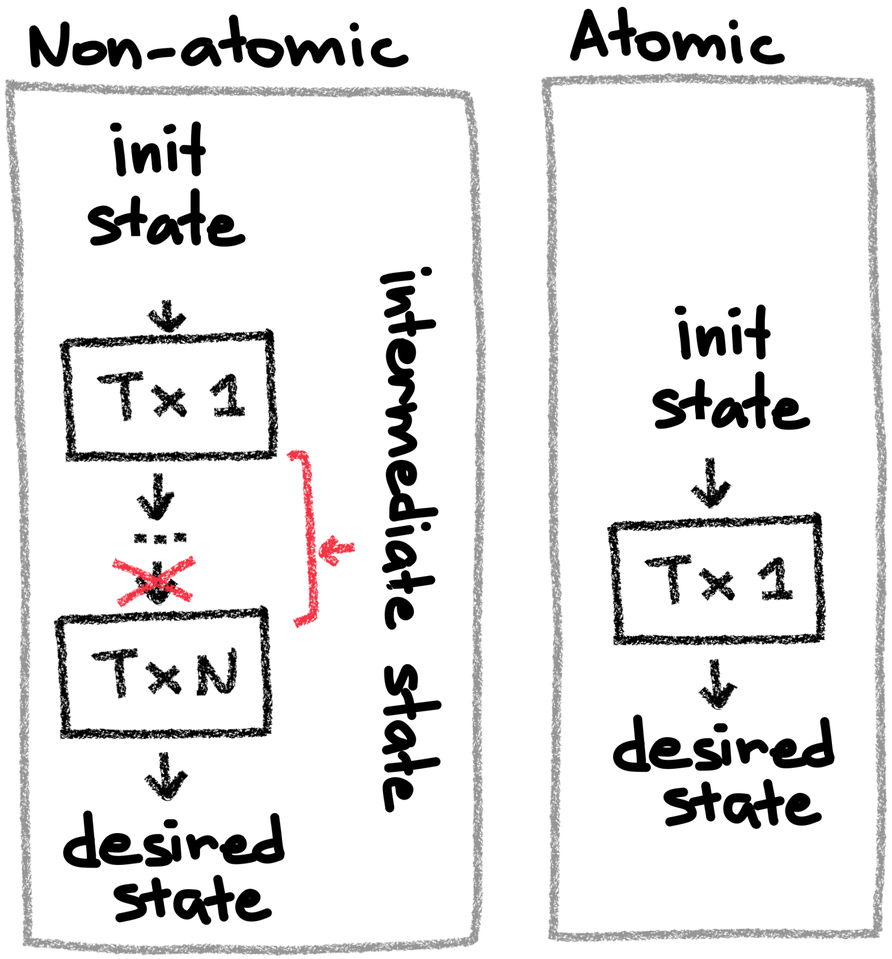
Application logic



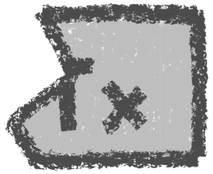
Application = application VP

II. Atomic state transitions

Atomic vs non-atomic

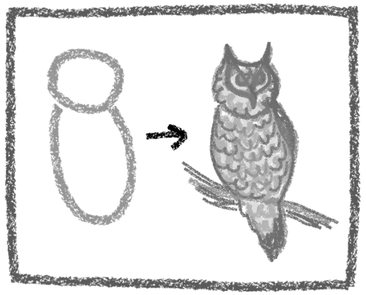


The recipe for atomic state transitions:



— partial transactions

— the intent application =



— solvers

Partial transactions

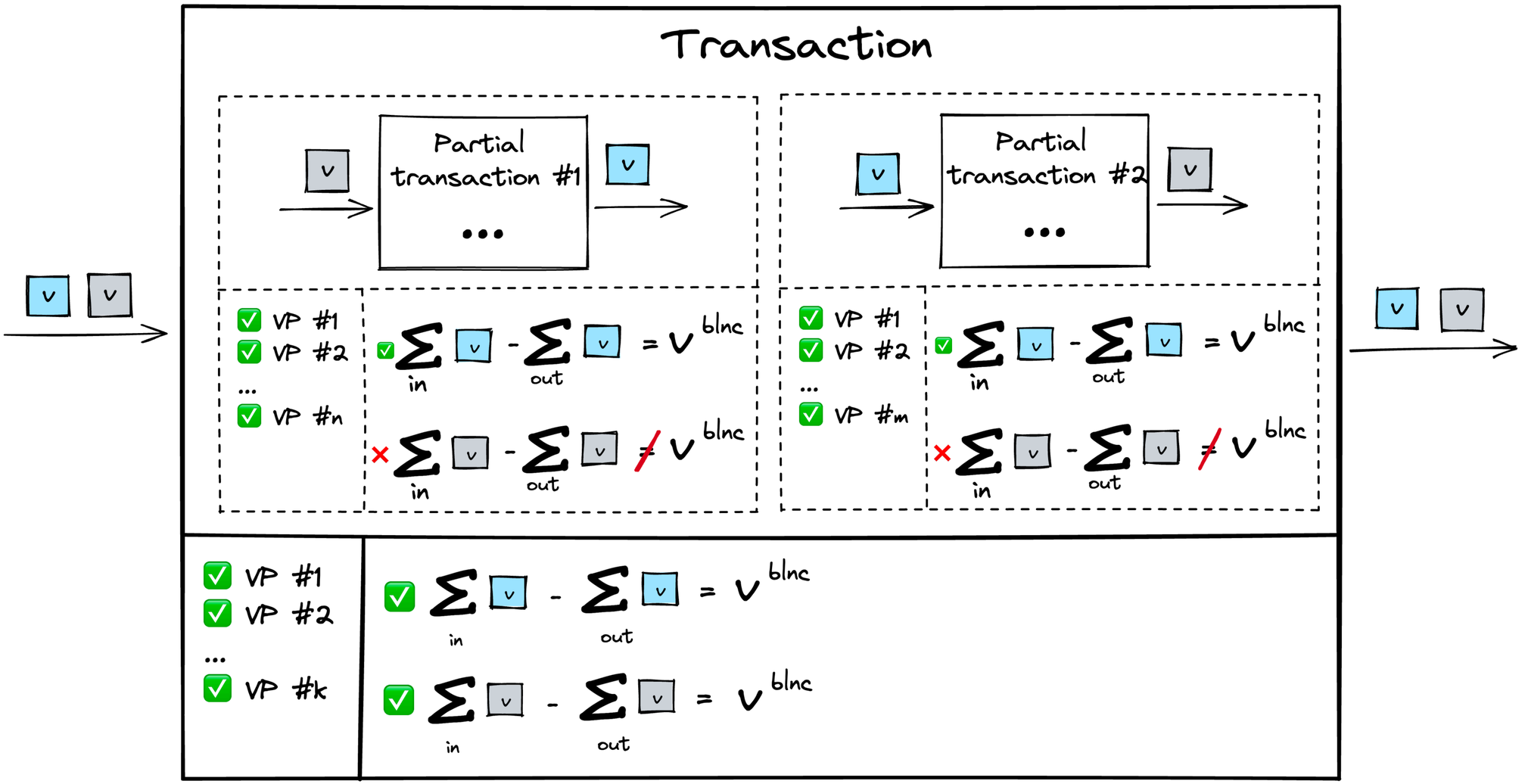
A valid Taiga transaction

	Transaction
Conditions	✅ VPs of all involved applications are satisfied
Balanced	✅ Yes
Can be published on the blockchain	✅ Yes

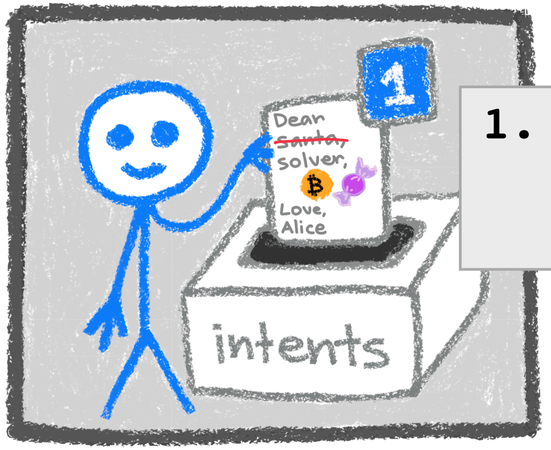
A partial transaction

	Transaction	Partial transaction
Validity	✅ VPs of all involved applications are satisfied	✅ VPs of all involved applications are satisfied
Balanced	✅ Yes	❌ No
Can be published on the blockchain	✅ Yes	❌ No

Transaction from partial transactions



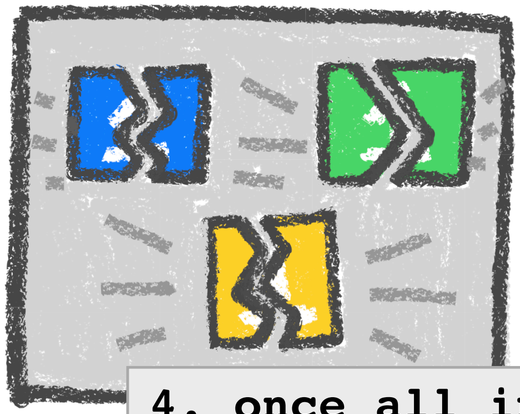
The intent application



1. create an intent note and send it along with your intent

2. while this intent note exists, the tx cannot yet be balanced

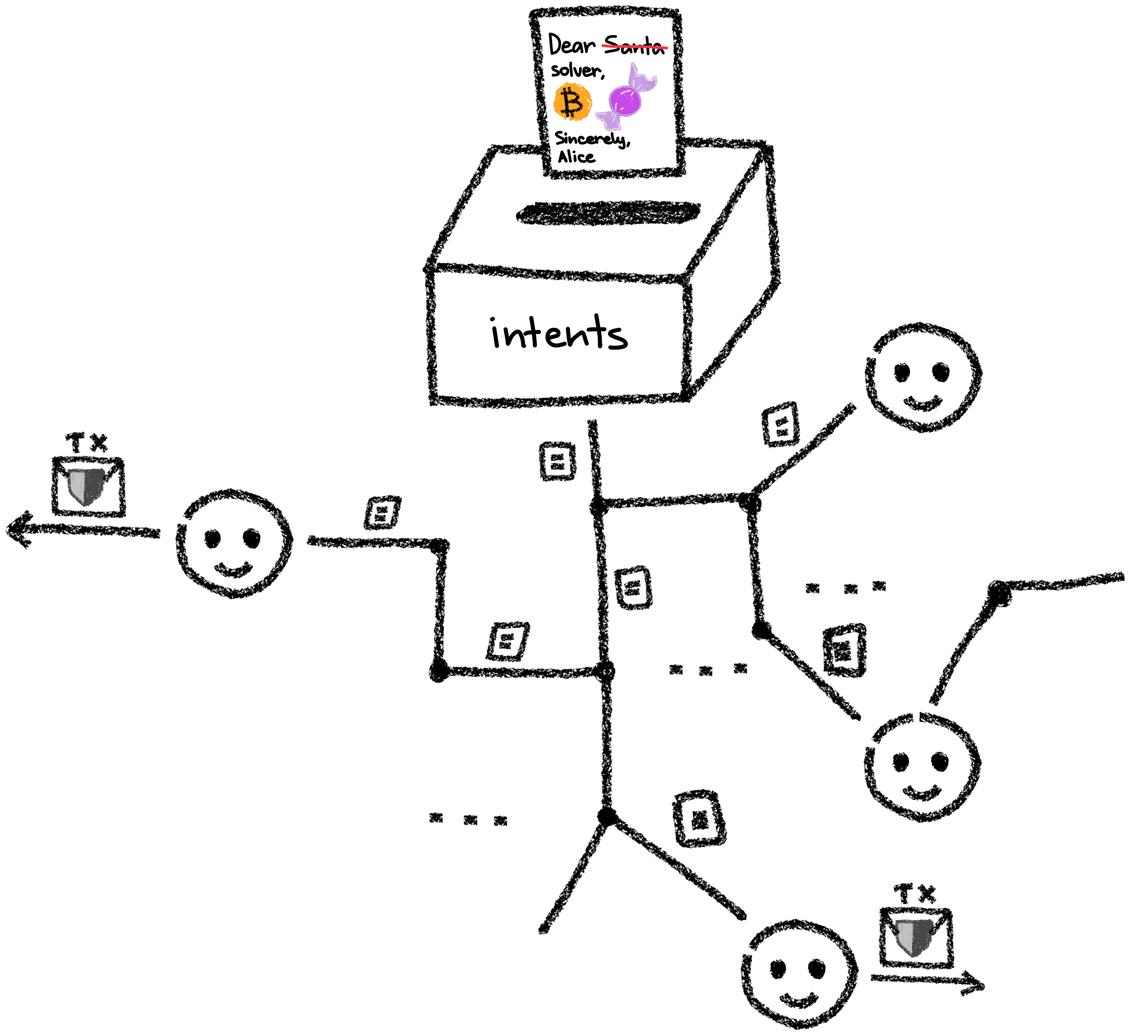
3. the intent note can only be destroyed once the intent is satisfied



4. once all intent notes are destroyed, the tx is balanced and can be published

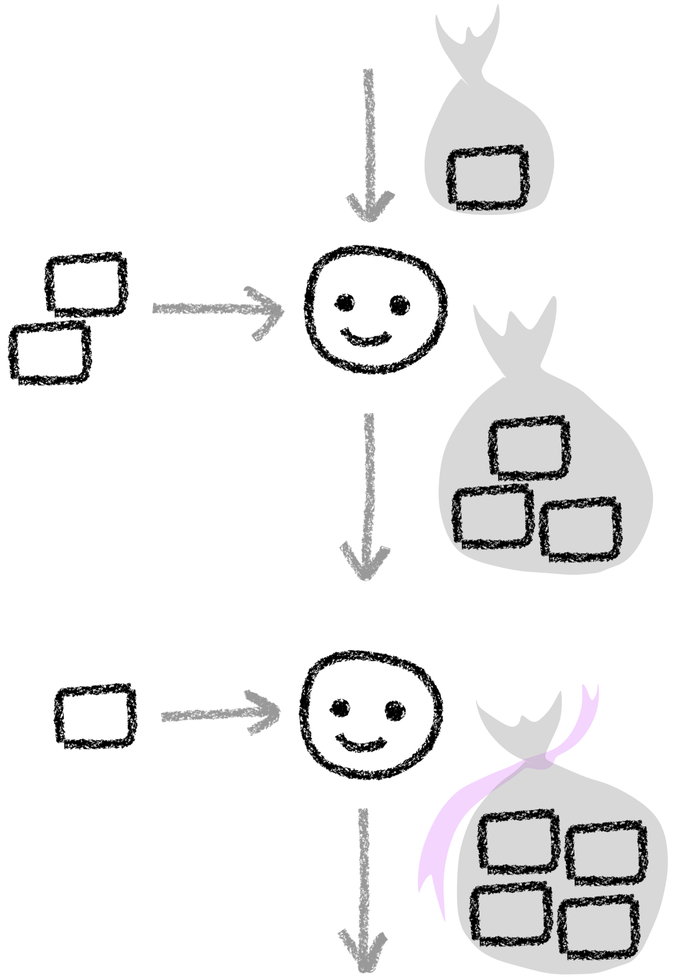
Who are solvers?

- **Solvers** are actors who match **intents** together and produce transactions



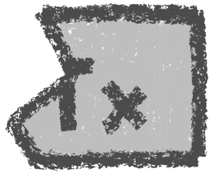
Solvers build, solvers prove

- to create an **atomic** transaction of **arbitrary** complexity, solvers:
  - create and share sets of partial transactions (including VP proofs)
  - and add new partial transactions to the set, until the set is balanced



The recipe for atomic state transitions:

cannot be published on the blockchain

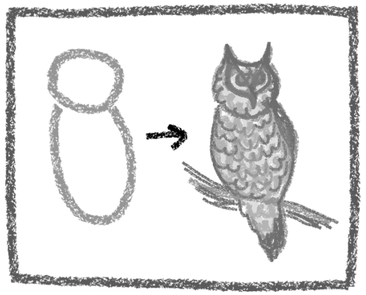


— partial transactions

makes sure the tx is unbalanced as long as the user intent is unsatisfied

— the intent application

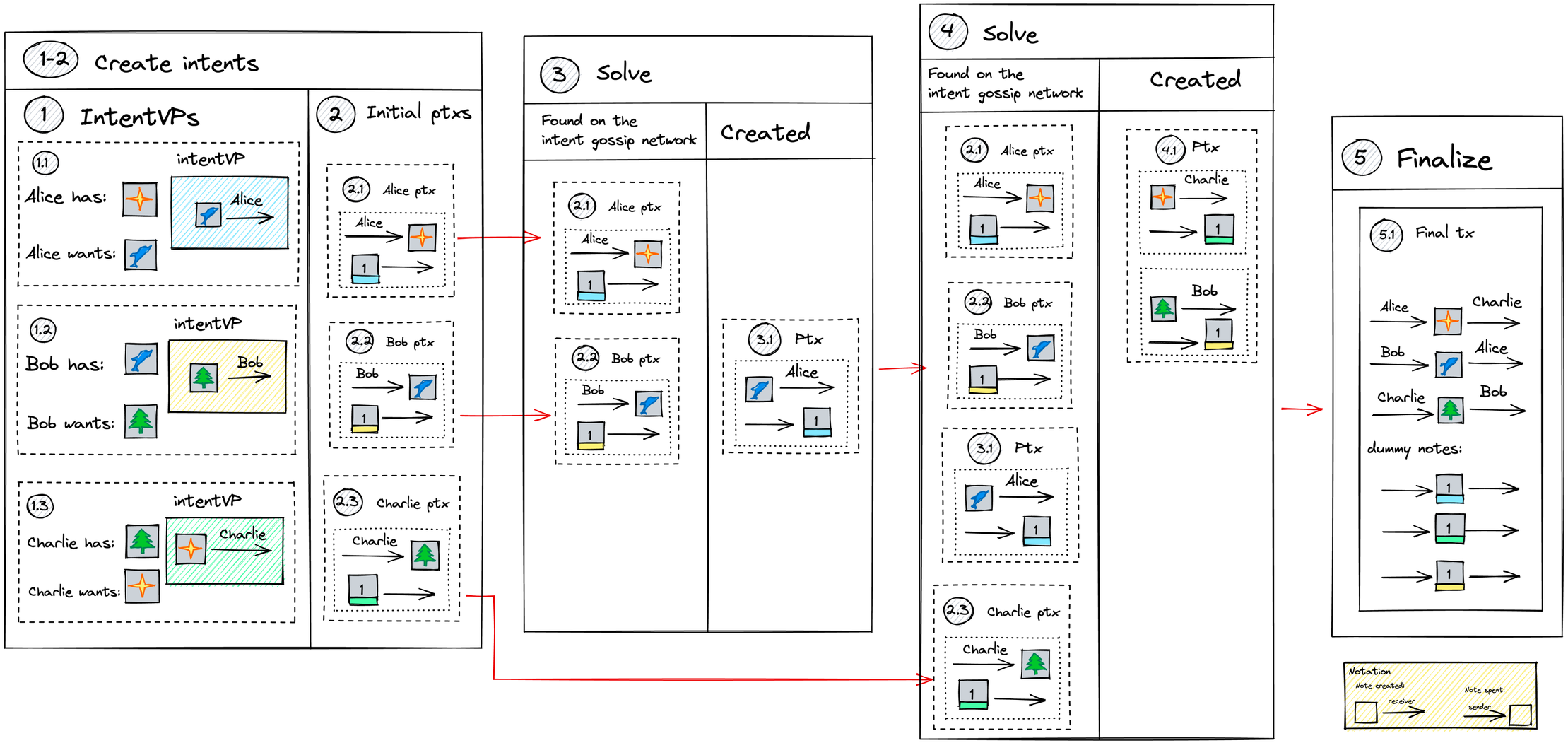
=



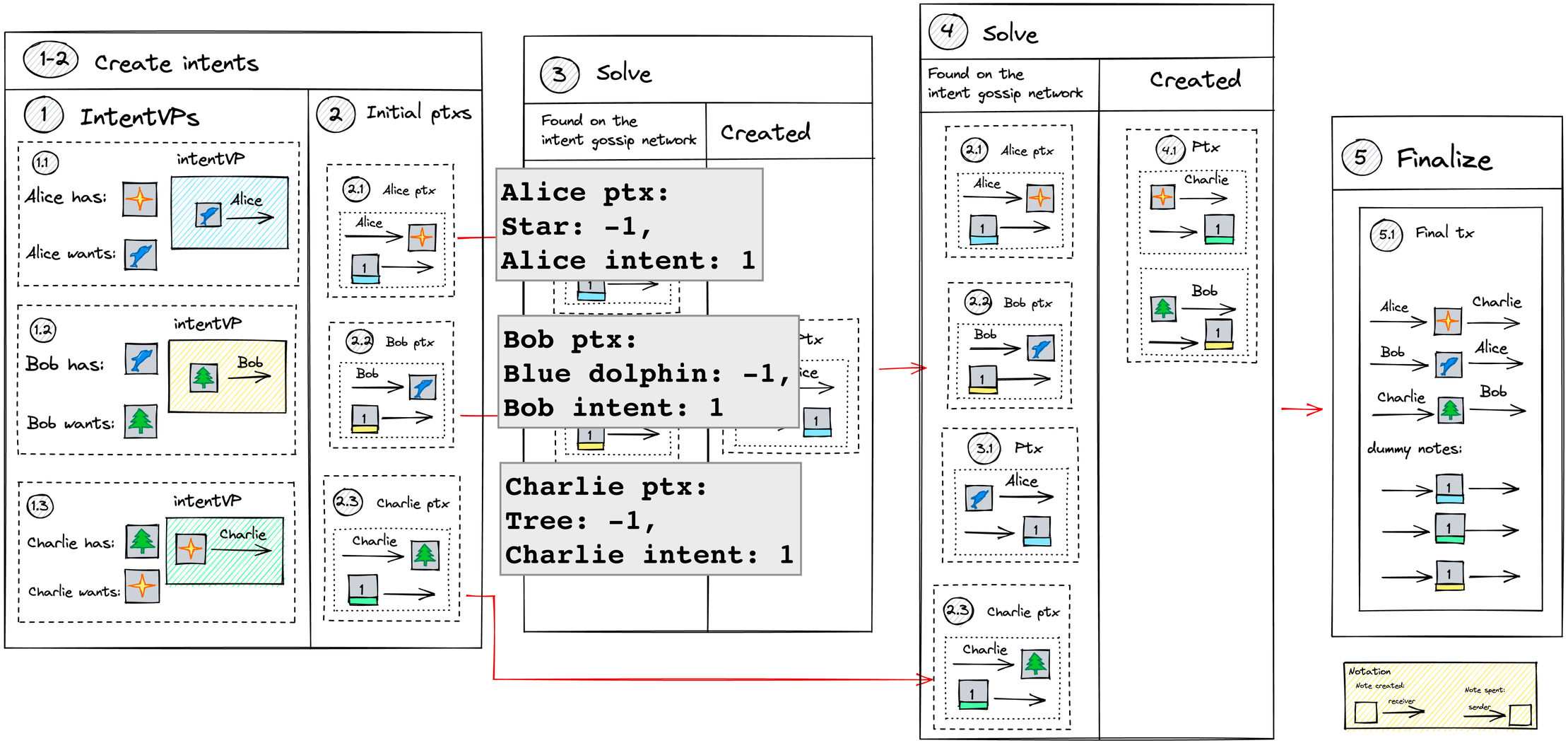
— solvers

update the transaction with new partial transactions until it reaches the final state

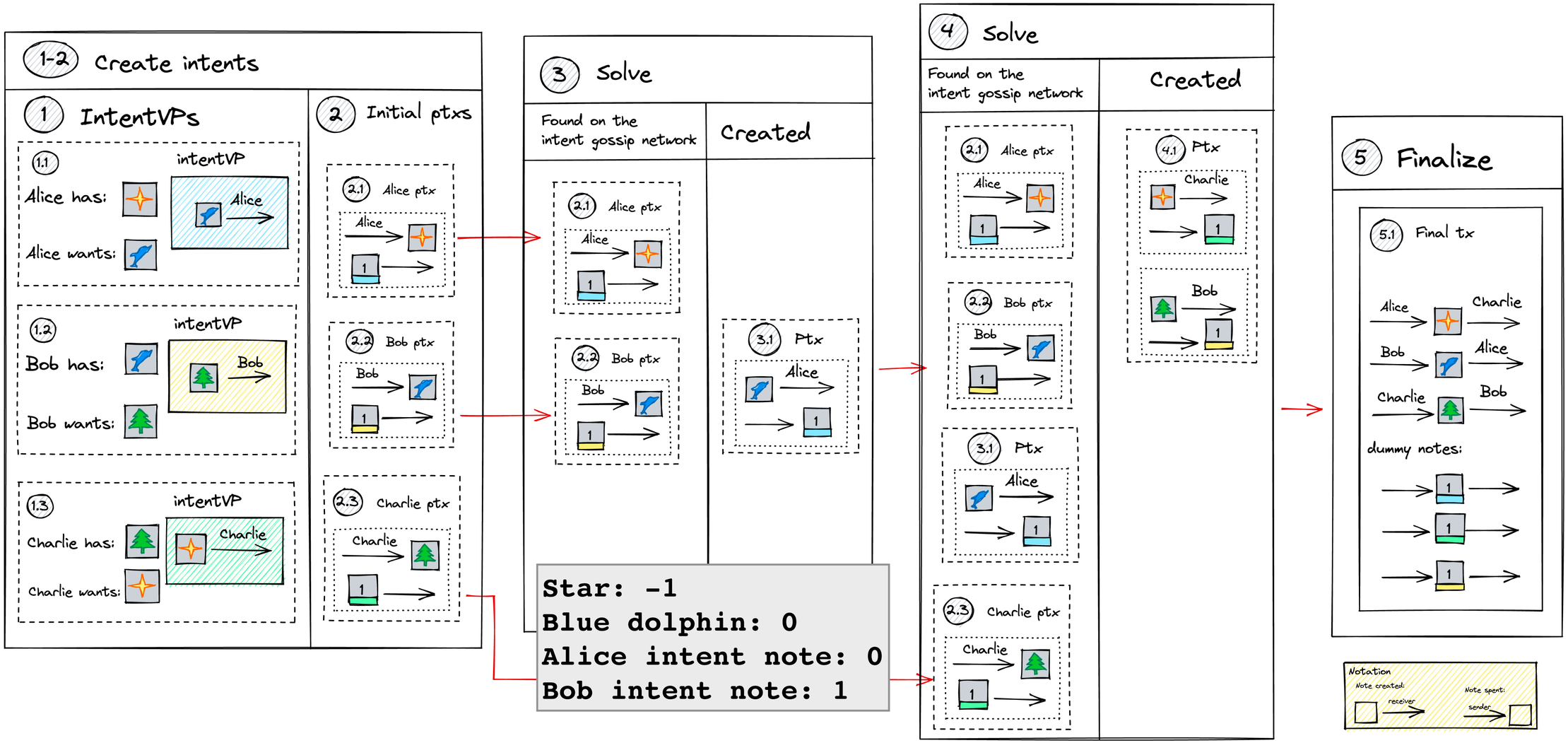
Example


1-2 Create intents
1 IntentVPs
1.1
intentVP
Alice has:
Alice wants:
Alice
1.2
intentVP
Bob has:
Bob wants:
Bob
1.3
intentVP
Charlie has:
Charlie wants:
Charlie
2 Initial ptxs
2.1 Alice ptx
2.2 Bob ptx
2.3 Charlie ptx
3 Solve
Found on the intent gossip network
Created
3.1 Ptx
4 Solve
Found on the intent gossip network
Created
4.1 Ptx
5 Finalize
5.1 Final tx
dummy notes:
Notation
Note created:
receiver
Note spent:
sender

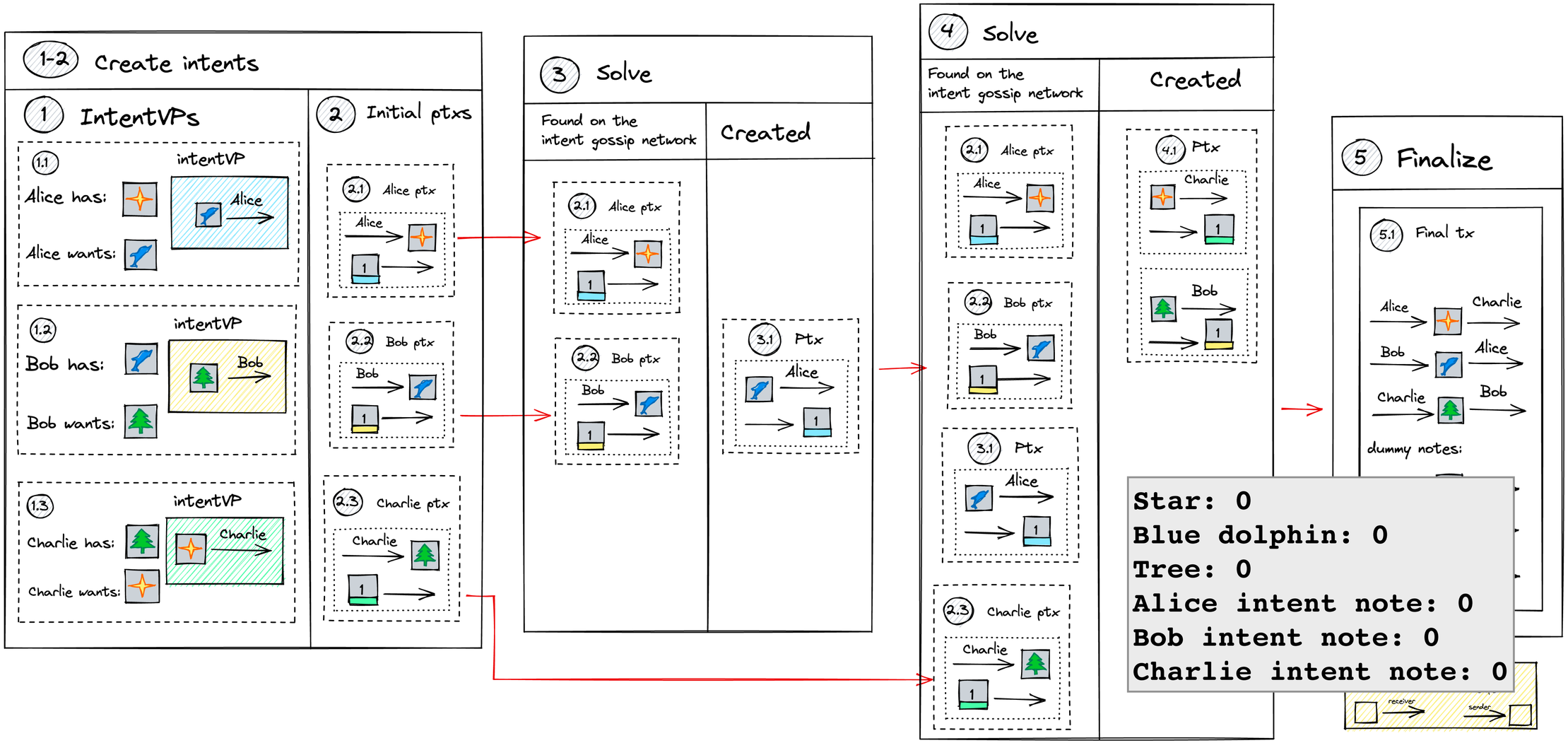
Example: steps 1-2



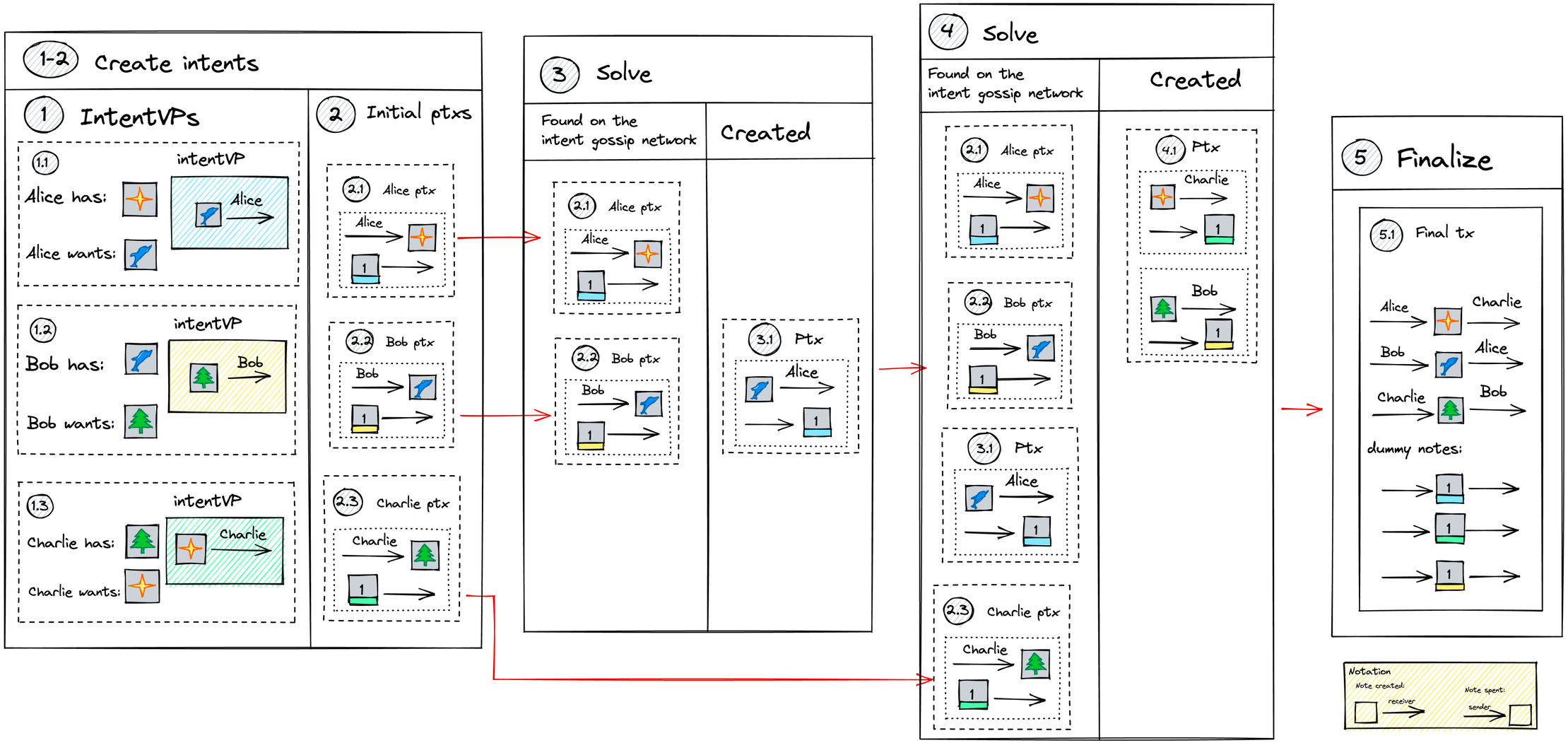
Example: step 3



Example: step 4



Example: step 4



III. Privacy in Taiga

Privacy: what and how

What		How
Data	Notes	Verifiable encryption, note commitment
	Transaction (notes, participants, quantities, etc)	ZKP, hashes, blinding
Function	Validity predicates	Recursion

Taiga 🤍 Halo2

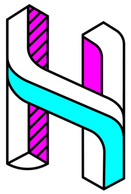
- Supports recursion and accumulation
- Has some helpful gadgets available
- No trusted setup - a bonus when dealing with lots of circuits

The current state of Taiga

- Many details remain to be confirmed
- The code is available at https://github.com/anoma/taiga

The future of Taiga

- Programmable data transport across public, shielded and private domains
  - Public = transparent
  - Shielded = ZKP (single-user private state)
  - Private = FHE (multi-user private state)
- Closer integration with Ferveo
    - Prove correspondence between private state and data encrypted to Ferveo using ZKPs



Thanks!

🌲 Taiga code: https://github.com/anoma/taiga

🌲 my twitter: @vveiln