Reference Stateの可能性

SBI R3 Japan株式会社　井本 翔太

自己紹介

井本 翔太(いもと しょうた)

- SBI R3 Japan エンジニアリング部(2021/12月～)

<略歴>
愛知県出身→大学進学に伴い上京(現在大学3年)

<主な業務内容>
Corda機能調査、デモアプリケーション改修、翻訳など

<興味分野>
ブロックチェーン、分散処理、P2P
将来は分散システム系に従事したいと考えています

SBI R3 Japan

目次

SBI R3 Japan

SBI R3 Japan

1. 3つのユースケースと共通課題について

1-1. ユースケース1　－DID/VCの説明

- DID/VC
  - ✓DID(Decentralized Identifier): Web3.0を実現する分散型のID
  - ✓VC(Verifiable credentials): DIDに紐づく検証可能な資格情報

- 今回はVCに焦点を置いたユースケースになります

SBI R3 Japan

1-1. ユースケース1　–想定シナリオ

- 前提
  - ✓Alice, VC発行会社, 企業が存在
  - ①Aliceはテストを受講し，VC発行会社の管理するVCを発行
  - ②AliceはそのVCを企業での就職活動時に提出

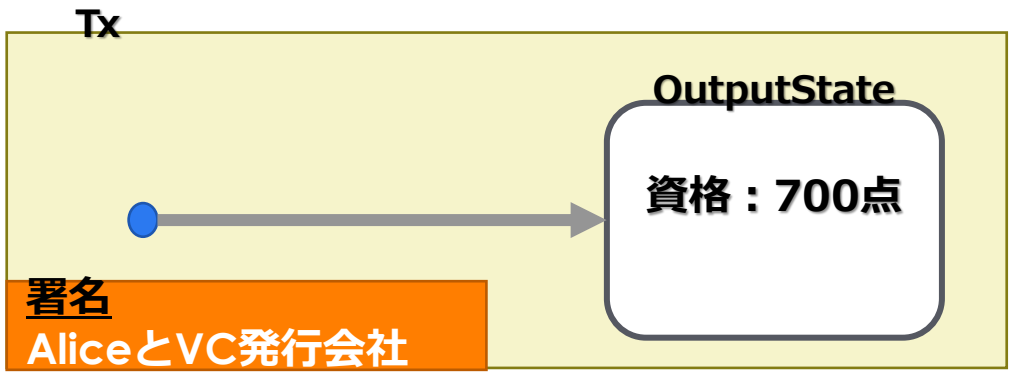

図1. スコア700点を示す資格の発行tx

1-1. ユースケース1　-イレギュラーの発生

- ここで、Aliceがテスト受講の際に不正していたことが判明
  - ✓VC発行会社は資格700点のVCを失効するtxを発行

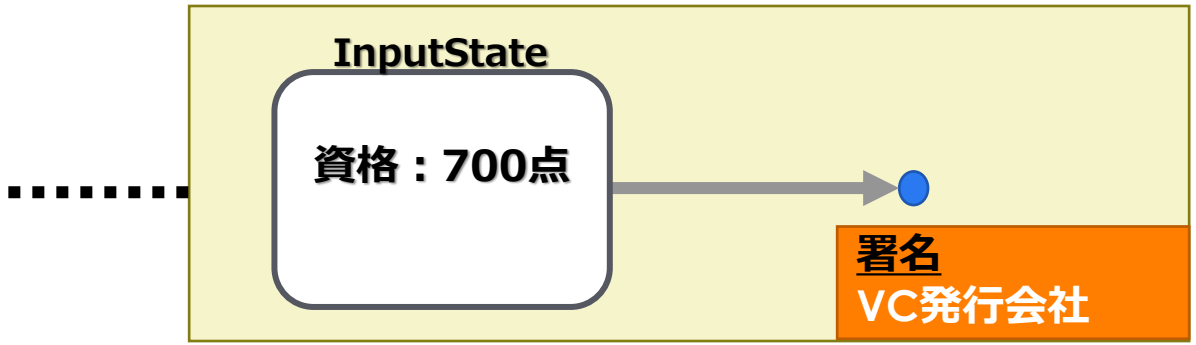

図2. スコア700点を失効するtxの発行

1-1. ユースケース1　–問題点

- Aliceが失効されたVCを企業に渡す事が技術的に可能

※パブリックチェーンでも同様の問題が発生します

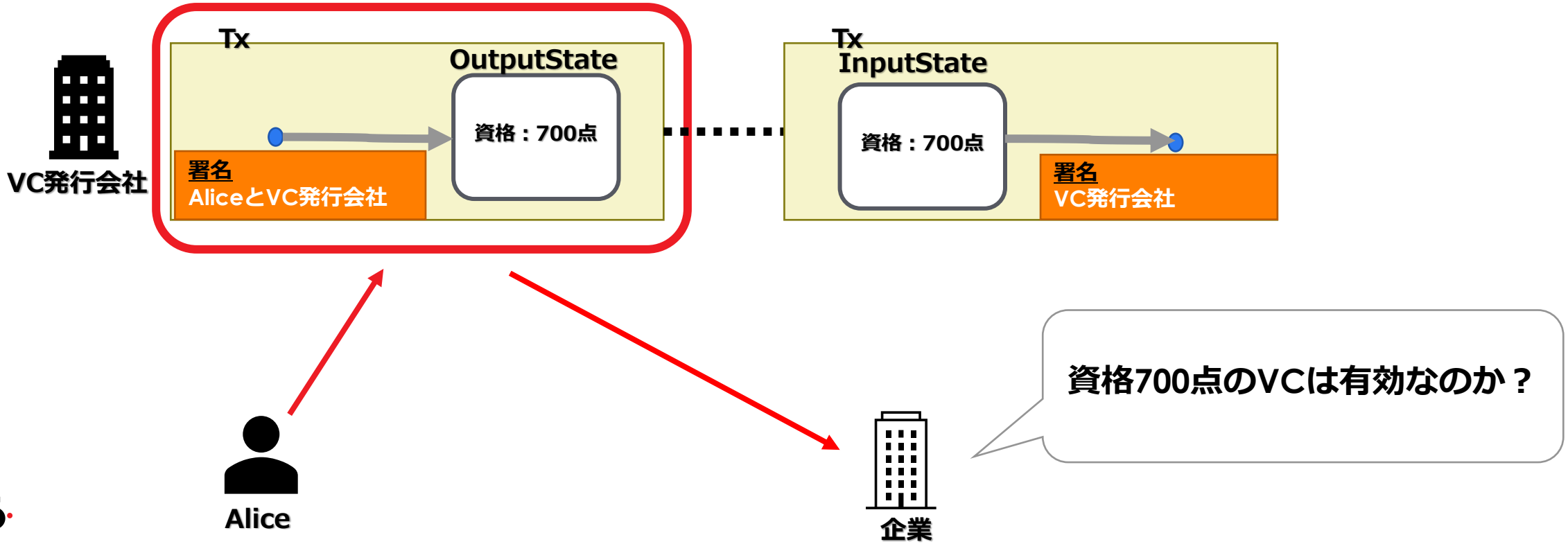

図3. Aliceが失効されたVCをCompany Bに渡す一連の流れ

参考スライド

- 企業は渡されたVCの有効性に関して不明
  - ✓VCに関するtxの当事者でないため
    - ➢ 既に失効されている可能性も…

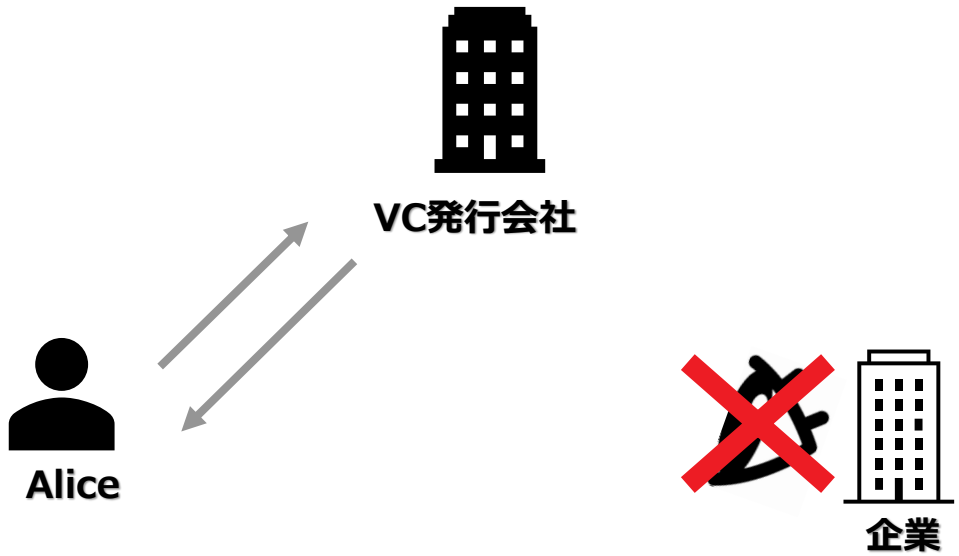

図4. VCに関する取引の第三者である企業のビジビリティを表す図

1-1. ユースケース1　-解決策

- Aliceから渡されたtxのVC state(700点)の有効性を検証
  - ✓**未消費**=VCは有効(誰も更新していないため)
    - ➢ VC stateをInputStateとする失効txは発行されていない
  - ✓**消費済み**=VCは無効
    - ➢ VC stateをInputStateとする失効txが発行された

SBI R3 Japan

1-2. ユースケース2　–想定シナリオ

- 前提
  - ✓Party A, Party B, Party Cが存在
  - ①Party Aが持つ日本酒をParty Bに移転

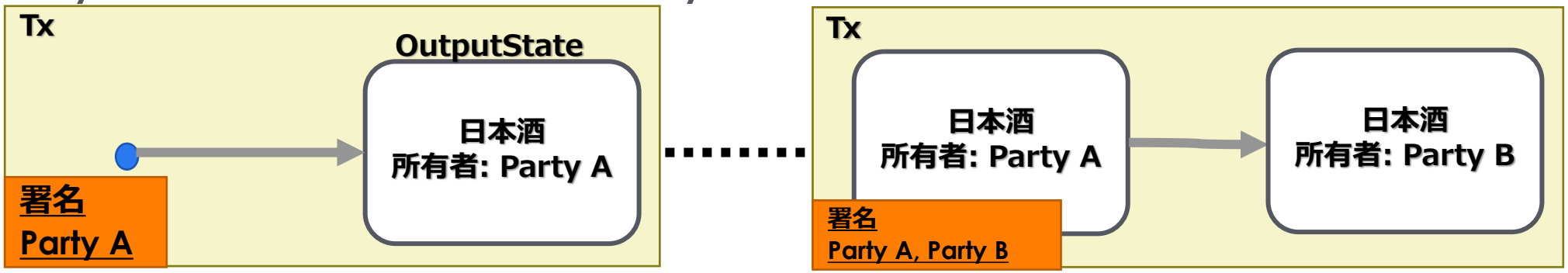

図5. Party AからParty Bへの日本酒の移転状況を示す図

  - ②Party CがParty Aに日本酒を購入したい意向を示す

SBI R3 Japan

参考スライド

- トークン取引前に商品の所在確認のため、
  商品の所有者に対して事前確認を行う必要がある

- 今回のユースケースでは、Party Aが日本酒所持を証明するためParty Cにtxを渡す＝事前確認

SBI R3 Japan

1-2. ユースケース2　-イレギュラーの発生

- Party Aは移転したにも関わらず商品の事前確認として誤って赤枠のtxを渡した

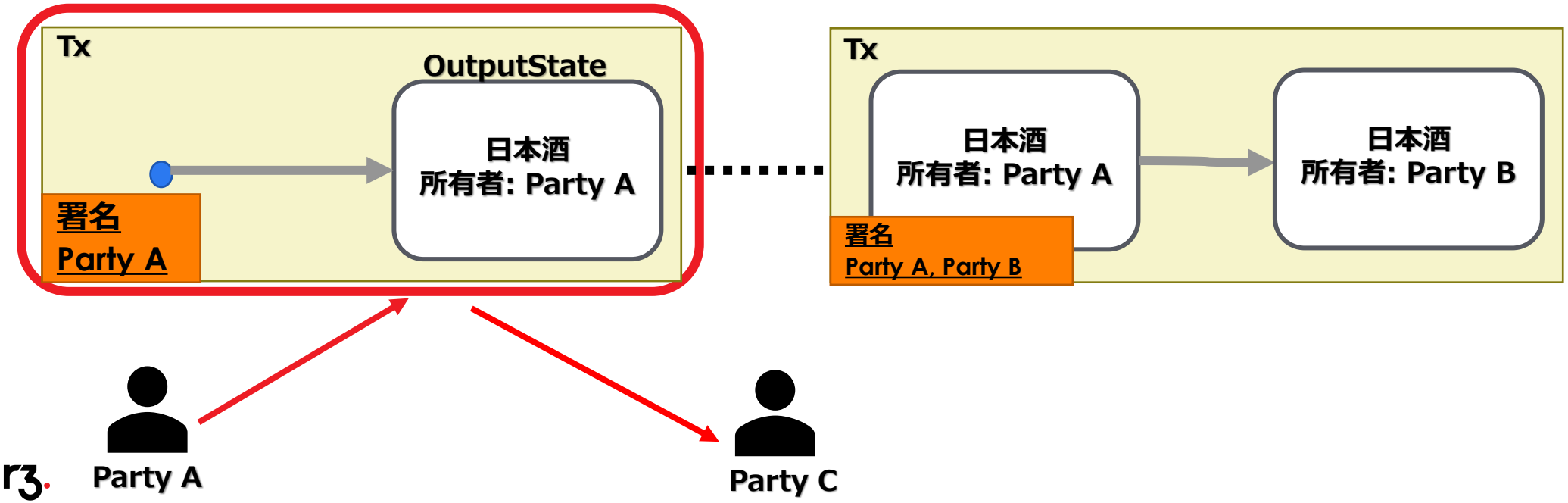

図6. Party Aから誤った日本酒の情報がParty Cに渡される流れを示す図

1-2. ユースケース2　-問題点

- Party Cは渡されたtxの有効性が不明

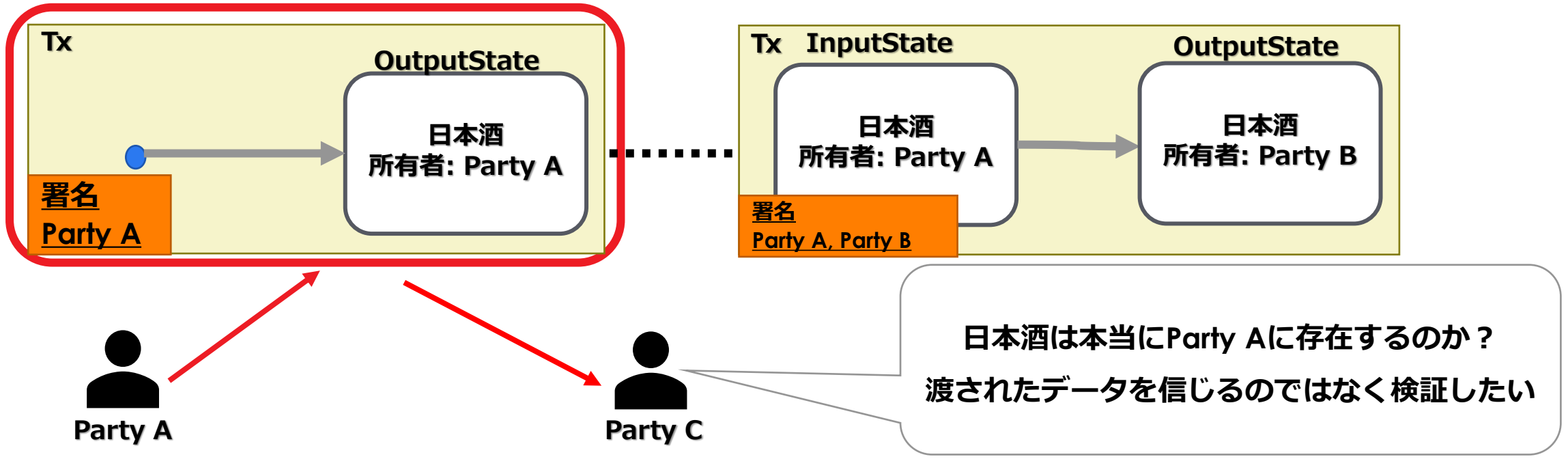

図6. Party Aから誤った日本酒の情報がParty Cに渡される流れを示す図

参考スライド

- Party Cは日本酒の移転に関するtxの存在を検知できない
  - ✓当事者はParty AとParty Bのため

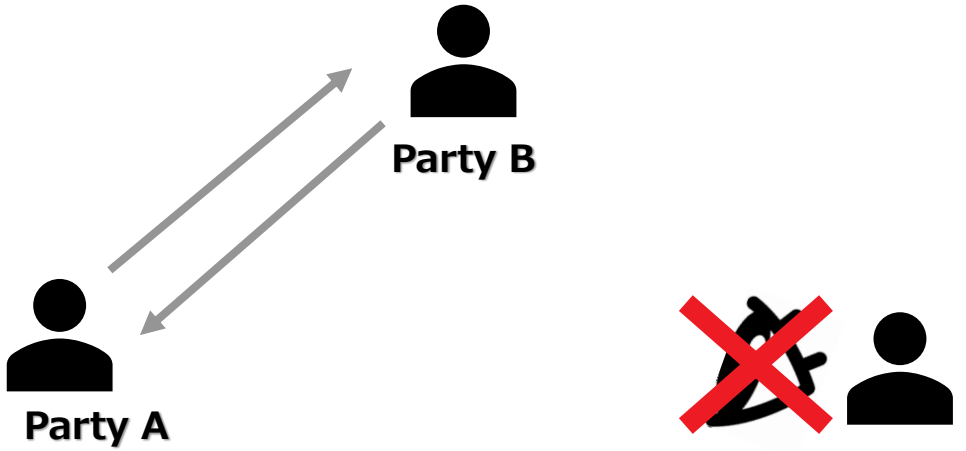

図7. 日本酒移転取引の第三者であるParty Cのビジビリティを表す図

1-2. ユースケース2　–解決策

- Party Cが、Party Aが渡したtxの日本酒stateを検証
  - ✓**未消費**=Party Aは日本酒を保持(更新されていないため)
    - 日本酒stateをInputStateとする移転txは発行されてない
  - ✓**消費済み**=Party Aは日本酒を保持していない
    - 日本酒stateをInputStateとする移転txは発行された

SBI R3 Japan

1-3. ユースケース3　–想定シナリオ

- 前提
  - ✓Node-Notary間で通信中にNodeで障害が発生
  - ✓Nodeは、時点スナップショットを用いたロールバックで復旧を試みる

SBI R3 Japan

1-3. ユースケース3　–想定シナリオ

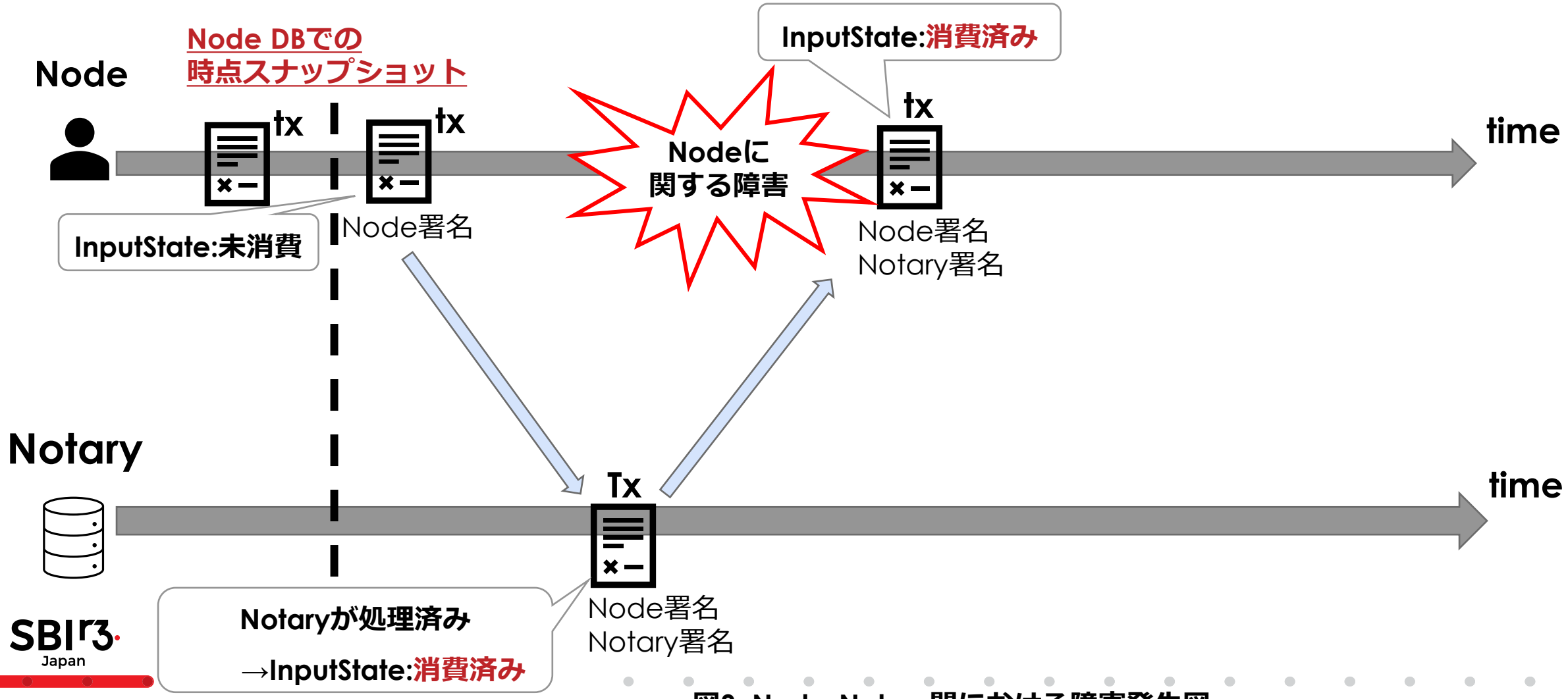

図8. Node-Notary間における障害発生図

1-3. ユースケース3　–イレギュラーの発生

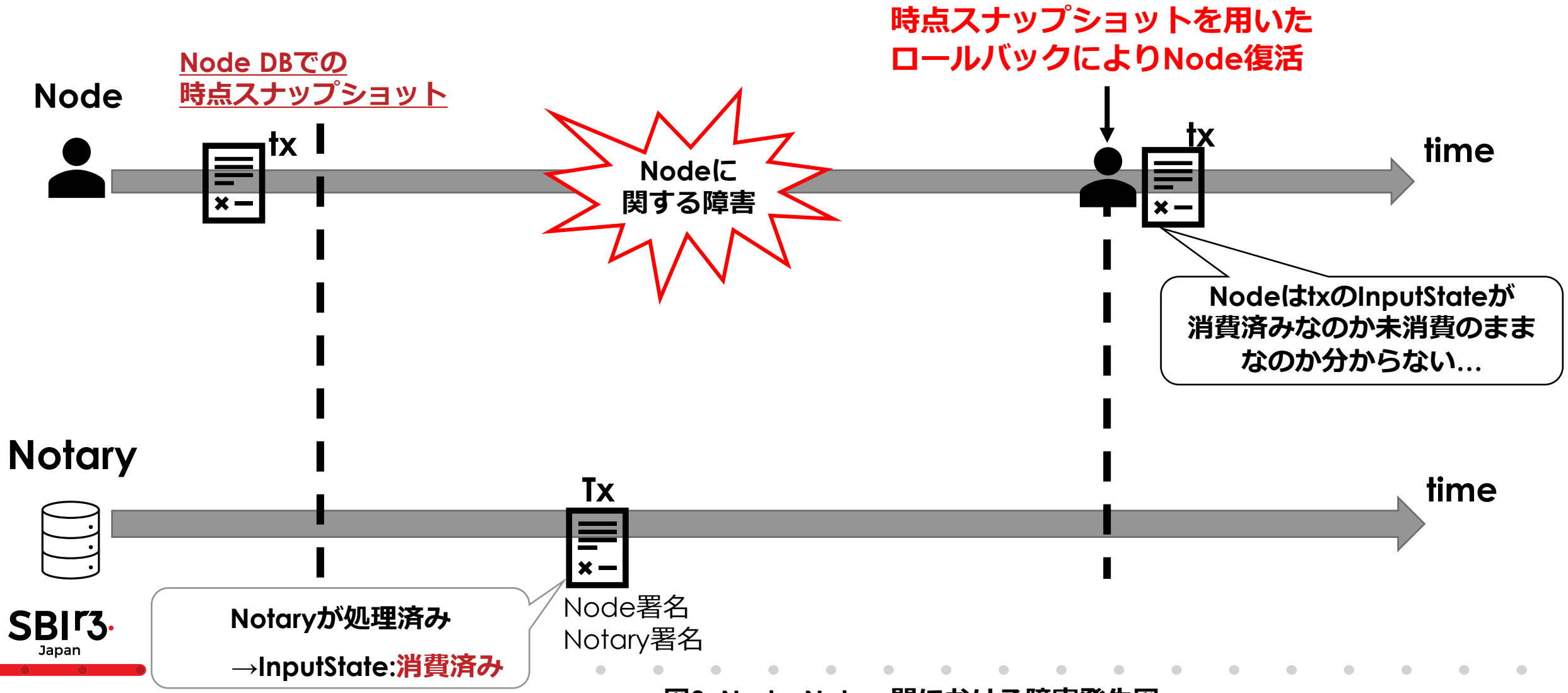

図8. Node-Notary間における障害発生図

1-3. ユースケース3　－問題点

- 時点スナップショット以降の消費状況が不明
  - ✓Nodeは時点スナップショットの位置の情報までしか復旧できない

時点スナップショット後のInputStateの消費状況は確認できないか？

SBI R3 Japan

1-3. ユースケース3　–解決策

- NodeがInputStateを検証
  - ✓**未消費**＝NotaryはInputStateを処理していない
  - ✓**消費済み**＝NotaryはInputStateを処理した

SBI R3 Japan

1-4. ユースケースに共通する課題

- あるtxの中のstateの**消費状況**を知りたい
  - ✓そのstateをInputStateとしてtxに含めNotaryに問い合わせては？

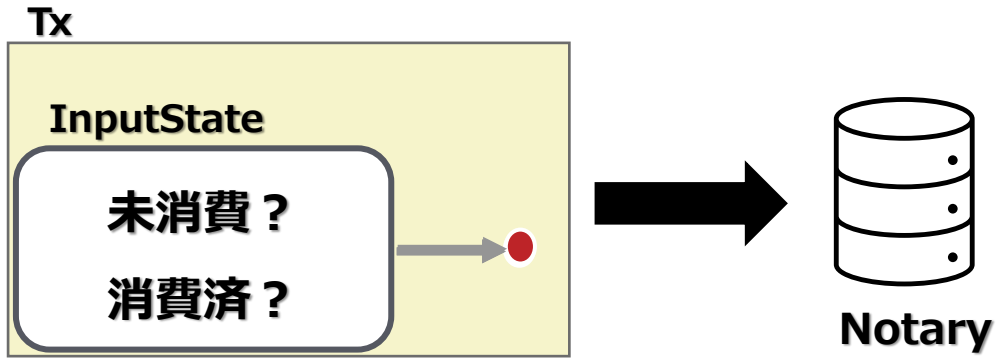

図11. ユースケースに共通するstateの問い合わせ処理を示した図

1-4. ユースケースに共通する課題

- 問い合わせにより強制的に消費済みへ…

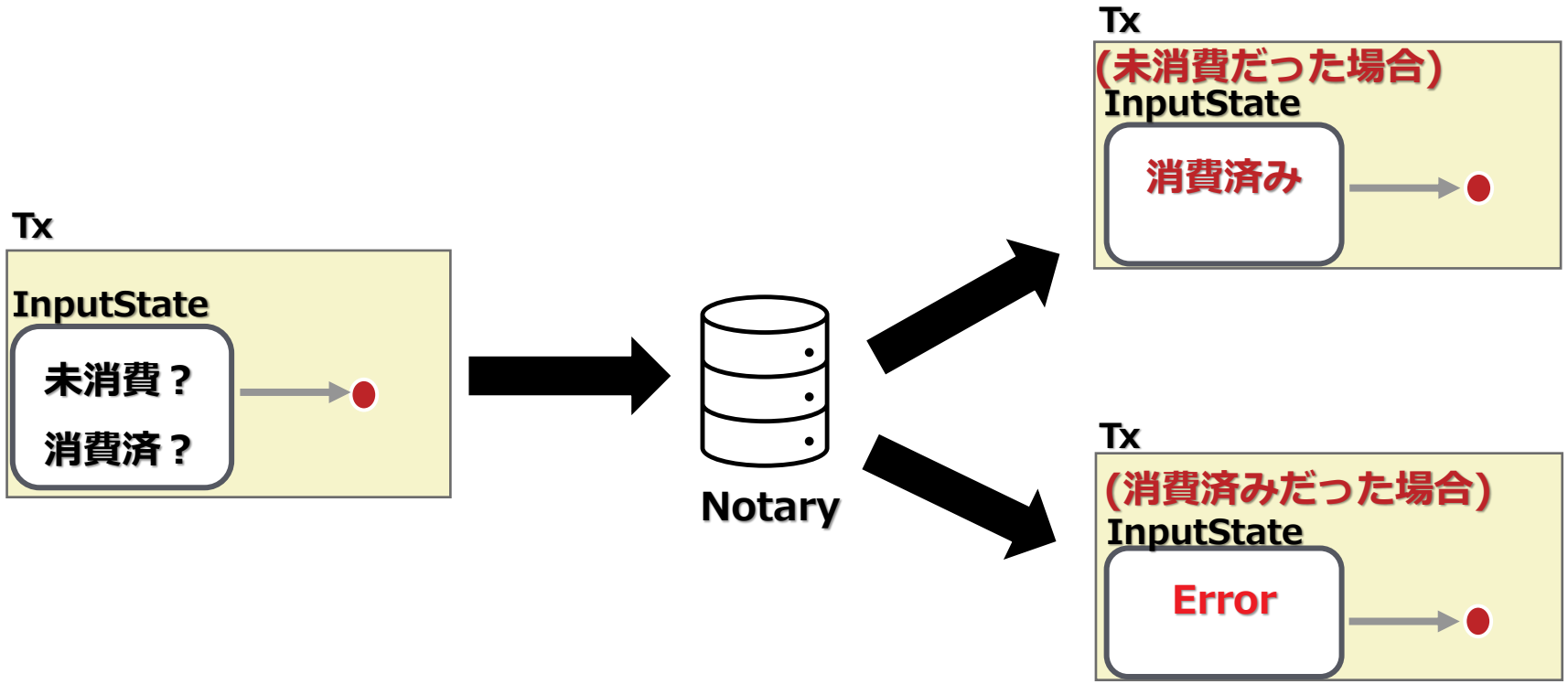

図12. ユースケースに共通するstateの問い合わせ処理を示した図

1-4. ユースケースに共通する課題

- 目的のStateを消費することなく確認したい
  - ✓消費状況確認後、別のtxのInputStateとして使用するため

結果的に**目的のstateをInputStateとして**

消費したtxの存在を確認可能

SBI R3 Japan

参考スライド

- あるtxの当事者以外が、その**存在**を確認することは難しい
  - ✓Cordaの特徴であるプライバシーが確保されているため
- 大前提として取引の当事者が第三者(企業、Party C)にtxを渡していた
  - ✓そのtxとチェーンで繋がっているtxのみ存在検知可能
- **データはもちろん閲覧不可**
  - ✓あくまで存在検知のみ

SBI R3 Japan

SBI r3 Japan

2. Ref.stateによる解決方法

2-1. Ref.stateによる解決方法

Ref.state=Reference state

- 現状の課題
  - ✓消費状況の確認によって、stateが強制的に消費済になる事

- 解決方法
  - ✓Ref.Stateを活用により、消費されずにstateの消費状況を確認

SBI R3 Japan

補足. Ref.stateとは？

- Reference state=Ref.state

- StateAndRef, StateRefとは全く異なるモノ
  - ✓StateRef: あるstateへの参照(txHashとindex)
  - ✓StateAndRef: Stateの情報とStateRefとのペア

SBI R3 Japan

補足. Ref.stateとは？

- あるtxの参照情報を表すstate
  - ✓元stateをRef.stateとしてtxに追加
    - ➢ メソッドはaddReferenceState()

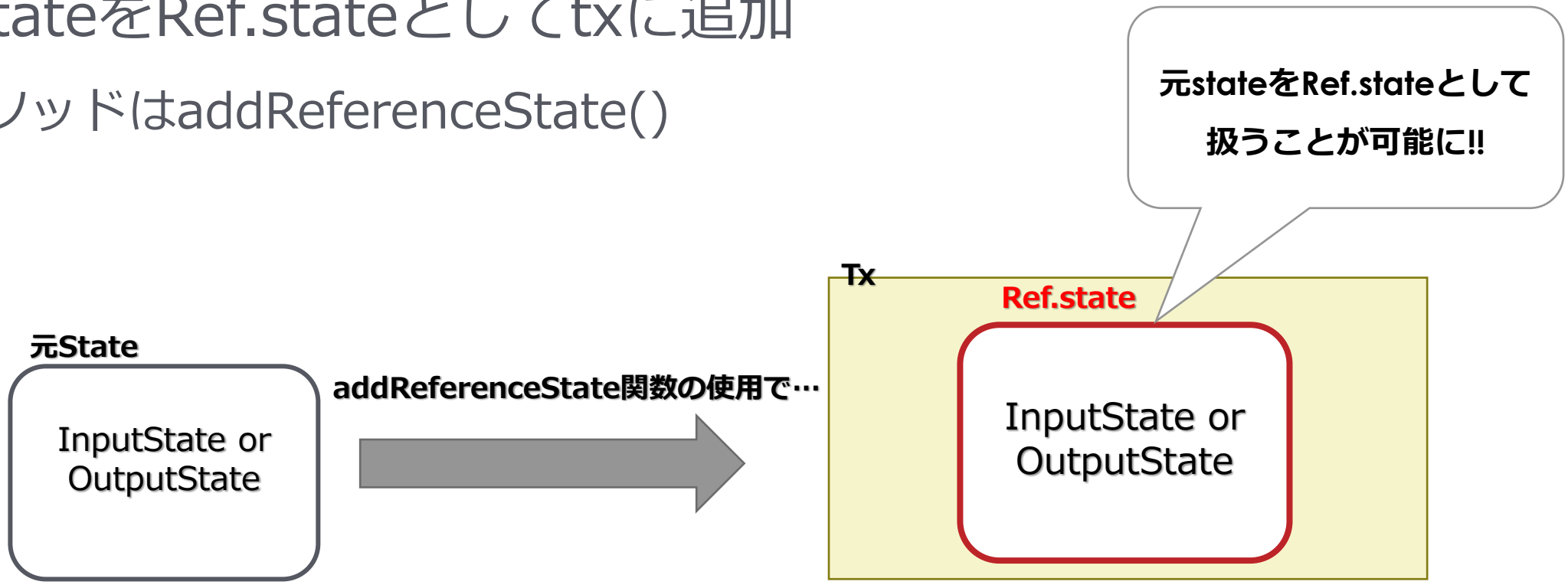

図13. Ref.stateの生成に関する図

補足. Ref.stateとは？

- 以下の図の赤枠に示す通り、参照情報として利用される

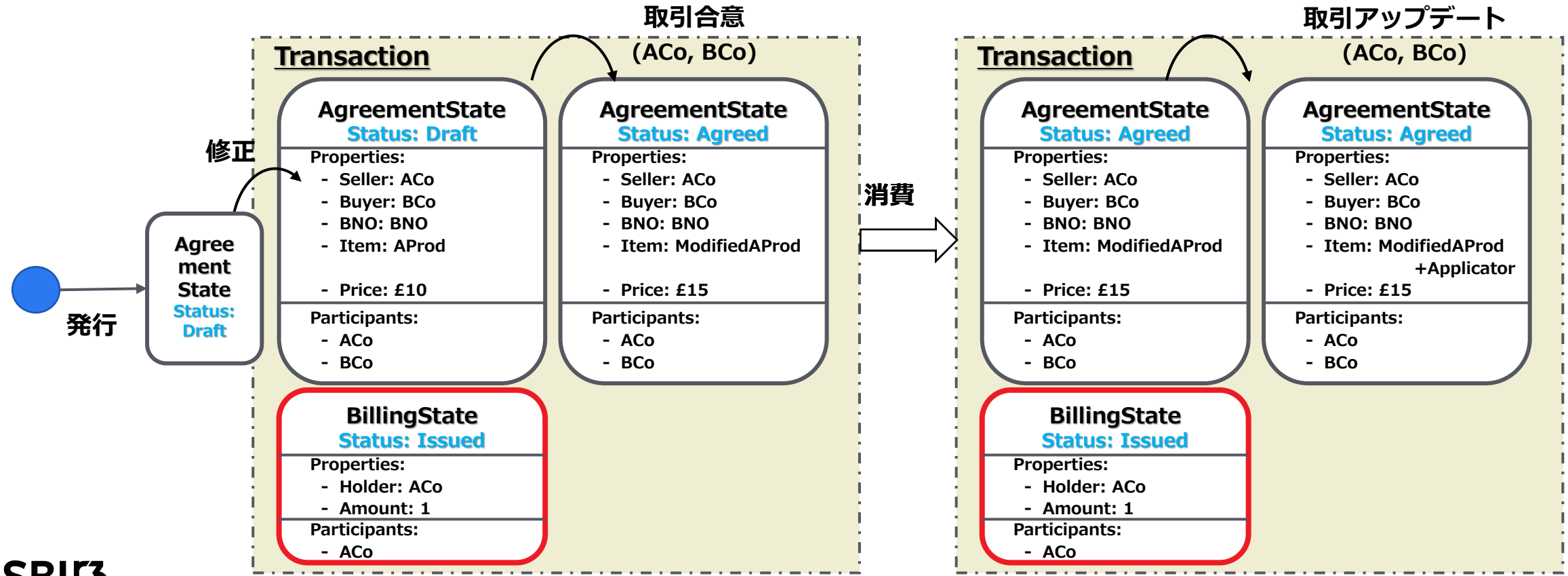

図14. あるtxにおけるRef.Stateの使用例

補足. Ref.stateの特徴

https://docs.r3.com/en/tools/cdl/ledger-evolution-view/reference-states.html

- Notaryを通しても**元stateの消費状況に影響なし**
  - ✓If 元stateが未消費→状態は未消費のまま
  - ✓If 元stateが消費済み→状態は未消費、but二重消費を検知!

図15. Ref.stateの特徴を示す図

2-2. Ref.stateを利用した消費状況の確認方法

- 元stateは未消費のまま消費状況を確認
  - ✓従来手法との大きな違い

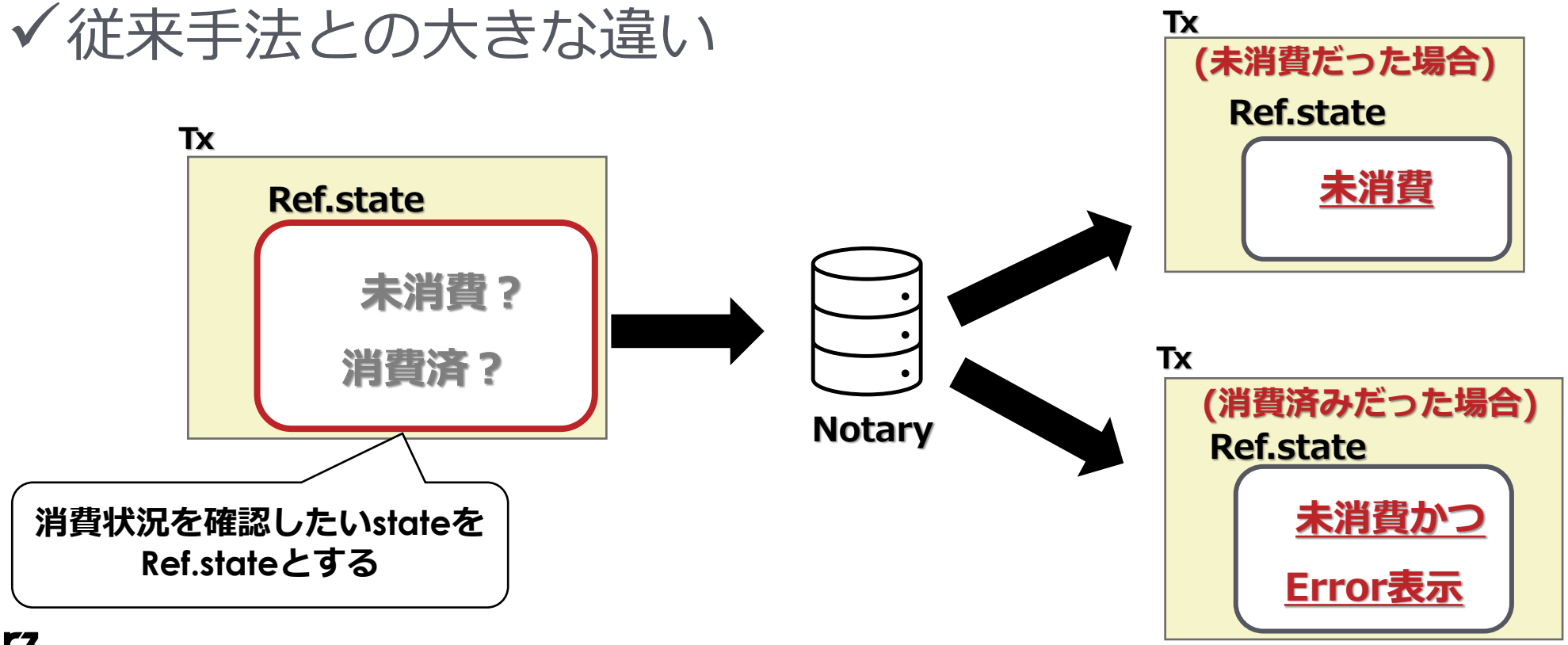

図16. Ref.Stateを用いた状態問い合わせのフロー図

SBI r3 Japan

3. 調査１：Ref.Stateのみのtxは作成可能か？

3-1. Ref.Stateのみのtxは作成可能か？-背景

- 消費状況を確認したいstateをRef.stateとしてtxに含める
  - ✓InputStateなどその他コンポーネントは必要か？

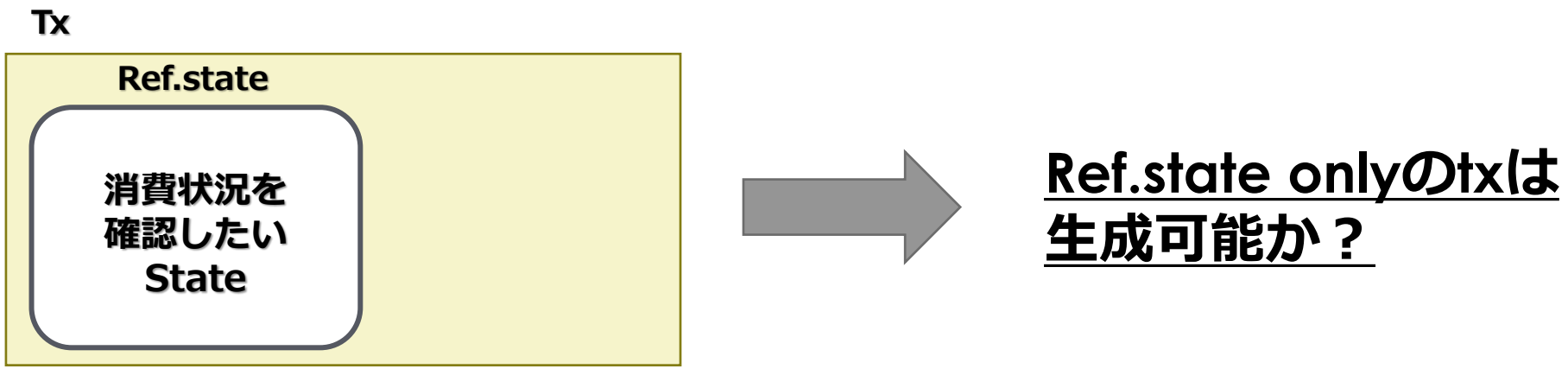

図17. 調査1の背景を示す図

3-2. Ref.Stateのみのtxは作成可能か？-方法

- Ref.stateのみのtxをNotaryに渡し、挙動を確認

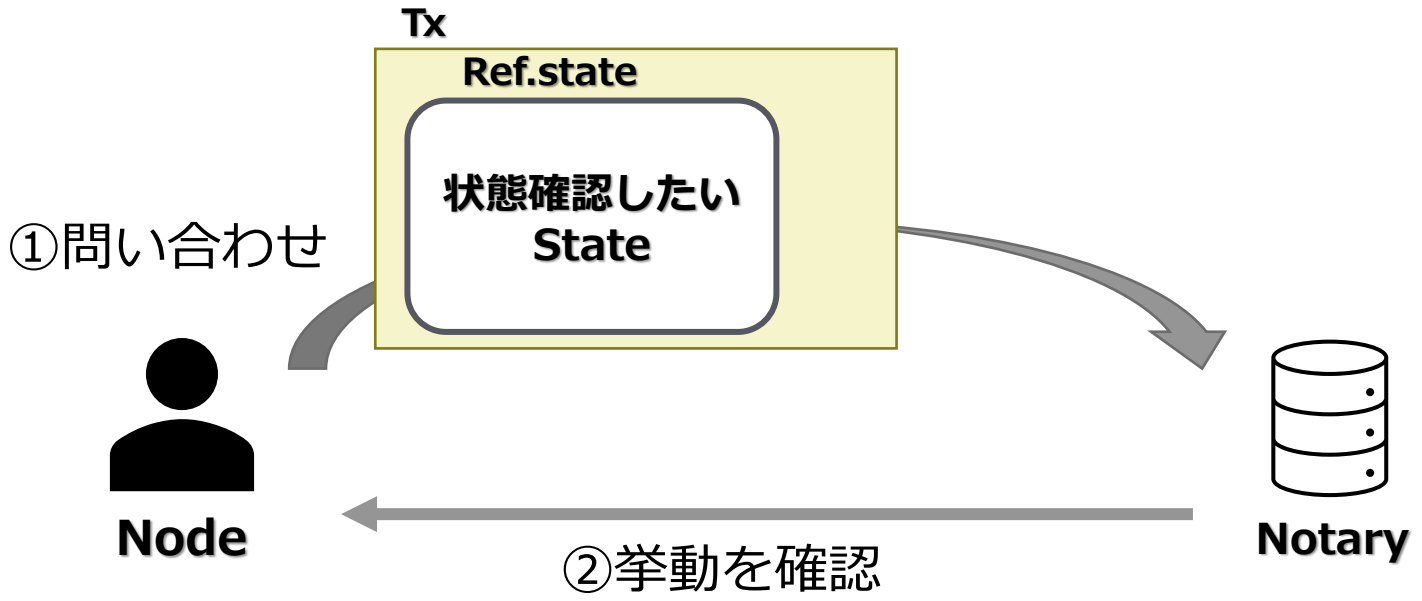

図18. 調査の概要を示す図

3-3. Ref.Stateのみtxは作成可能か？ -結果

- InputStateまたはOutputStateが空だとエラーが表示
  - ✓WireTransaction.kt(line 73)を参照

```
check(inputs.isNotEmpty() || outputs.isNotEmpty()) { "A transaction must contain at least one input or output state" }
```

WireTransaction.ktのURL:
https://github.com/corda/corda/blob/release/os/4.6/core/src/main/kotlin/net/corda/core/transactions/WireTransaction.kt#L73

SBI R3 Japan

3-4. Ref.StateのみのtxはSBI作成可能か？ -実装検討

- Ref.Stateのみのtxは作成不可能
  - InputState or OutputStateに消費されても問題ないDummy stateが必要

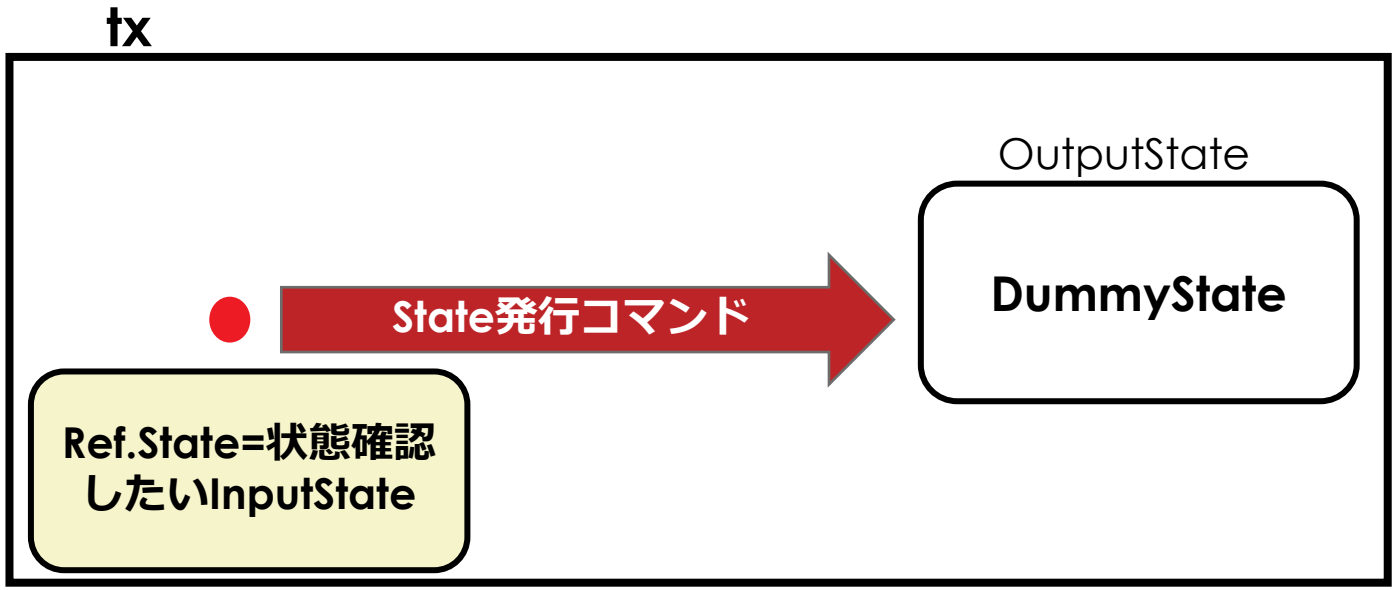

図19. Ref.stateを用いた状態問い合わせtxのモデル図

3-5. Ref.Stateのみのtxは作成可能か？ -テスト

- 未消費は正常に処理され、消費済みはハッシュ値にて通知

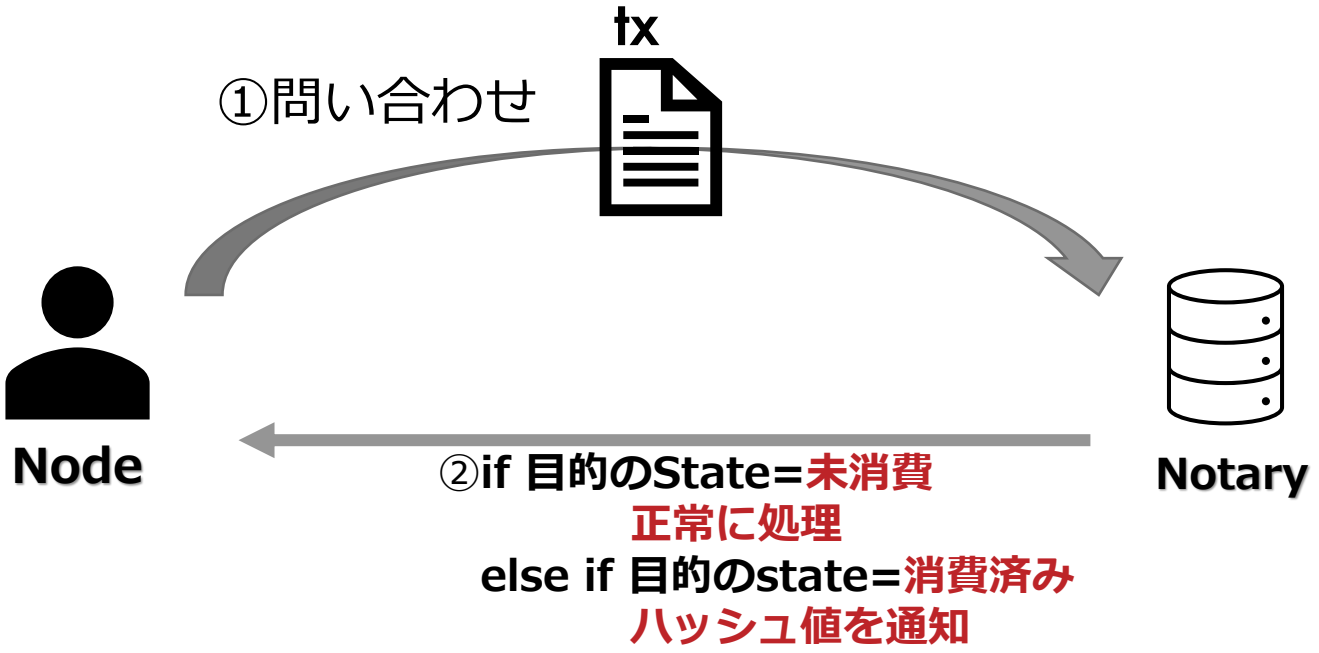

図20. Ref.stateを用いた状態問い合わせ処理のフロー図

SBI R3 Japan

4. 調査2:Notaryが検知した複数のRef.stateはログにて全て確認できるか

4-1. Notaryが検知した複数のRef.stateは全て確認できるか –背景

- 調査1のtxモデルにRef.stateを複数含めて効率的に消費状況を確認したい

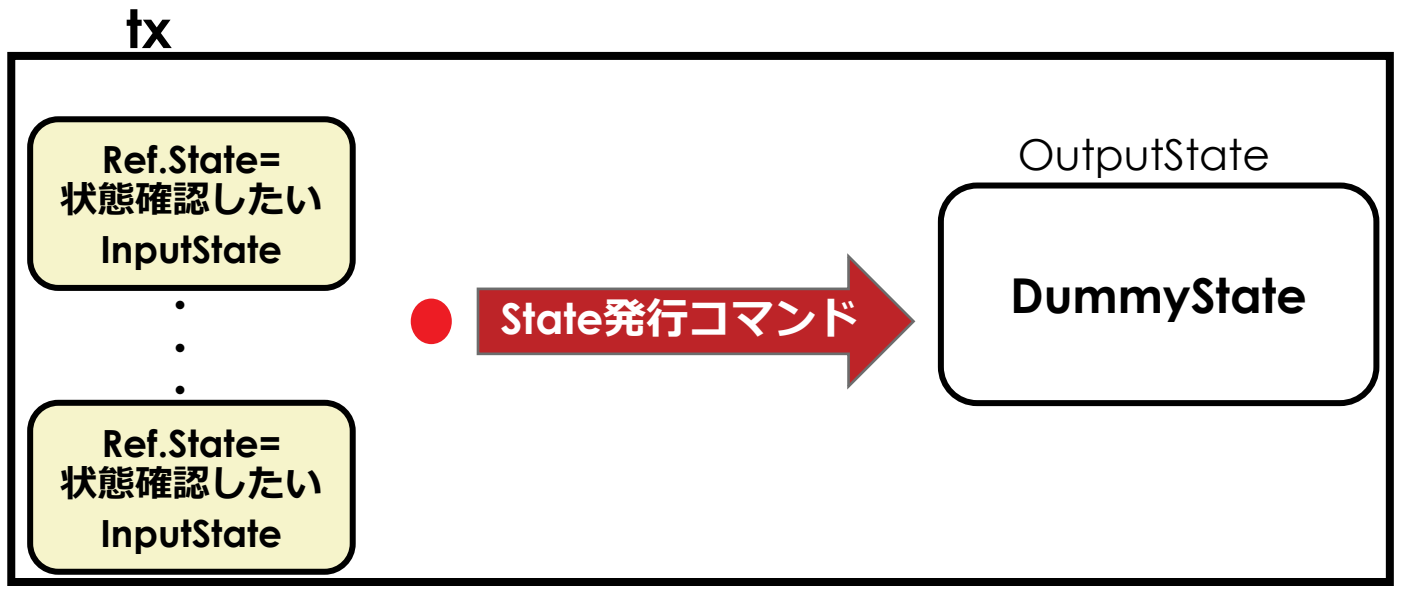

図21. 複数のRef.stateを用いた状態問い合わせtxの例

4-2. Notaryが検知した複数のRef.stateは全て確認できるか –方法

- 消費済みの元stateをRef.stateとして15個含め、挙動を確認
  - ✓Notaryがエラー検知=**“消費済み”**

SBI R3 Japan

4-3. Notaryが検知した複数のRef.stateは全て確認できるか –結果

- Ref.stateは**最大5個**表示可能
  - ✓NotaryError.kt(line 25&37)参照

```
const val NUM_STATES = 5
```

```
override fun toString() = "One or more input states or referenced states have already been used as input states in other transactions. " +
        "Conflicting state count: ${consumedStates.size}, consumption details:\n" +
        "${consumedStates.asSequence().joinToString(",\n", limit = NUM_STATES) { it.key.toString() + " -> " + it.value }}.\n" +
        "To find out if any of the conflicting transactions have been generated by this node you can use the hashLookup Corda shell command."
```

NotaryError.ktのURL:
https://github.com/corda/corda/blob/cd8cd60a0fd943198a87d266bd15b807eeaa20ba/core/src/main/kotlin/net/corda/core/flows/NotaryError.kt#L37

SBI R3 Japan

4-4. Notaryが検知した複数のRef.stateは全て確認できるか –解決策

解決策: ループで全て表示

✓目的のstateをRef.stateとして５個含める作業を繰り返し、Notaryに問い合わせ

- 表示制限にかかることなく全てのRef.stateの消費状況を把握可能

SBI R3 Japan

4-4. Notaryが検知した複数のRef.stateは全て確認できるか–解決策

- 先ほどのtxにRef.Stateを最大5個含め、Notaryへ問い合わせ
  - ✓この処理をループで繰り返す

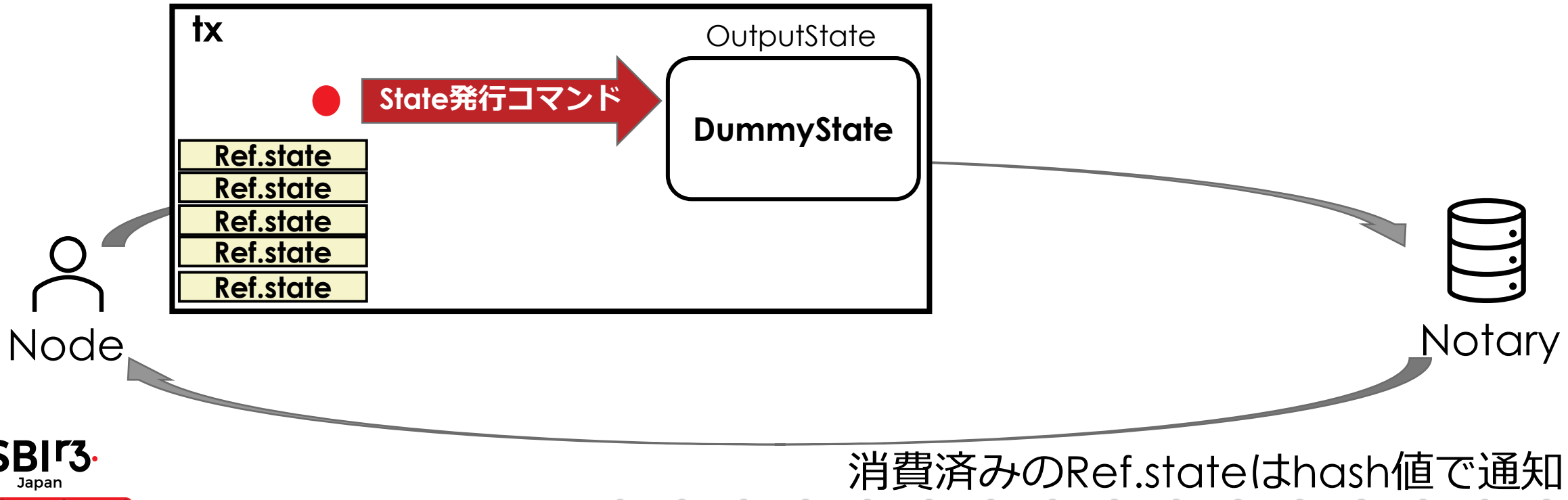

図22. ループを用いた解決方法を示す図

SBI R3 Japan

5. まとめ

5. まとめ

- 特定のstateの消費状況をNotaryに問い合わせたい要望が存在

  VC:　　企業がVCの有効性を確認したい

  日本酒:　取引相手の商品の所在を確認したい

  障害:　　NodeがtxにInputStateの消費状況を確認したい

Ref.stateを活用して手法を提案

SBI R3 Japan

5. まとめ

- Ref.stateのみのtxは作成不可能
  - ✓通常のstate(Dummy State)に紐づける必要あり

- Notaryが検知したRef.stateは最大5個まで表示可能
  - ✓元stateが二重消費と検知された全てのRef.stateを
    確認する方法としてループを使った手法が現状効果的では？

SBI R3 Japan

5. まとめ

- プライバシーを確保しつつ、透明性が向上
  - ✓目的のstateをInputStateとして消費したtxの**存在**が確認可能

Ref.stateの活用は、Cordaに新たな可能性をもたらした

SBI R3 Japan

参考文献とコード

- Corda公式ドキュメント
  - ✓ https://docs.r3.com/en/tools/cdl/ledger-evolution-view/reference-states.html
- Cordaトレーニングドキュメント
  - ✓ https://training.corda.net/corda-advanced-concepts/reference-states/#:~:text=Reference%20states%20are%20part%20of,usual%2C%20verification%20outcomes%20are%20deterministic.
- 調査を行うにあたって作成したコード
  - ✓ https://github.com/ShotaIMO/possibilityOfReference

SBI R3 Japan