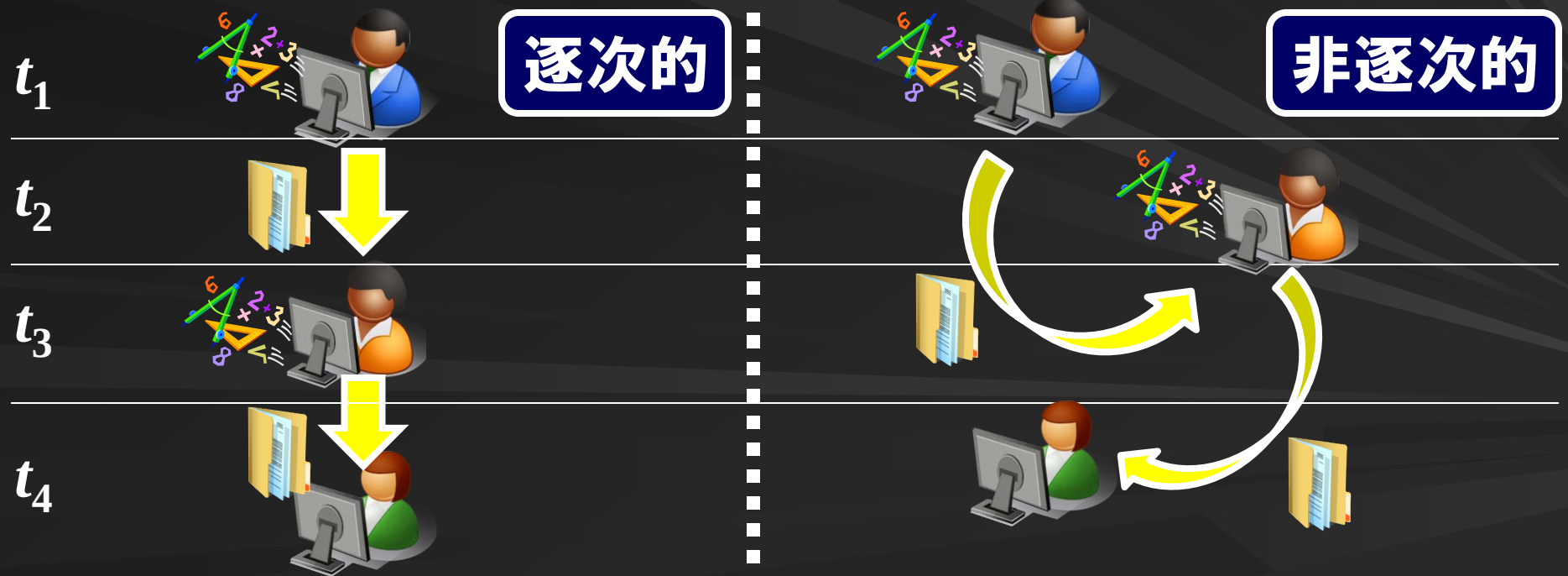
強識別不可能性理論への非逐次的スケジューリングの適用

2010/3/8

米山 一樹 (NTT)

発表内容

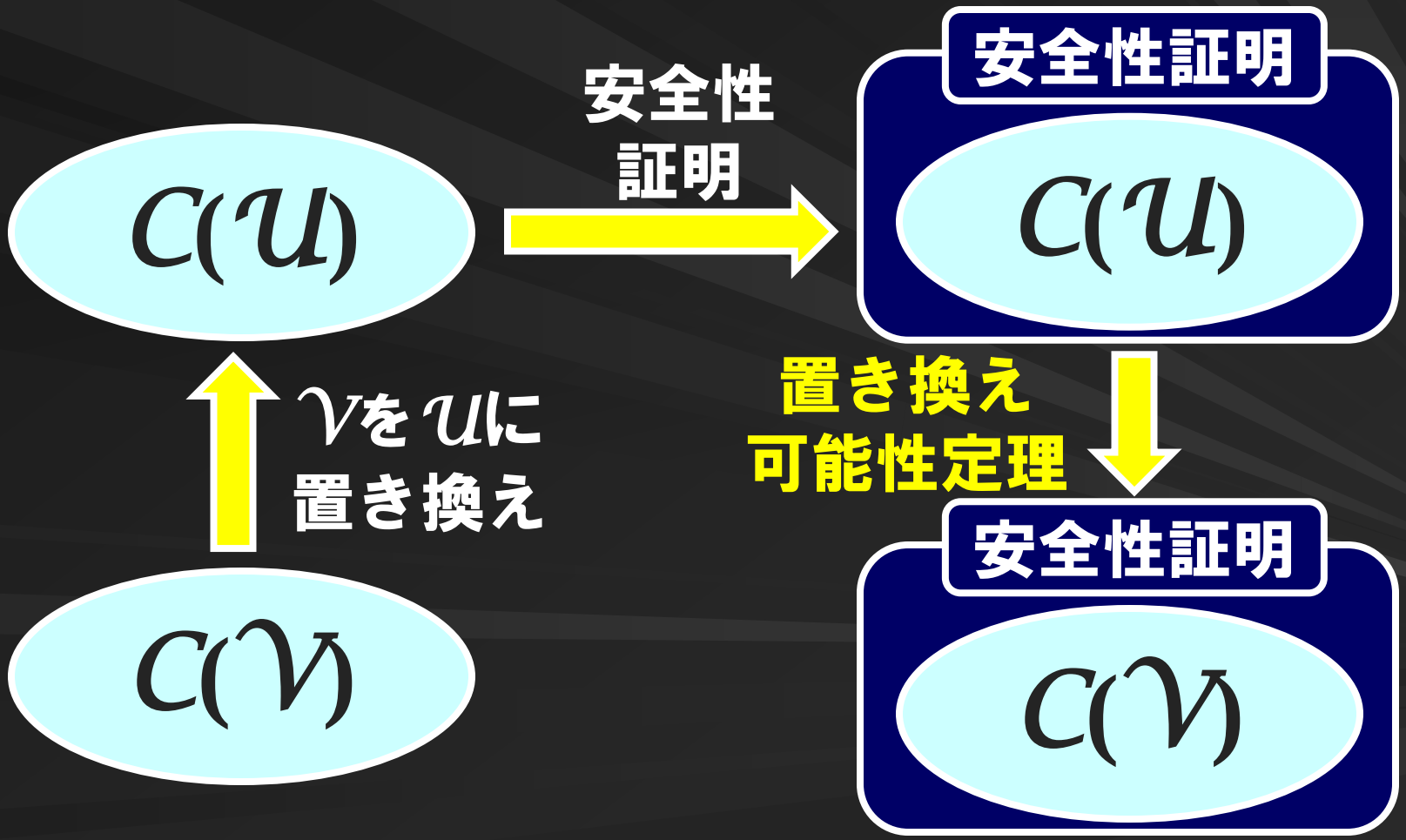
- スケジューリングの違いが**強識別不可能性フレームワーク**に与える影響の考察
  - **非逐次スケジューリング**版強識別不可能性の定義
  - **置き換え可能性**の証明
  - 逐次版と非逐次版の**定義間の関係**



背景

「置き換え」による安全性証明の簡易化

C：ある暗号システム
$\mathcal{V}$：構造が**複雑**な資源（プロセス）
$\mathcal{U}$：構造が**単純**な資源（プロセス）



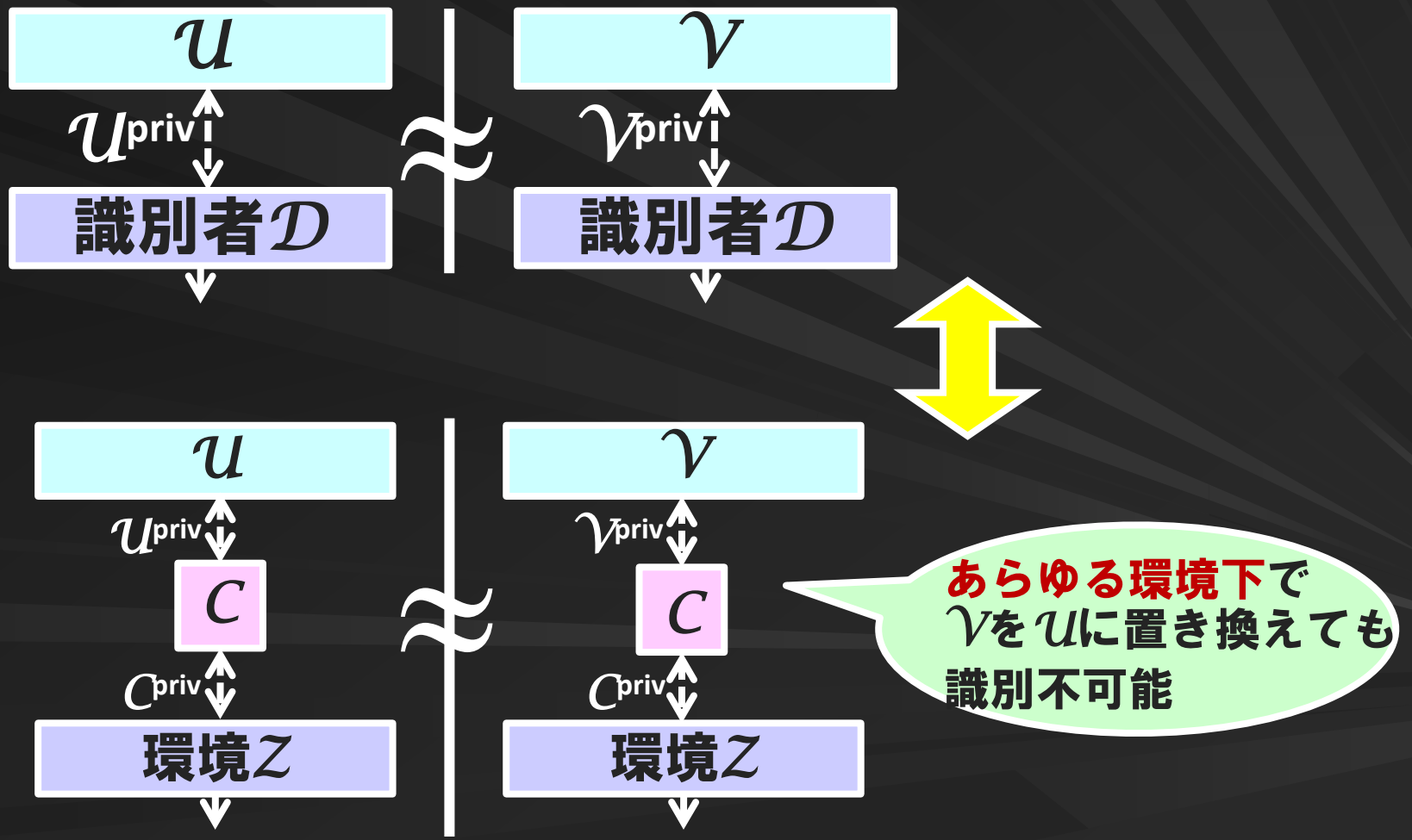
ランダムオラクルを用いた置き換え

- ハッシュ関数を**ランダムオラクル(RO)**として置き換え
  - 「置き換え可能性が成り立てば」，安全性証明が流用可能

否定的結果[CGH98]：ROは現実の関数との置き換え可能性を満たさない

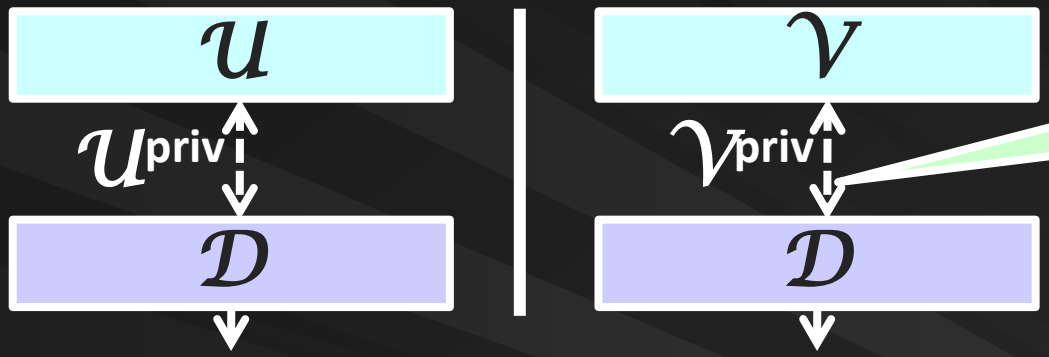
置き換え可能性の証明

- 資源$\mathcal{U}$が資源$\mathcal{V}$との**識別不可能性**を満たすならば，$\mathcal{V}$は$\mathcal{U}$と置き換え可能である．



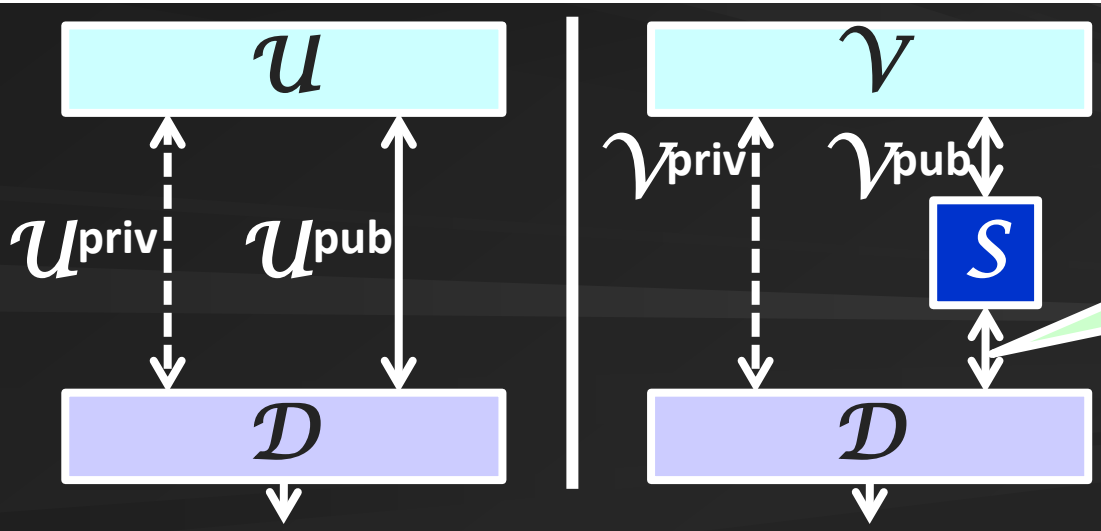
識別不可能性と強識別不可能性

識別不可能性（indistinguishability）



識別者はhonestなやり取りだけを見て識別

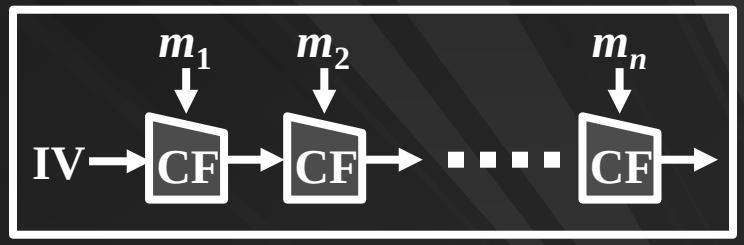
強識別不可能性（indifferentiability）[MRH04]

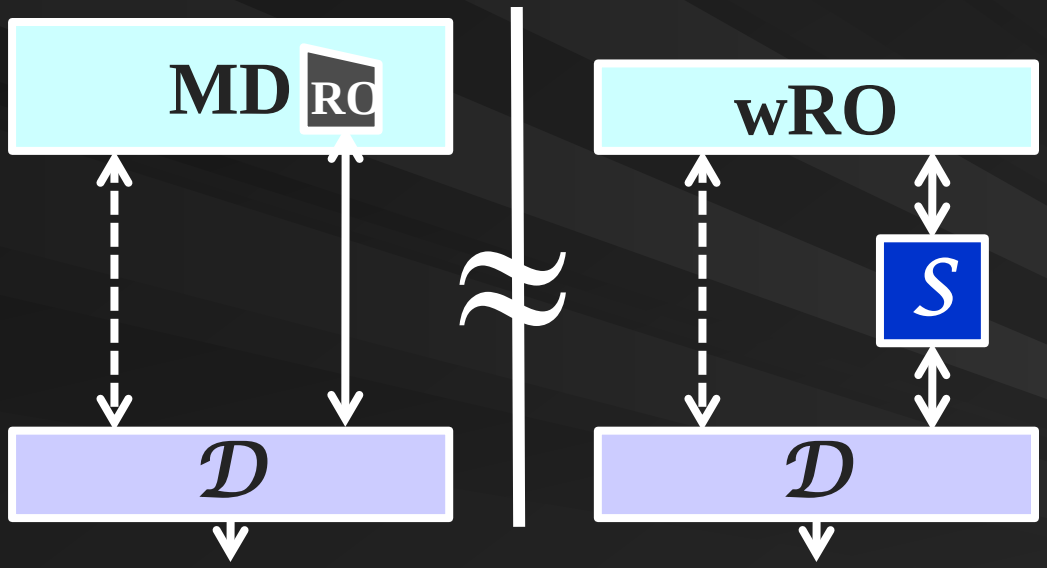


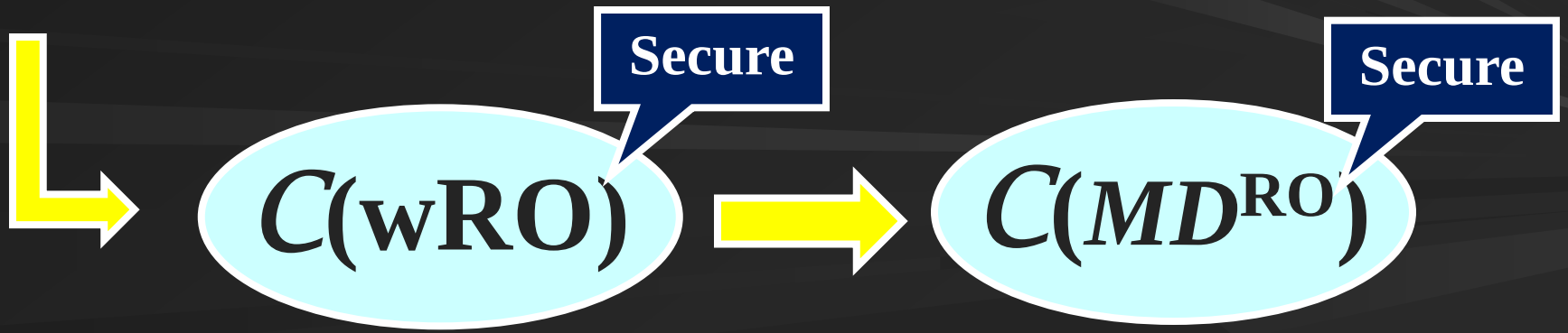
識別者は攻撃者用I/Fも観察可能

強識別不可能性による置換えの例

- Merkle-Damgårdハッシュと弱いROの置き換え可能性 [NYWO09]

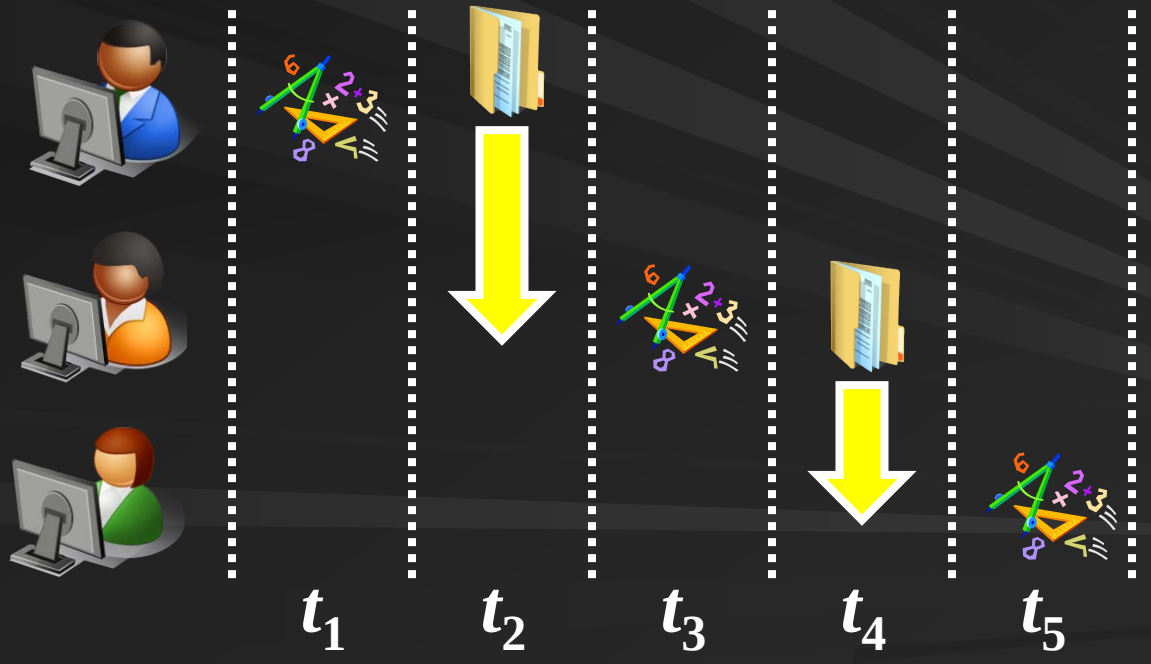






スケジューリングモデル

- 強識別不可能性フレームワークでは**逐次スケジューリング**を仮定
  - 一度にアクティブになるプロセスは**一つ**
  - **メッセージを送る**ことで次のプロセスをアクティブ化



逐次スケジューリングの問題

- 大規模システムでは**実行順の制御**は難しい
  - e.g. 多人数が関わる暗号プロトコル（電子投票など）
- 実装時に効率化のために**予期せぬ並列実行**が行われるかもしれない
  - e.g. メッセージを受け取る前に乱数生成

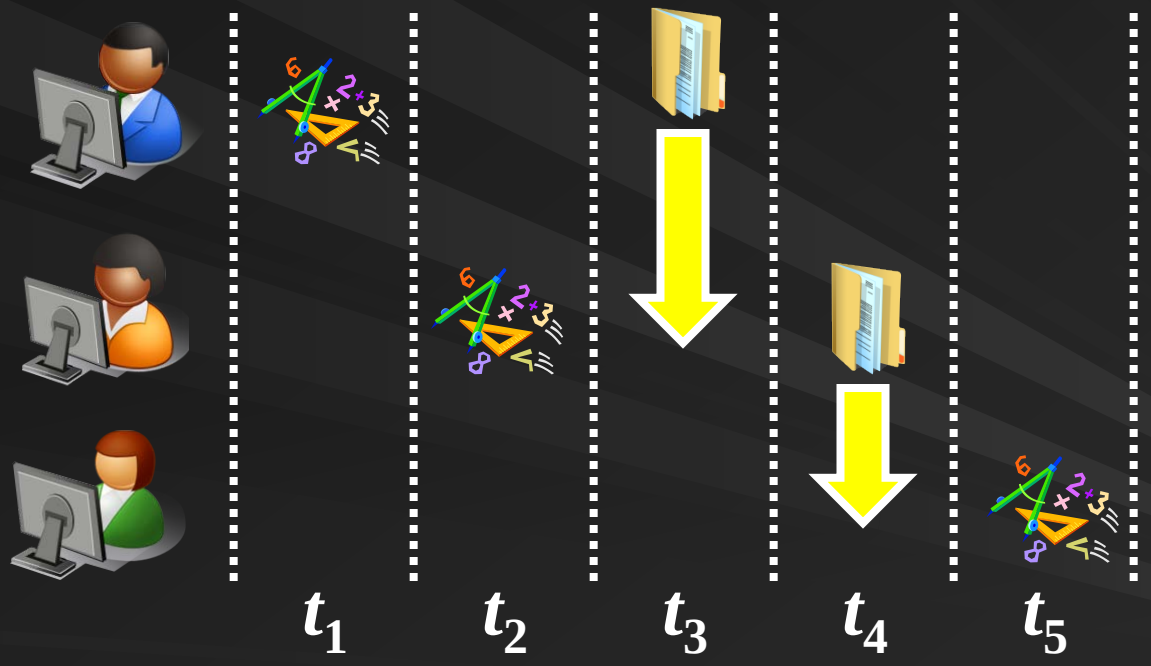
逐次スケジューリングでは上記のような状況下での安全性を厳密には捉えられない

↓

非逐次スケジューリングが必要

非逐次スケジューリング

- メッセージの授受とは独立にプロセスがアクティブ化されうる



タスクPIOA
確率的多項式時間プロセス計算

今日の目次

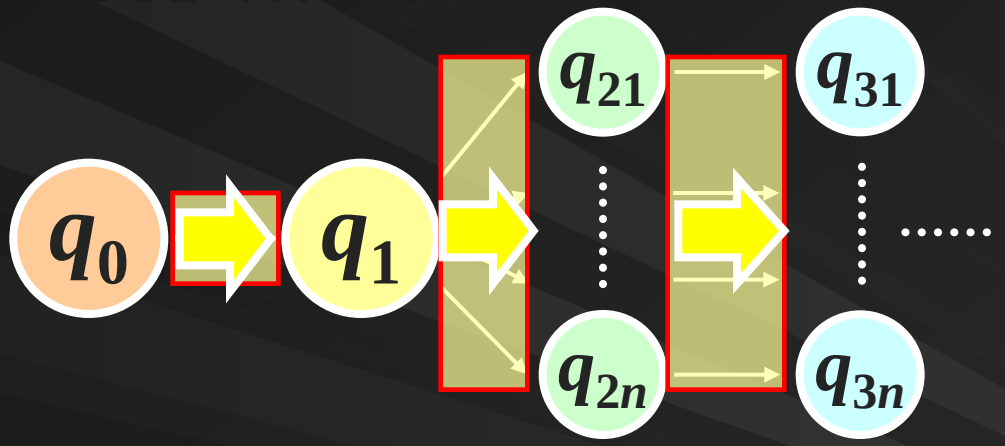
- 非逐次スケジューリング版強識別不可能性
  - タスクPIOAによる非逐次スケジューリングの表現
  - 定式化
  - 置き換え可能性定理
- 逐次版定義と非逐次版定義の関係
  - 逐次版モデル $\not\to$ 非逐次版モデル
  - 非逐次版モデル $\not\to$ 逐次版モデル

  } 互いに独立
- まとめ

非逐次スケジューリング版強識別不可能性の定式化

タスクPIOAにおける非逐次スケジューリング

- タスク：出力アクションと内部アクションを同値類でまとめたもの



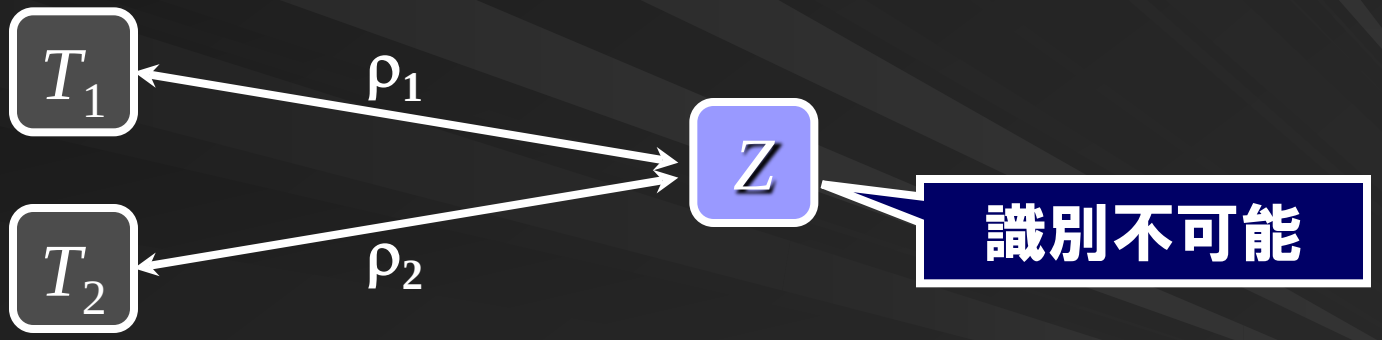
- **タスクスケジューラ**によって非逐次スケジューリングを表現

$$\rho = task_1 .task_2 .task_3 ....$$

事前条件の範囲内で
全ての列をとりうる

システム間の実現関係

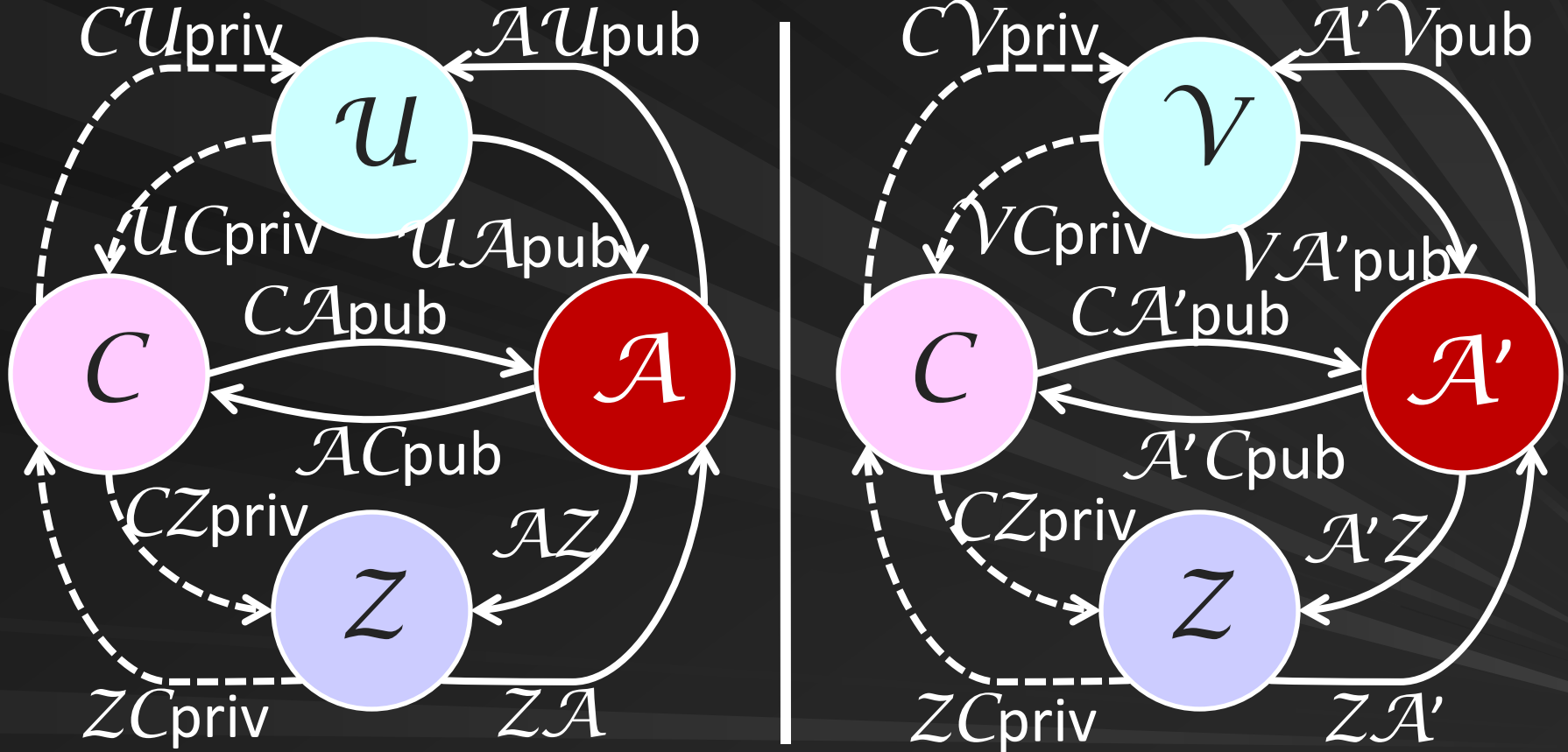
- タスクPIOA間の**実現関係**：$T_1 \leq T_2$
  - ∀ 環境 Z and ∀ タスクスケジューラ ρ_1 of T_1, ∃ タスクスケジューラ ρ_2 of T_2 s.t. ρ_1実行時の$T_1||Z$の出力の確率分布とρ_2実行時の$T_2||Z$の出力の確率分布の差が無視できる



「≤」を用いて安全性を記述する

暗号システムの安全性（$C(\mathcal{U}) \succ C(\mathcal{V})$）

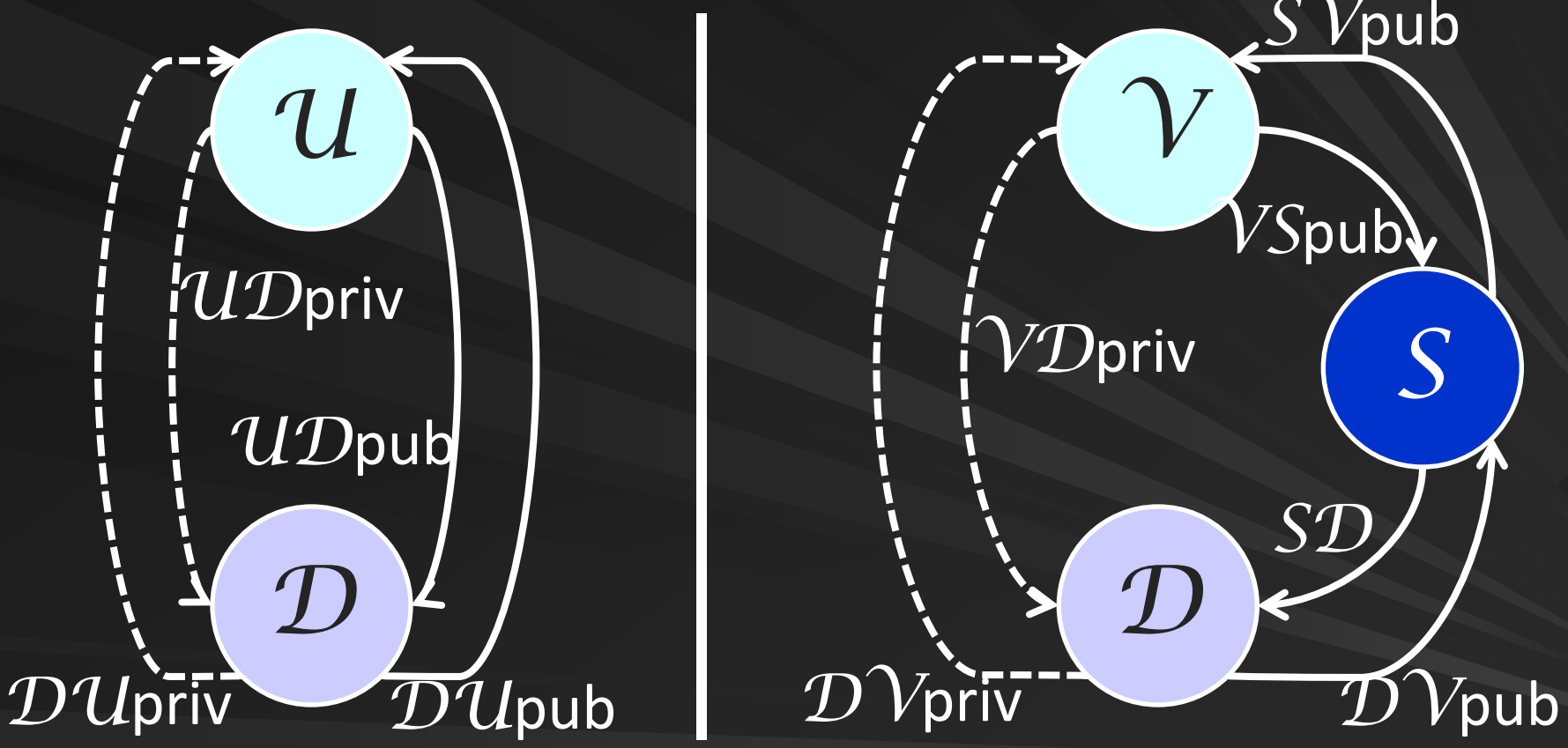
- 全てのプロセスをタスクPIOAとして表現



$\mathcal{U}||C||\mathcal{A}||Z \leq \mathcal{V}||C||\mathcal{A}'||Z$ならば$C(\mathcal{U})$は$C(\mathcal{V})$より安全

非逐次版強識別不可能性（$\mathcal{U} \sqsubset \mathcal{V}$）

- 同様にタスクPIOAで表現



$\mathcal{U}||\mathcal{D} \leq \mathcal{V}||S||\mathcal{D}$ならば$\mathcal{U}$は$\mathcal{V}$と強識別不可能

非逐次版の置き換え可能性

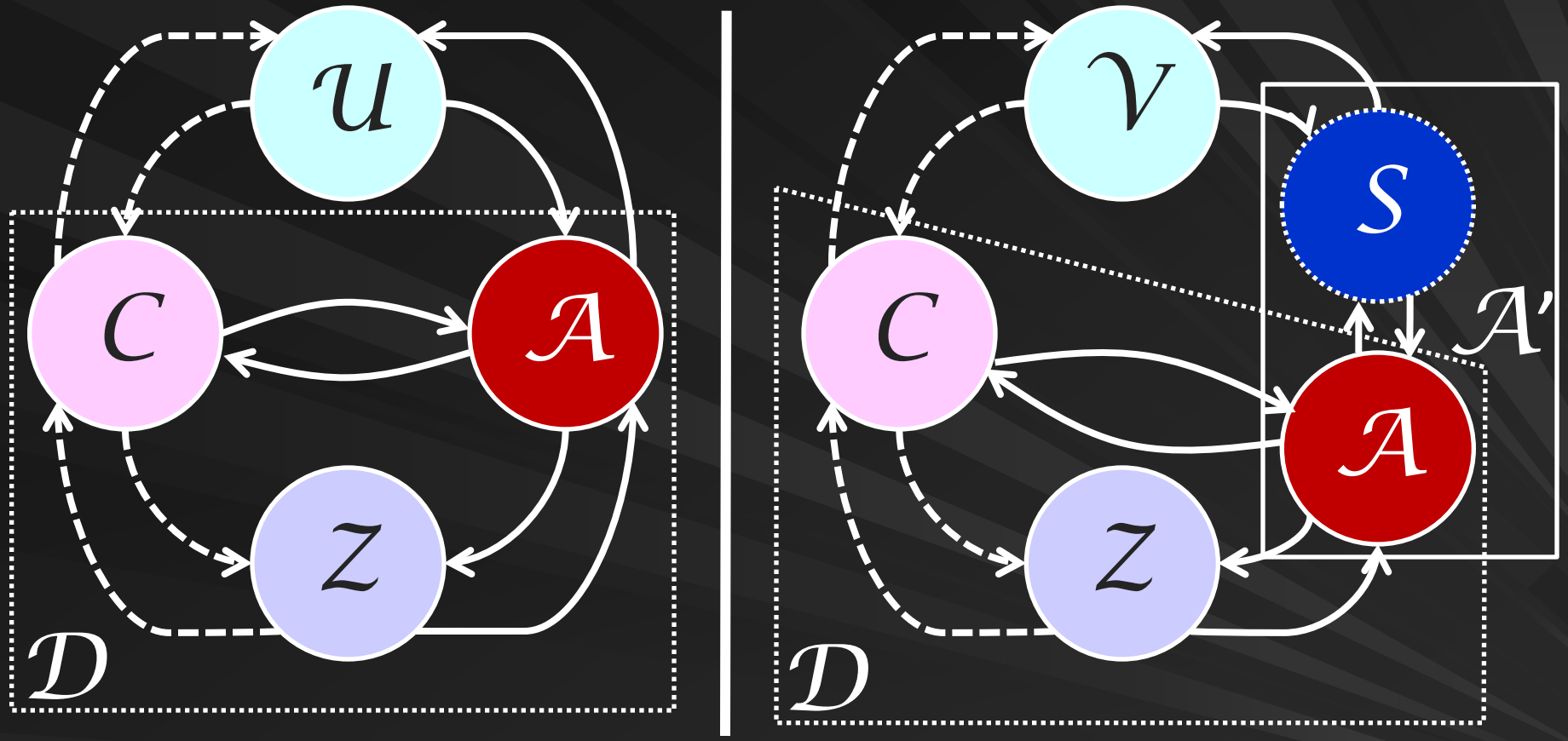
- $\mathcal{U}$が$\mathcal{V}$と強識別不可能であるとき，$C(\mathcal{U})$が$C(\mathcal{V})$より安全となれば$\mathcal{V}$は$\mathcal{U}$で置き換え可能

定理

$$\mathcal{U} \sqsubset \mathcal{V} \leftrightarrow \forall C : C(\mathcal{U}) \succ C(\mathcal{V})$$

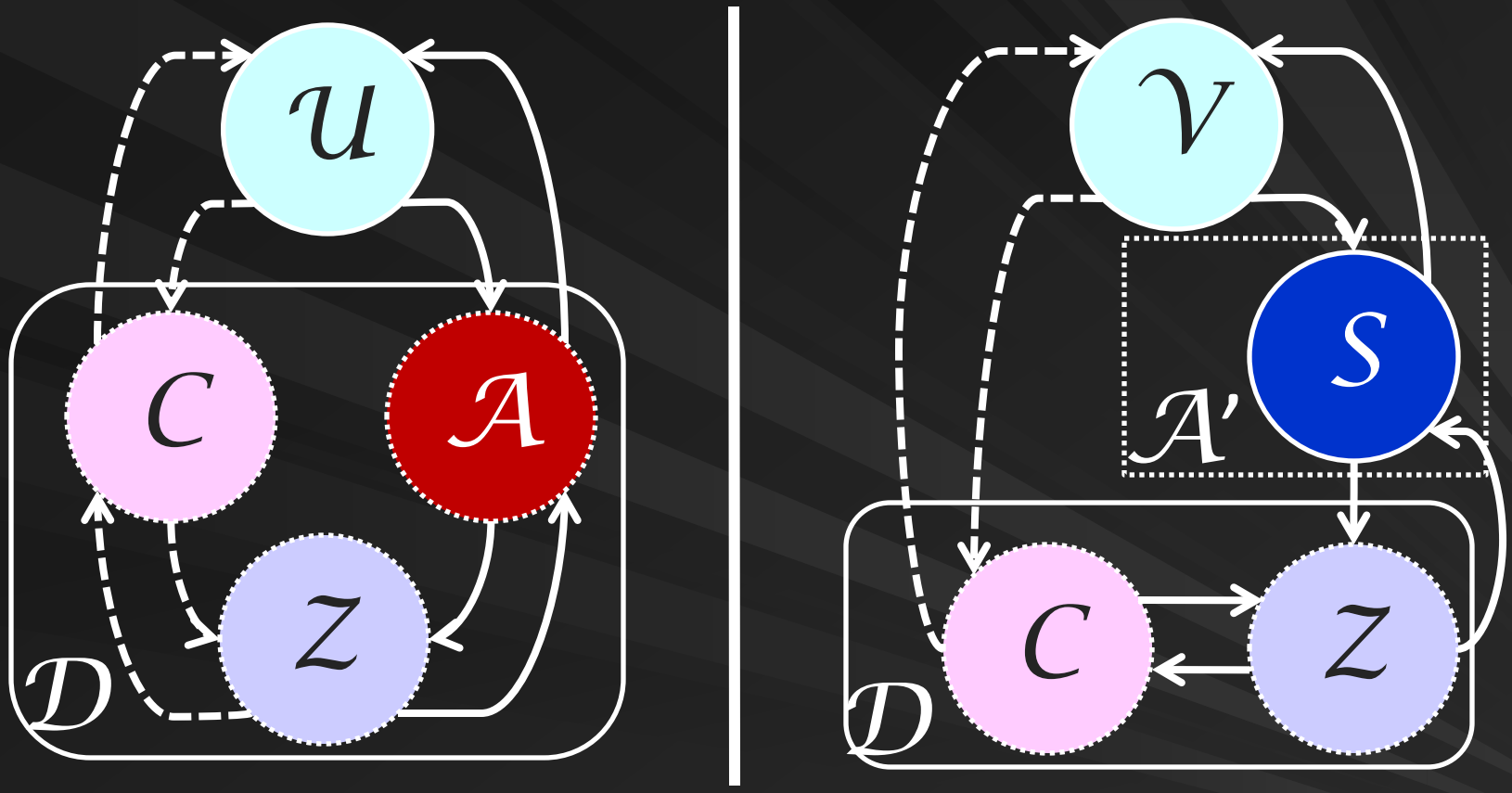
非逐次スケジューリング時も置き換え可能性は保証される

$\mathcal{U} \sqsubset \mathcal{V} \rightarrow C(\mathcal{U}) \succ C(\mathcal{V})$の証明



$$\mathcal{U}||\mathcal{D} \leq \mathcal{V}||S||\mathcal{D}$$
$$\mathcal{U}||C||\mathcal{A}||\mathcal{Z} \leq \mathcal{V}||S||C||\mathcal{A}||\mathcal{Z}$$
$$\mathcal{U}||C||\mathcal{A}||\mathcal{Z} \leq \mathcal{V}||C||\mathcal{A}'||\mathcal{Z}$$

$C(\mathcal{U}) \succ C(\mathcal{V}) \rightarrow \mathcal{U} \sqsubset \mathcal{V}$の証明



$\mathcal{U}||C||\mathcal{A}||Z \leq \mathcal{V}||C||\mathcal{A}'||Z$

$\mathcal{U}||\mathcal{D} \leq \mathcal{V}||\mathcal{A}'||\mathcal{D}$

$\mathcal{U}||\mathcal{D} \leq \mathcal{V}||S||\mathcal{D}$

逐次版定義と
非逐次版定義の関係

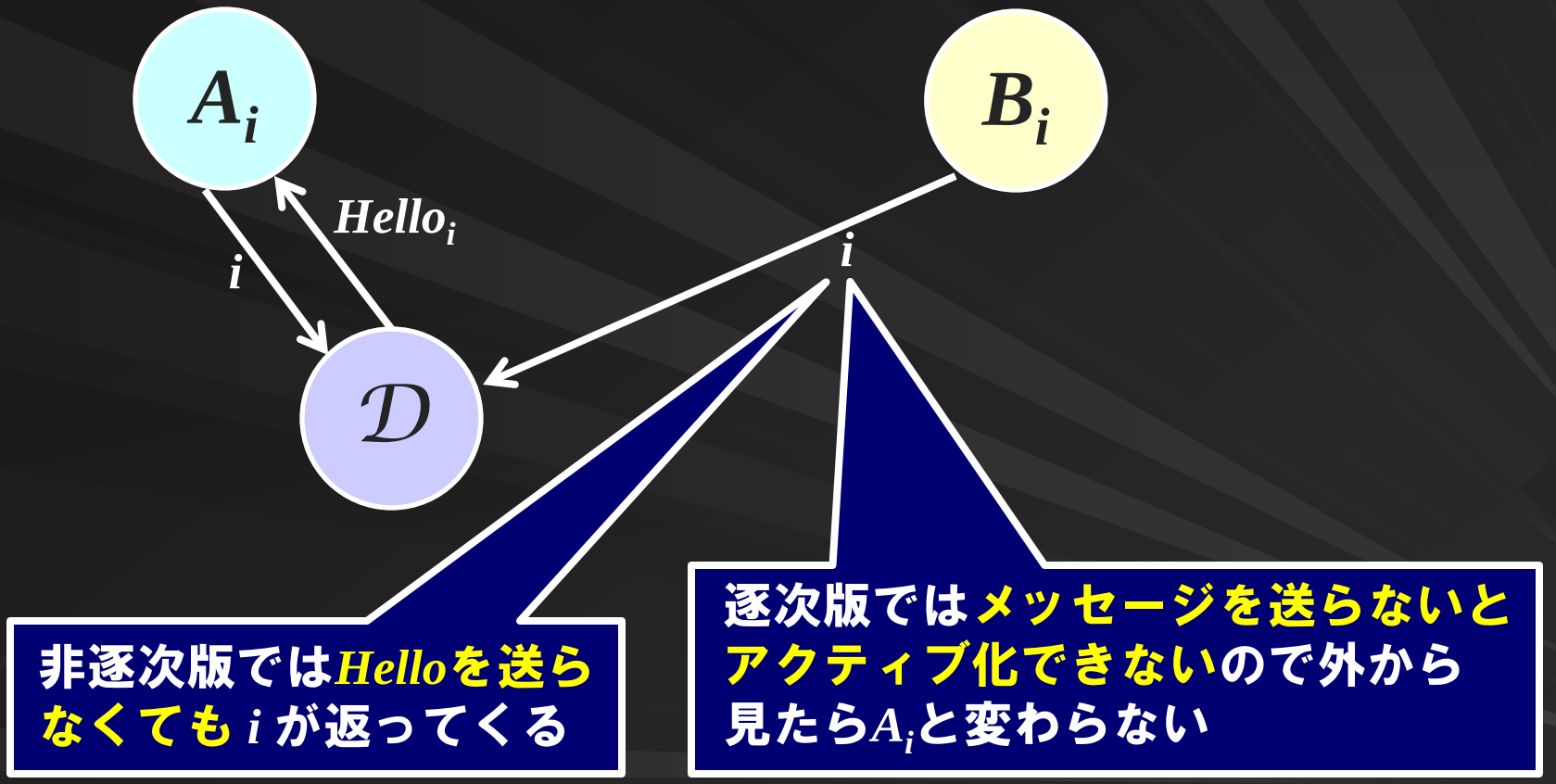
定義の隔離

- 逐次版定義 $\not\rightarrow$ 非逐次版定義
  - アクティブ化順序の違いを利用
  - 逐次スケジューリング：強識別不可能
  - 非逐次スケジューリング：識別攻撃が存在
- 非逐次版定義 $\not\rightarrow$ 逐次版定義
  - 攻撃者の能力の違いを利用
  - 逐次スケジューリング：識別攻撃が存在
  - 非逐次スケジューリング：強識別不可能

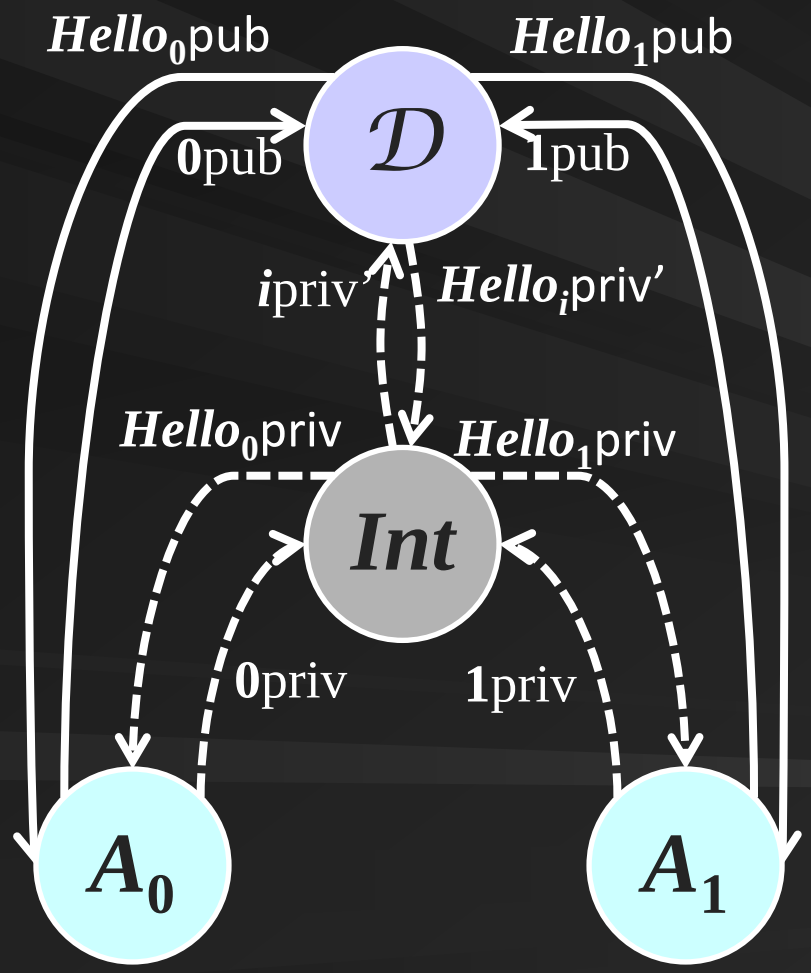
逐次版と非逐次版スケジューリングにおける強識別不可能性は互いに独立である

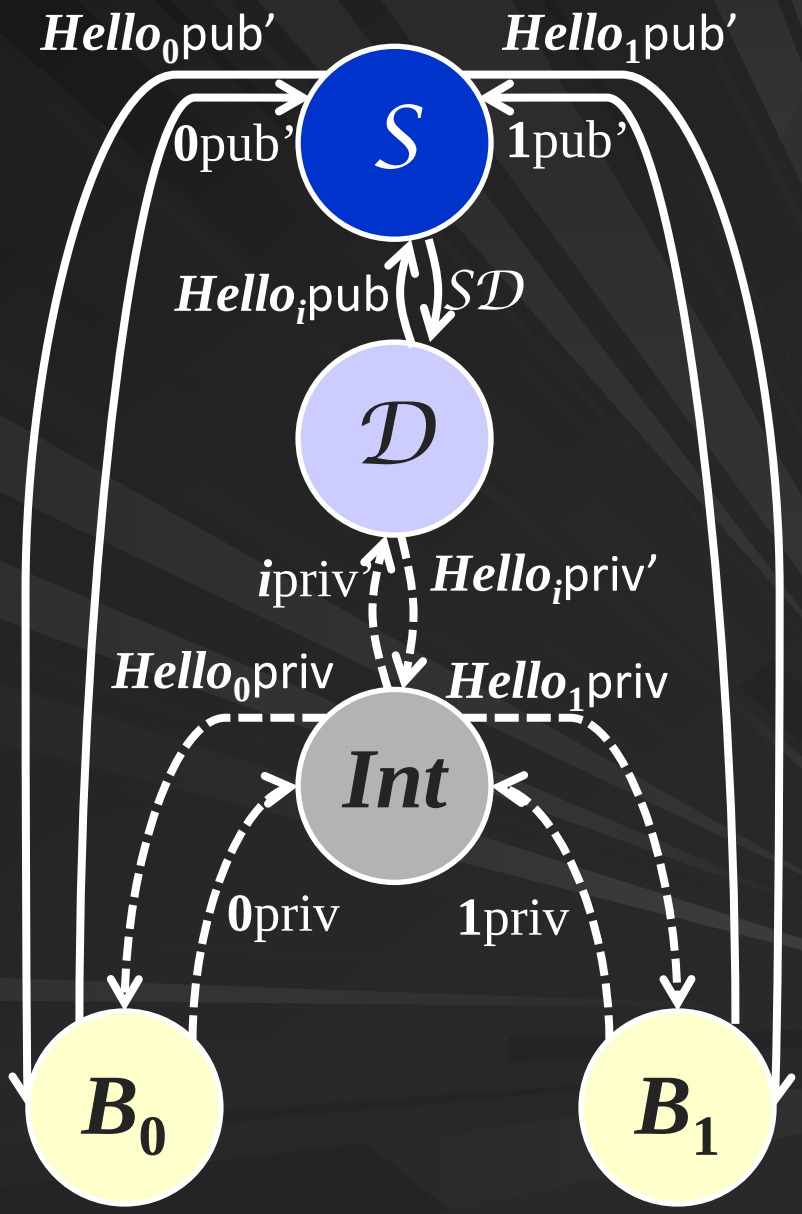
逐次版定義 ↛ 非逐次版定義

- Answer PIOAとBeacon PIOA



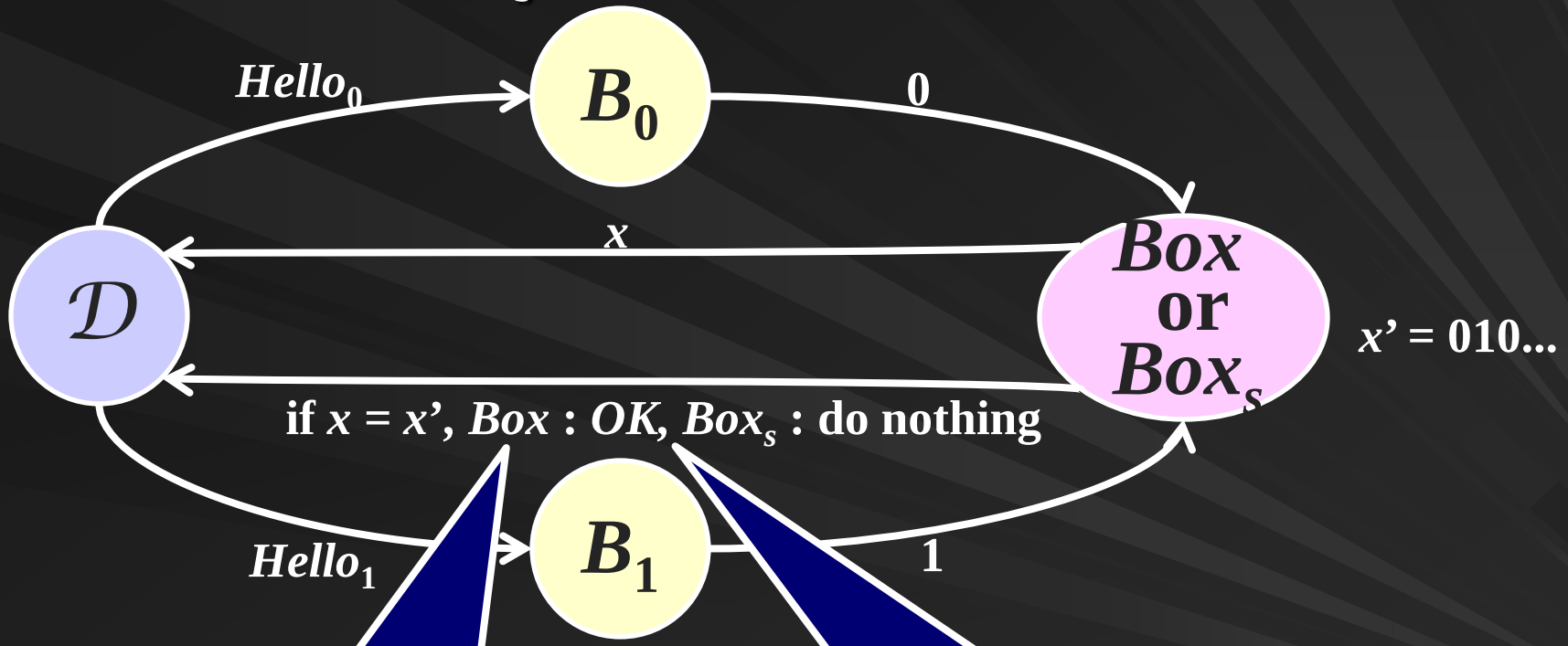
逐次版定義 ↛ 非逐次版定義





非逐次版定義 ↛ 逐次版定義

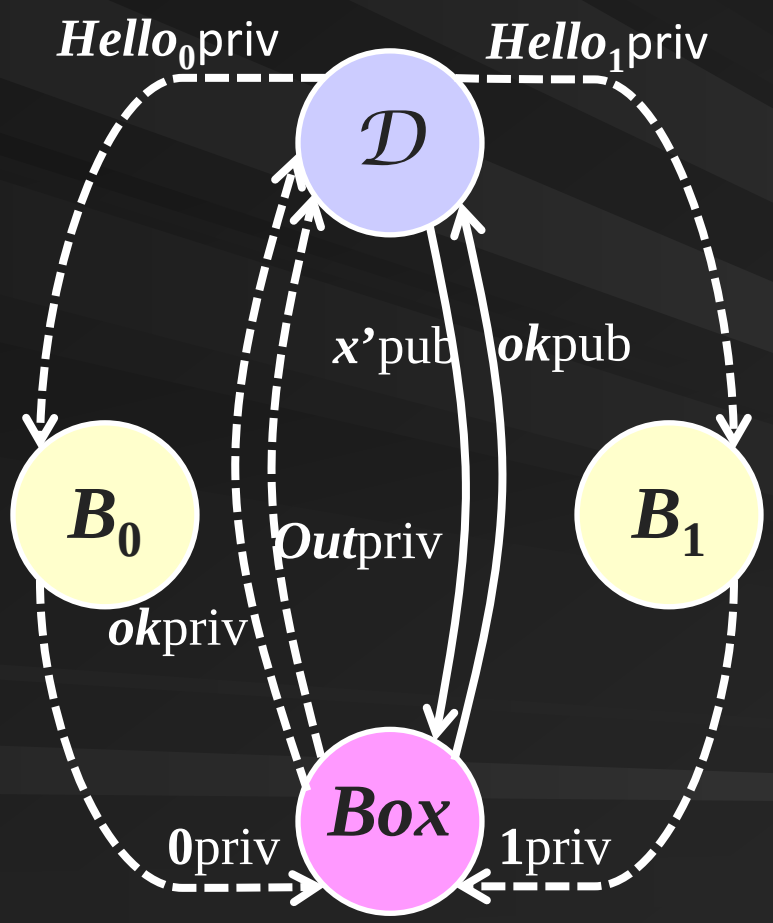
- Box PIOAとBox$_s$ PIOA

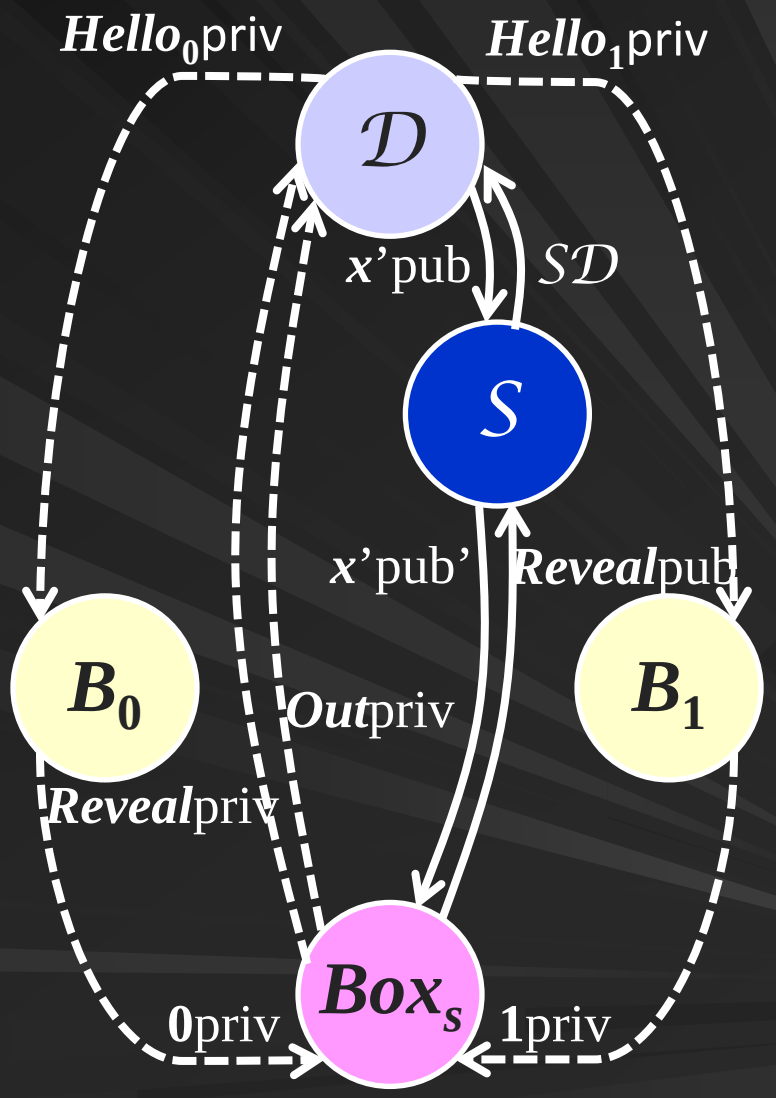


非逐次版では$x = x'$となるスケジューラは無視できる確率でしか実行時間内に発生しない

逐次版では$x = x'$となるようにBeaconをアクティブ化する識別者は確率1で識別できる

非逐次版定義 ↛ 逐次版定義





なぜ非逐次版 ↛ 逐次版が起きうるのか？

- 非逐次スケジューリングでは**プロセスの順序が制御不可能**

攻撃者にとっても！

- 一方，逐次スケジューリングでは**攻撃者による実行順序の制御**を許容

まとめ

- 非逐次スケジューリング版強識別不可能性
  - **非逐次実行時**の暗号システムに対する置き換え可能性の議論を可能にした
- 逐次版と非逐次版定義の**隔離**
  - **用途に応じて**適切な方を使う必要あり
  - シミュレーションパラダイムにおける結果[CCLP07]と同様
- 帰着可能性(reducibility)も同様に示すことが可能