モデ脳
～モデリング力を鍛えよう～

東京大学
吉田 塁

モデ之進

目次

- 自己紹介
- モデリングを詳しく知る前に
  - 現代人が主に使う能力
  - モデリングで鍛えられる能力
- モデルに慣れる（モデルの穴埋め）
- モデリング能力向上のための方法
  - 0 からモデリング
  - 既存モデルの添削

モデ脳 = 現象を簡単にモデリングできる脳力

目次

- 自己紹介

- モデリングを詳しく知る前に
  - 現代人が主に使う能力
  - モデリングで鍛えられる能力
- モデルに慣れる（モデルの穴埋め）
- モデリング能力向上のための方法
  - 0 からモデリング
  - 既存モデルの添削

モデ脳 = 現象を簡単にモデリングできる脳力

自己紹介

【名前】吉田 塁（よしだ るい）

【所属】東京大学大学院 新領域創成科学研究科
博士後期課程

【好きなもの・こと】

- プログラミング
  - 学部時代は**画像処理**の研究、モデリングの重要性をここで感じ、タイミングよく**モデ脳**と出会う
- 研究
  - 現在、**生体医工学（医学+工学）**に関する研究に従事
- 教育
  - **高等教育システムを改善**しようと日々精進中

目次

- 自己紹介
- モデリングを詳しく知る前に
  - 現代人が主に使う能力
  - モデリングで鍛えられる能力
- モデルに慣れる（モデルの穴埋め）
- モデリング力向上のための方法
  - 0 からモデリング
  - 既存モデルの添削

モデ脳 = 現象を簡単にモデリングできる脳力

現代人が主に使う能力

Input → Process → Output

（入力）　（処理）　（出力）

- Input（入力）

  情報を収集する（信頼性、新古）　（× 論文引用で Yahoo 知恵袋）

- Process（処理）

  情報をまとめる（要素の抽出、整理）　（× 全ての情報を載せる）

- Output（出力）

  情報を表現する（配置、大きさ）　（× 重要度と大きさが反比例）

現代人が主に使う能力

目的意識

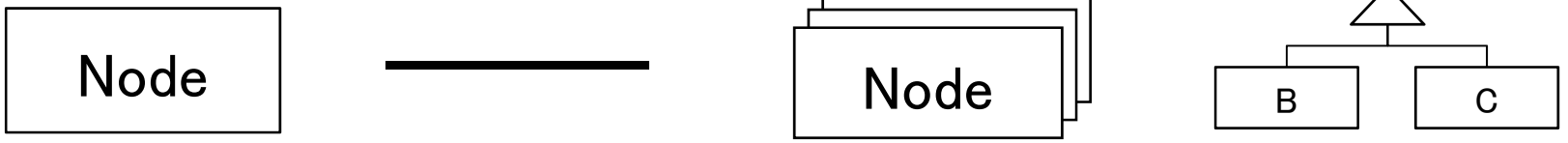
Input（入力） → **Process（処理）** → **Output（出力）**

- **Input（入力）**
  情報を収集する（信頼性、新古）　（× 論文引用で Yahoo 知恵袋）
- **Process（処理）**
  情報をまとめる（要素の抽出、整理）　（× 全ての情報を載せる）
- **Output（出力）**
  情報を表現する（配置、大きさ）　（× 重要度と大きさが反比例）

モデ脳*とは

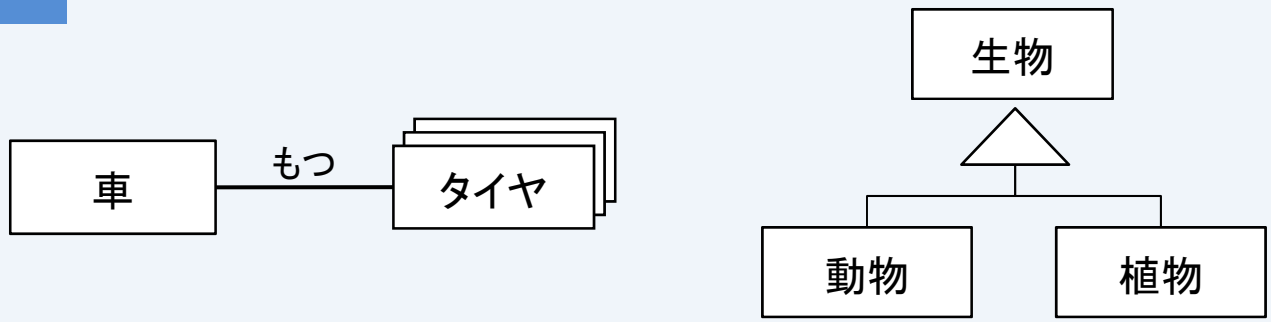
* UMLモデリング推進協議会(以後UMTP)が考案

- Unified Modeling Language(UML)を簡略化した表記法を用いる



「もの」　「つながり」　「もの」の複数形　「もの」の分類

例



モデ脳で鍛えられる能力

目的意識

Input → Process → Output

- **Input（入力）**

  情報を収集する（信頼性、新古）　（× レポートで Yahoo 知恵袋）

- **Process（処理）**

  ★ **情報をまとめる（要素の抽出、整理）**　（× 全ての情報を載せる）

  ★★ **<u>情報を抽象化し、具体化する［追加］</u>**

- **Output（出力）**

  ★ **情報を表現する（配置、大きさ）**　（× 重要度と大きさが反比例）

情報を抽象化・具体化する重要性

- 会社の研修
  - 新入社員による学びのまとめ
    - ソフトウェア A の ～～ という使い方を学んだ
  - 期待された学びのまとめ
    - サーバの仕組み、動作を実機を使って学んだ
- リーンスタートアップ
  - リーン生産方式
    - 最低限のコストで最大限の生産性

      ex.) TOYOTA のジャストインタイム生産方式
  - リーンスタートアップ
    - 最低限のコストで最重要な仮説の定量的検証

      ex.) MVP でコンバージョンレートの評価

      (Minimum Viable Product)

関西大学での講義

- 2013年10月7日 約 50 名の学生に 90 分の講義
- 感想 (のごく一部)
  - UMLモデルはプログラミングに使用するという先入観をもっていたのですが、今回の講義で学ぶ中でもっと広義的に使えるということがわかり、様々な面で活かせるようになりたいと思いました。
  - モデ脳を使いこなすことが出来たら情報を伝える、得るときに便利だと思いました。モデ脳に答えが無いところがいいと思います。具体例がとてもわかりやすかったです。

目次

- 自己紹介
- モデリングを詳しく知る前に
  - 現代人が主に使う能力
  - モデリングで鍛えられる能力
- モデルに慣れる（モデルの穴埋め）
- モデリング能力向上のための方法
  - 0 からモデリング
  - 既存モデルの添削

モデ脳 = 現象を簡単にモデリングできる脳力

雨だれ石を穿つ

説明文

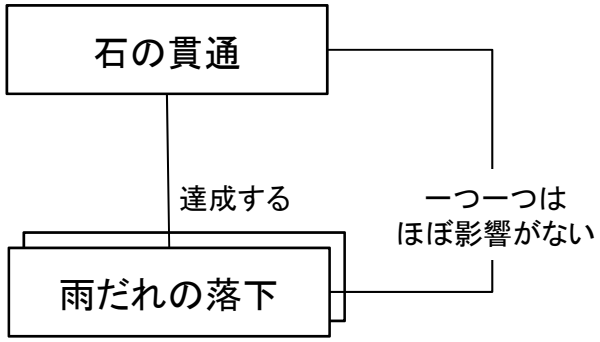
ある雨の日、雨だれが石を打っていました。
当然ですが、雨だれ一滴が石を打っても石の形は変わることなく、ただリズム良くピチョンピチョンと小気味よい音がしているだけでした。
雨が降るたびに、その石は雨だれに打たれ、心地よい音を奏で続けました。
長い年月を経て石を見てみると、なんと雨の日に雨だれが打っていた箇所に穴が空いているではありませんか。
長い年月を経て、雨だれが石を穿ったのです。

(穿(うが)つ：穴をあける。掘る。突き通す。貫く。)

雨だれ石を穿つ

- 具体モデル

  この話の内容をモデルに当てはめるとどうなりますか？
  空いているところに、適切な語句を入れてください。

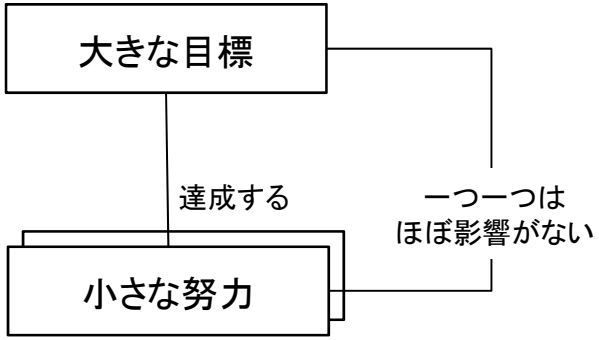


(a) 雨だれの落下 (b) リズム (c) 音 (d) 長い年月 (e) 短い年月

雨だれ石を穿つ

- 抽象モデル（抽象化）

  この話を抽象化するとどうなりますか？
  空いているところに、適切な語句を入れてください。

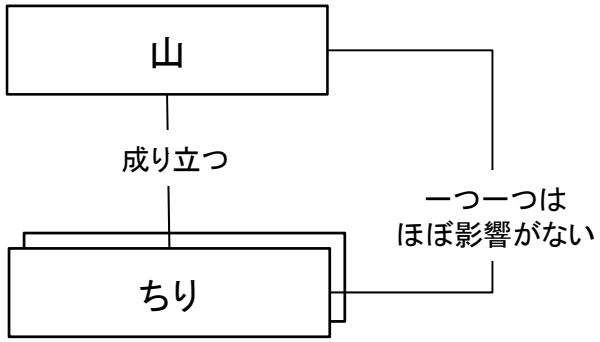


(a)長期間 (b) 短期間 (c) 大きな努力 (d) 小さな努力
(e)達成する (f) 破壊する

雨だれ石を穿つ

- 異なる具体モデル（具体化）

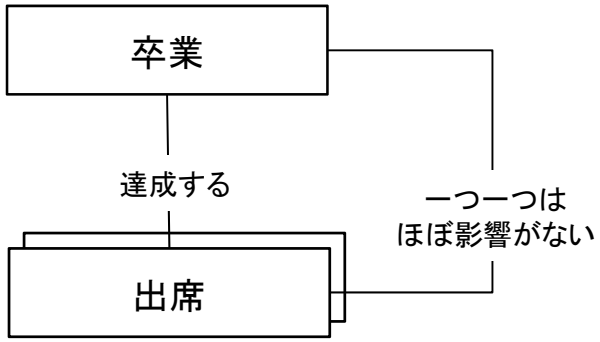
「ちりも積もれば山となる」も同様の構造を持っています。
具体モデルを作ってください。



雨だれ石を穿つ

- 異なる具体モデル（具体化）

「ちりも積もれば山となる」も同様の構造を持っています。
具体モデルを作ってください。



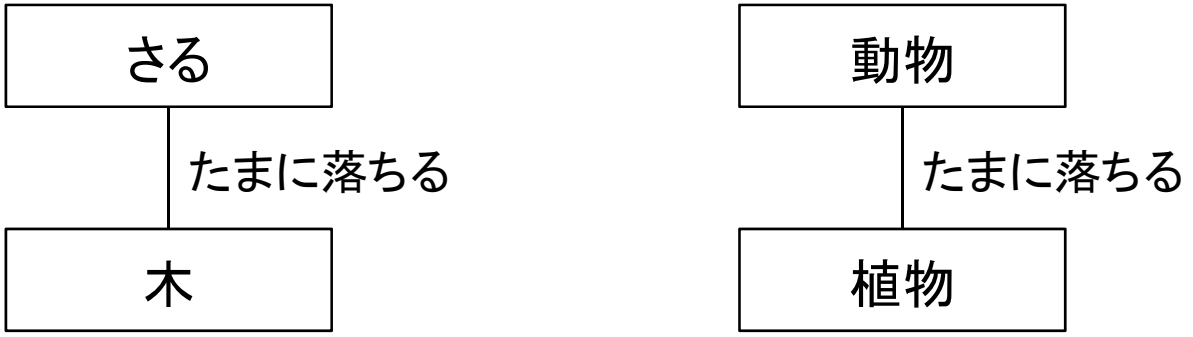
目次

- 自己紹介
- モデリングを詳しく知る前に
  - 現代人が主に使う能力
  - モデリングで鍛えられる能力
- モデルに慣れる（モデルの穴埋め）
- モデリング能力向上のための方法
  - 0 からモデリング
  - 既存モデルの添削

モデ脳 = 現象を簡単にモデリングできる脳力

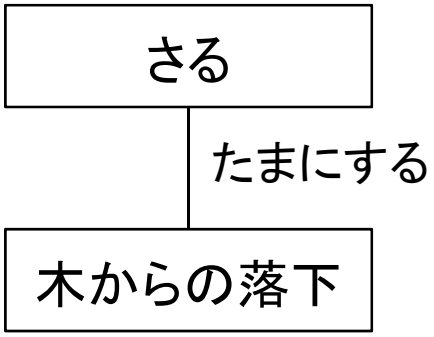
猿も木から落ちる

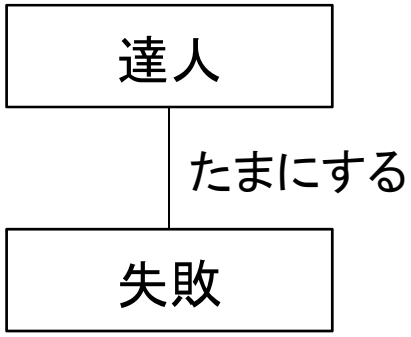
- 単純な具体モデルと抽象モデル



猿も木から落ちる

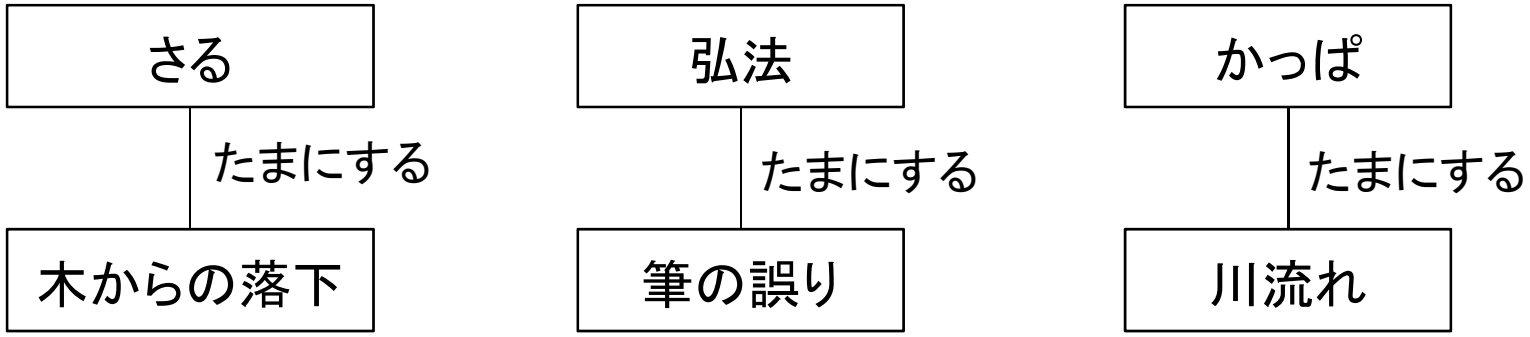
- 目的意識をもった具体モデルと抽象モデル





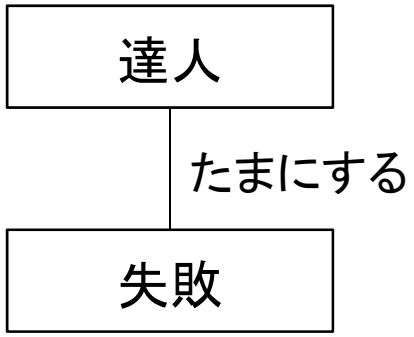
猿も木から落ちる

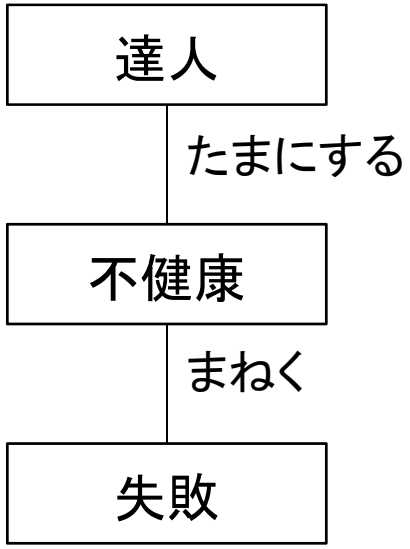
- 具体モデルの応用

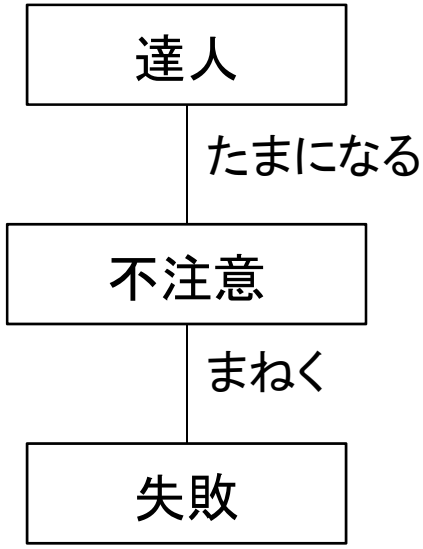


猿も木から落ちる

- 抽象モデルを改良していく
  - モデルに Why? を入れ込む

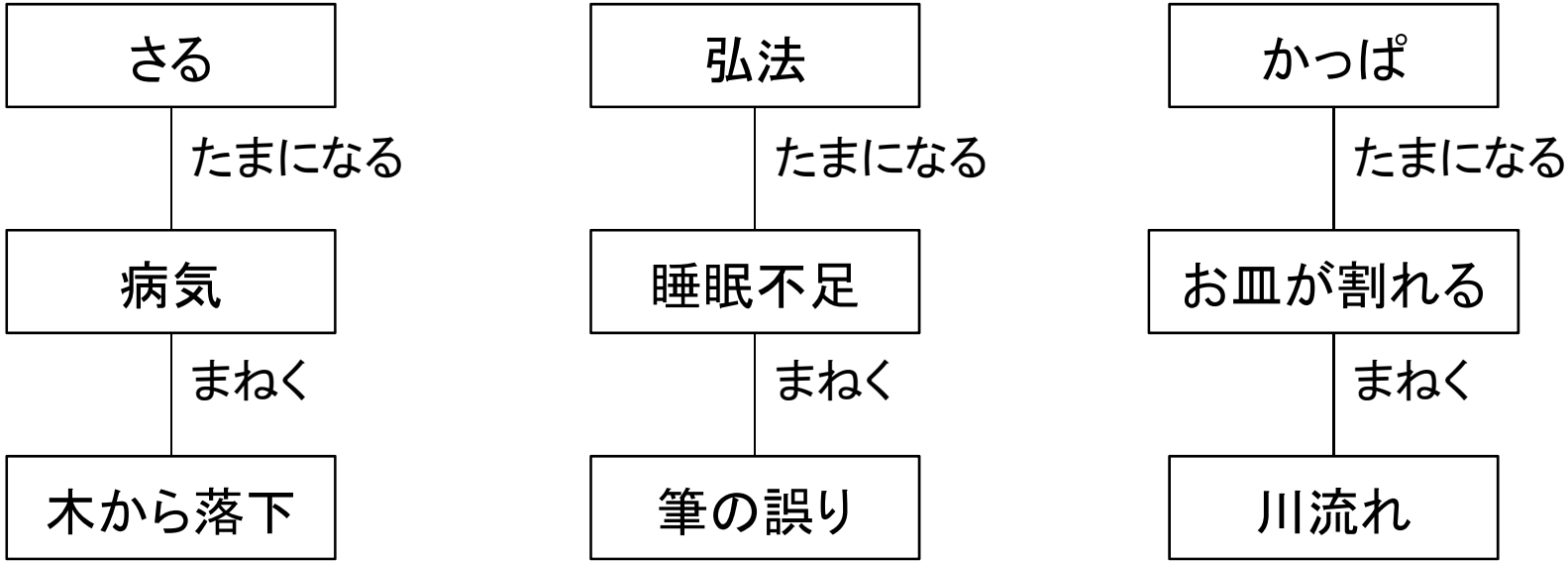






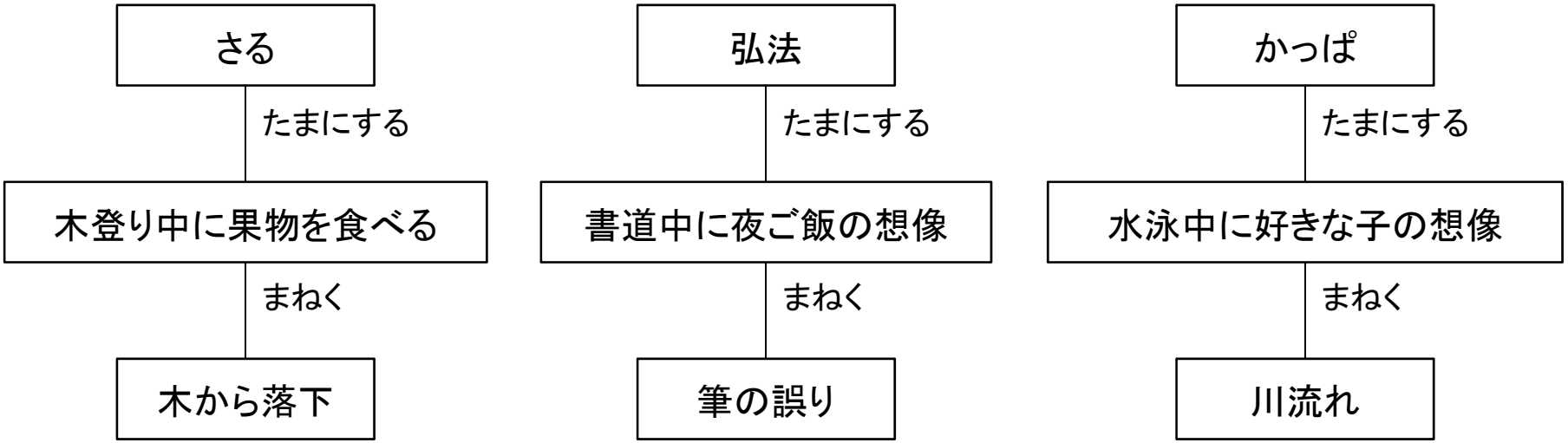
猿も木から落ちる

- 改良した抽象モデルを具体化する（不健康）



猿も木から落ちる

- 改良した抽象モデルを具体化する（不注意）



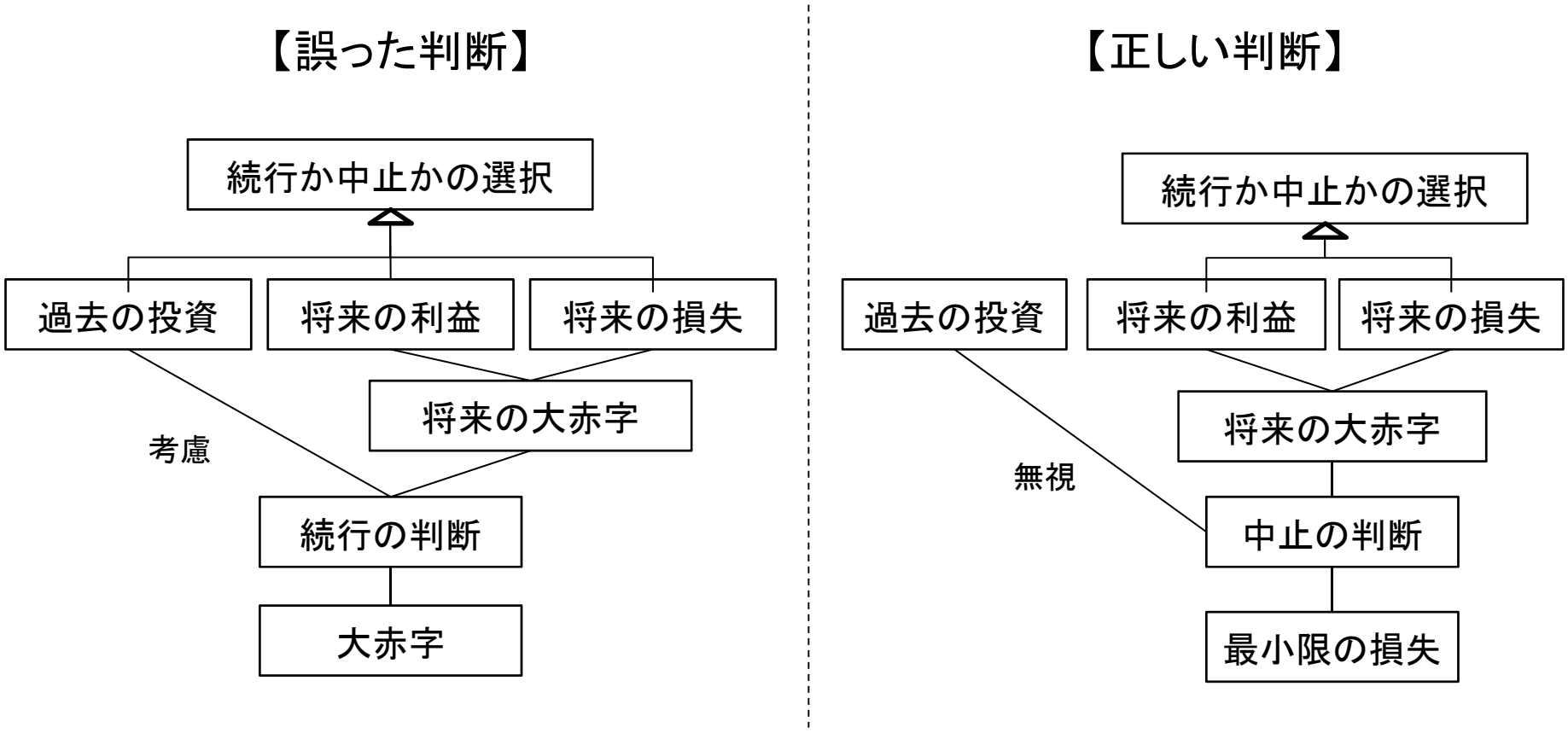
コンコルドの誤り

説明文

超音速旅客機「コンコルド」に莫大な費用をかけて開発してきたイギリスとフランス。
しかし、開発の途中段階で「コンコルド」が完成しても開発費用の採算が取れない事が明らかになりました。つまり利用者から乗車賃を貰って運用しても、採算が取れない計画だったのです。本来ならば、直ちに開発を中止して損失を最小限にすべきですが、「今までの投資が無駄になる」との理由で開発は続行されました。そしてその結果大赤字になってしまいました。
このように、今までの努力が無駄になってしまうからと、非効率なプロジェクトを続行してしまうことを「コンコルドの誤り」と呼びます。過去の投資は、どう足掻いても取り返せず、将来の損益にも全く影響を与えない「埋没費用」ですから、正しい判断をする上では無視しなければいけません。しかし、人はこの埋没費用を考慮してしまい、誤った決断を下しがちなのです。

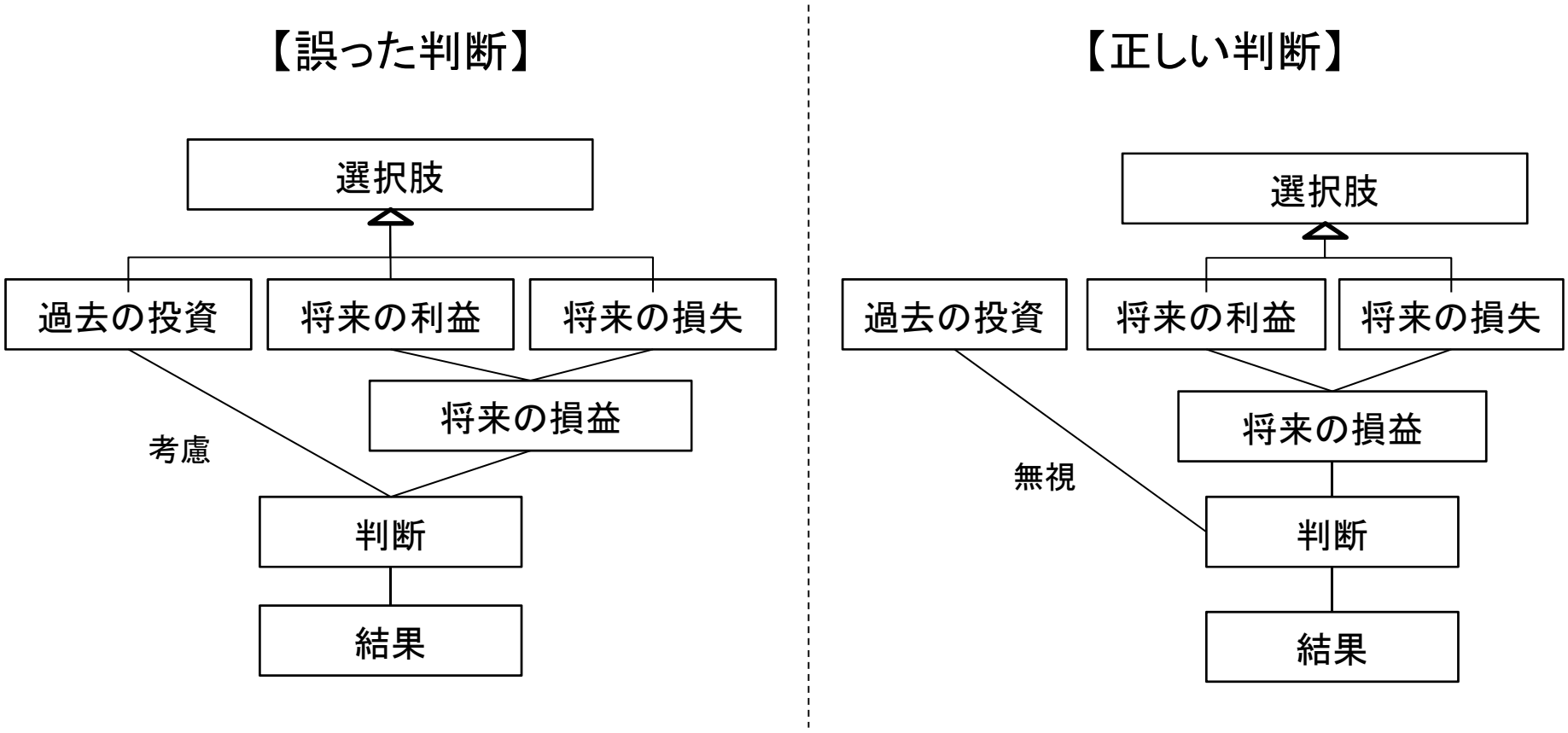
コンコルドの誤り

- 既存モデル（具体）



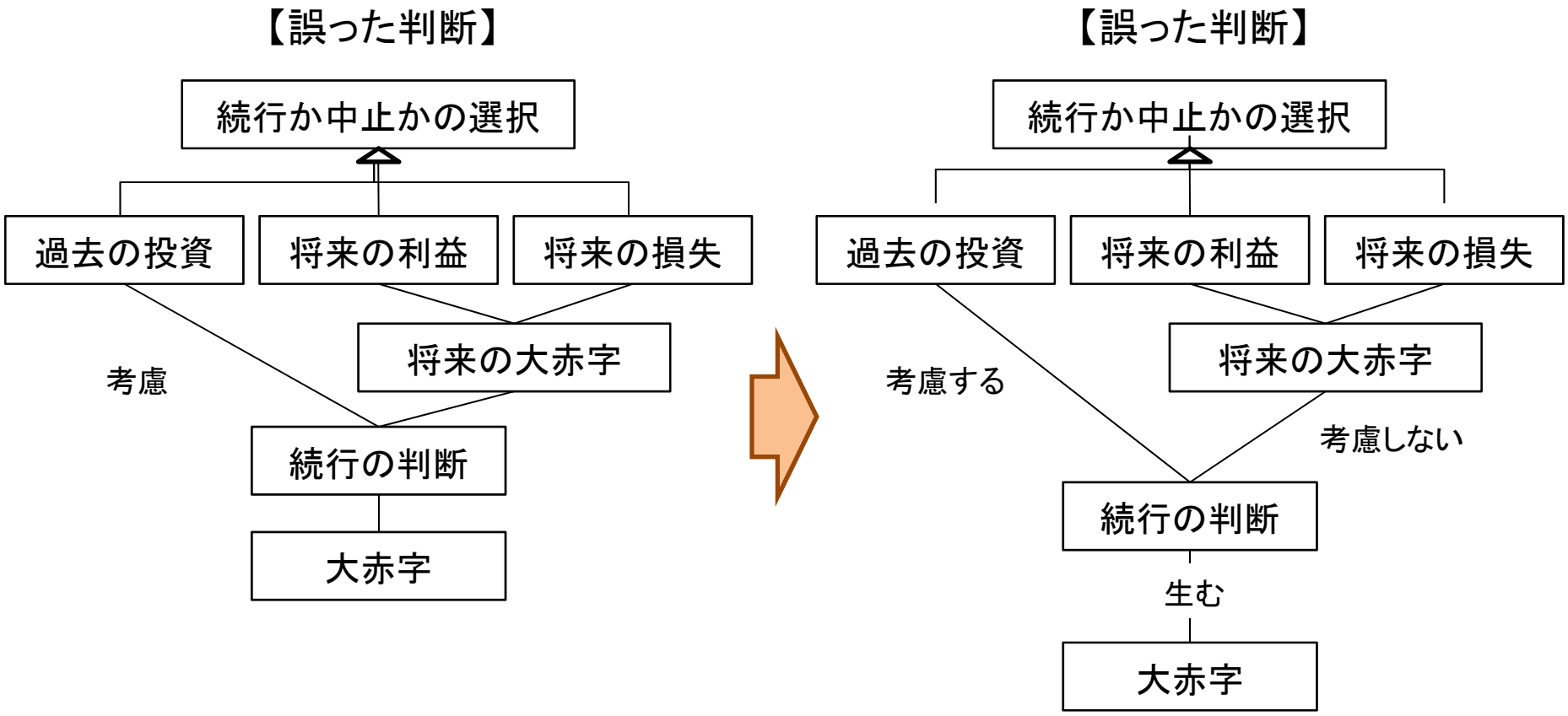
コンコルドの誤り

- 既存モデル（抽象）



コンコルドの誤り

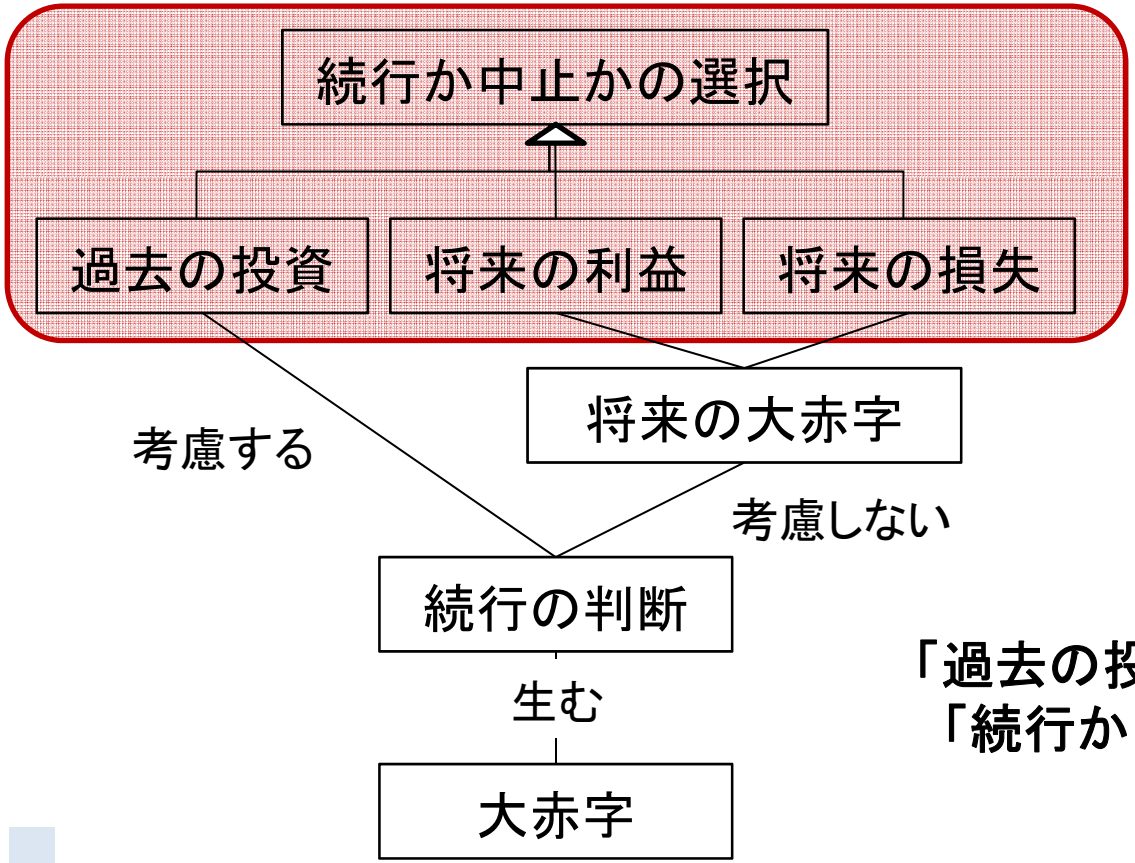
- 改善点①: 関連名を書く



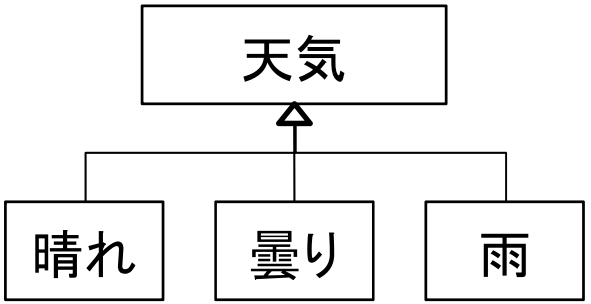
コンコルドの誤り

- 改善点②: 分類を正しく使う

【誤った判断】



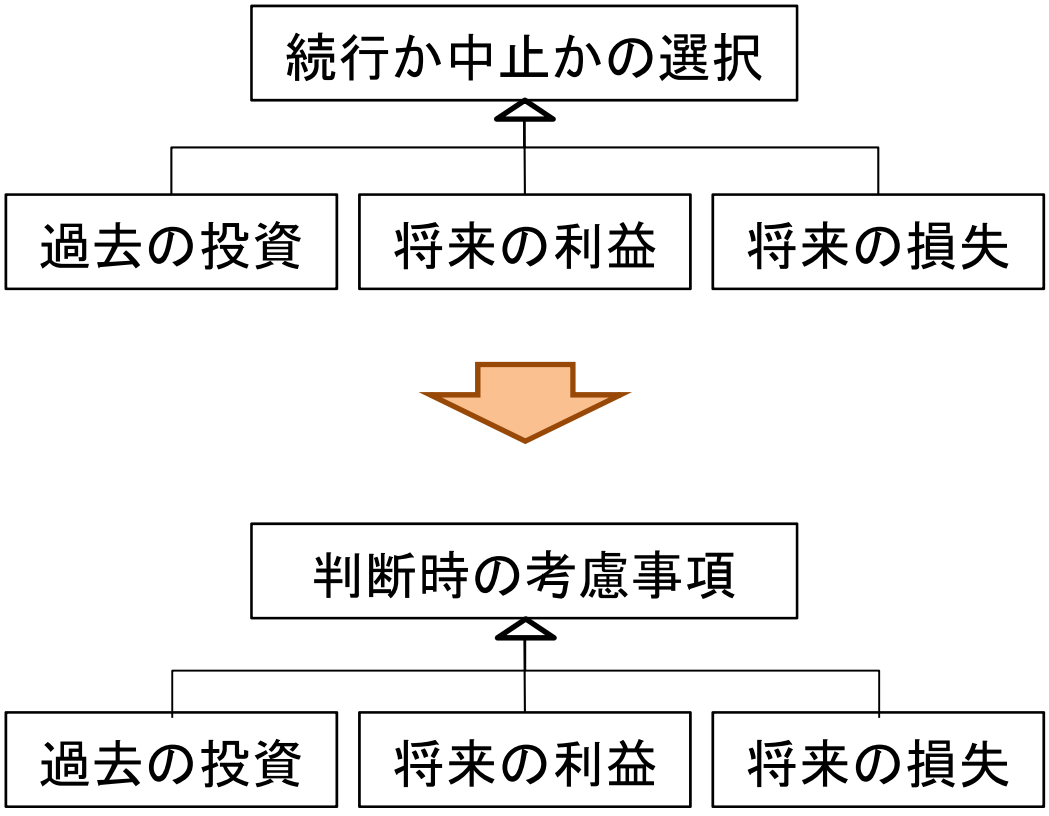
【分類の一例】



「過去の投資 」「将来の利益・損失」は
「続行か中止かの選択」ではない！

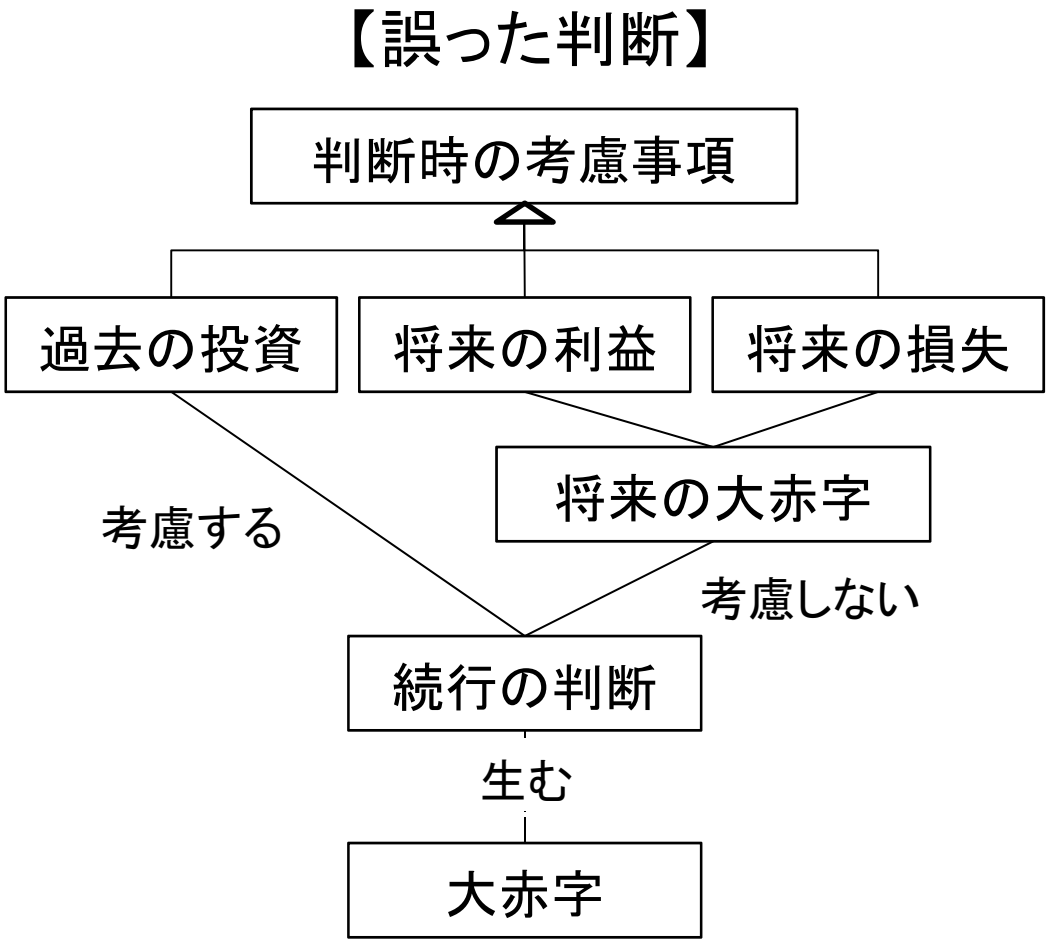
コンコルドの誤り

- 改善点②: 分類を正しく使う



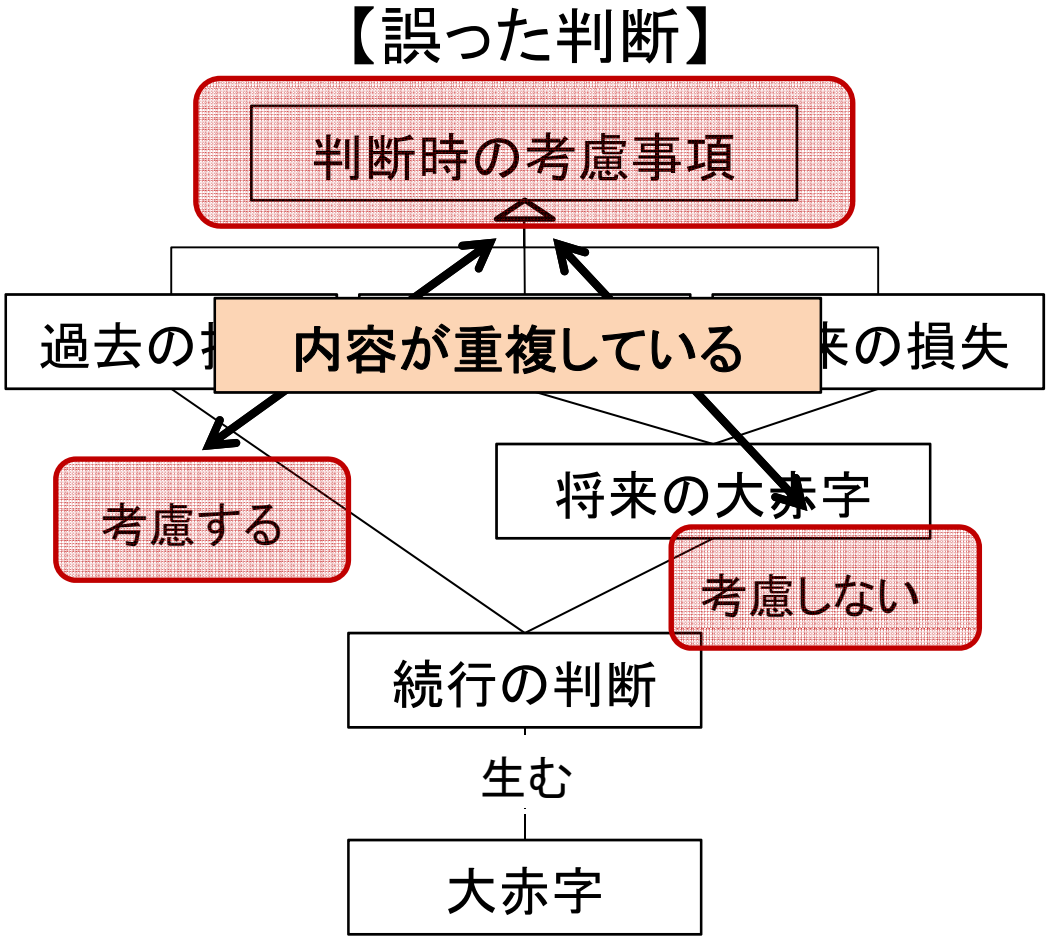
コンコルドの誤り

- 改善点②: 分類を正しく使う



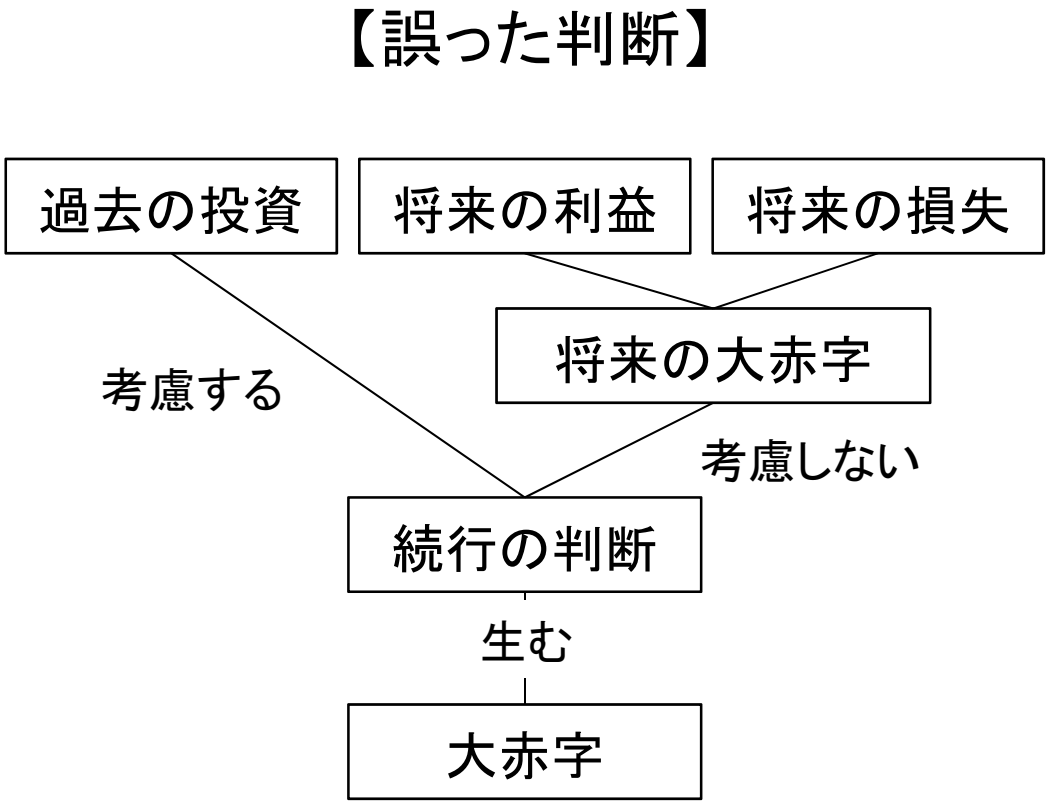
コンコルドの誤り

- 改善点③: 重複の除去



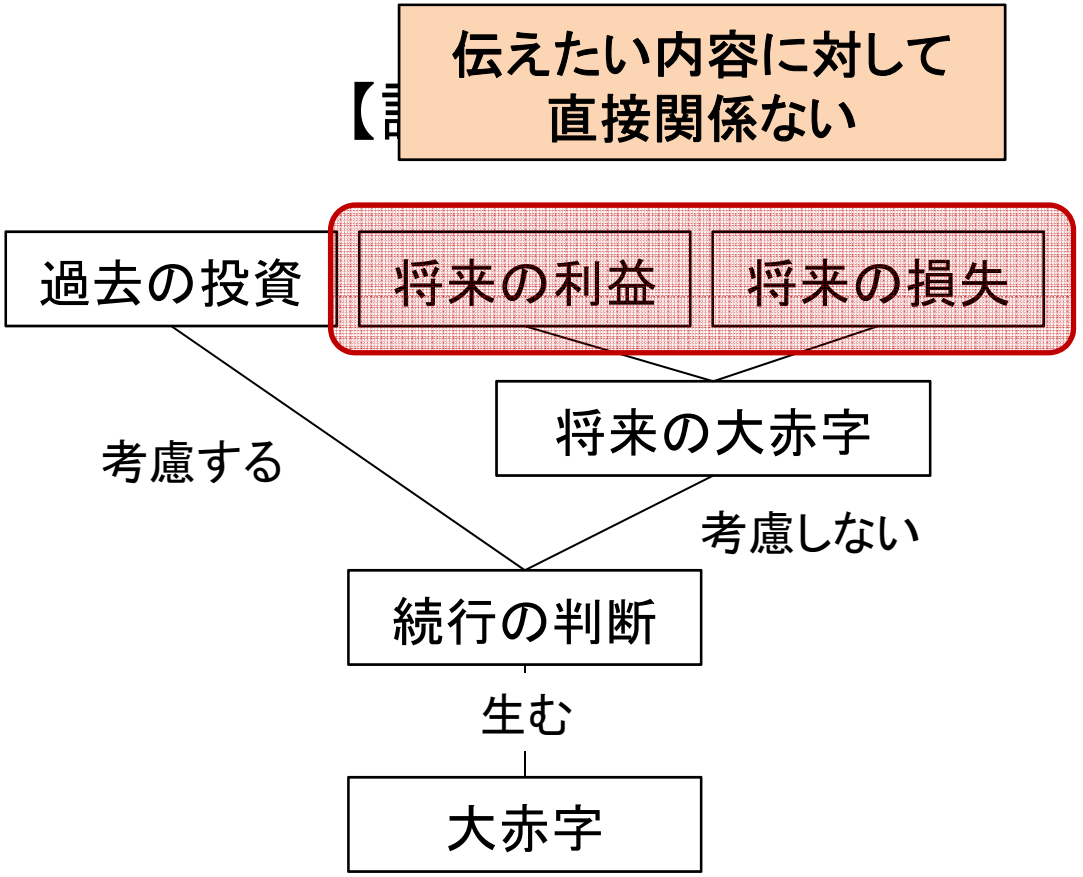
コンコルドの誤り

- 改善点③: 重複の除去



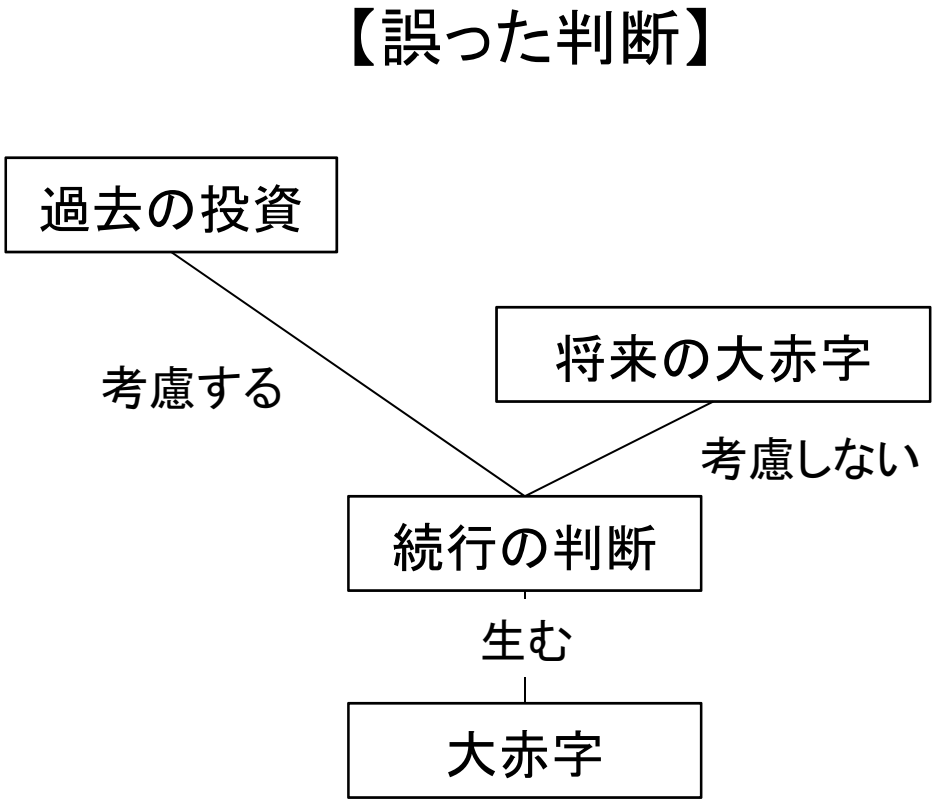
コンコルドの誤り

- 改善点④: 不必要な要素の除去



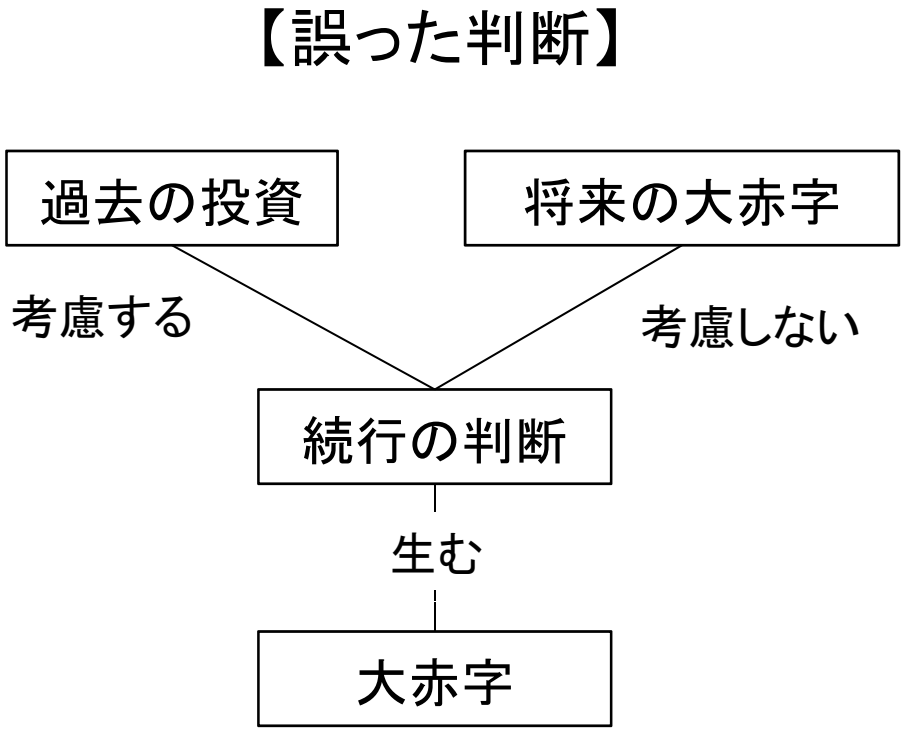
コンコルドの誤り

- 改善点④: 不必要な要素の除去



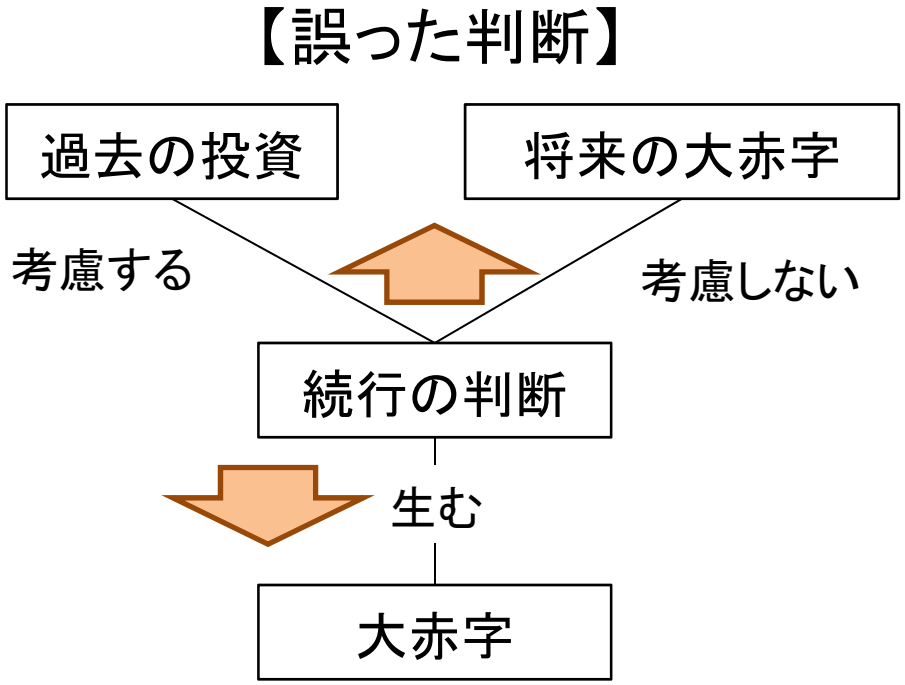
コンコルドの誤り

- 改善点④: 不必要な要素の除去



コンコルドの誤り

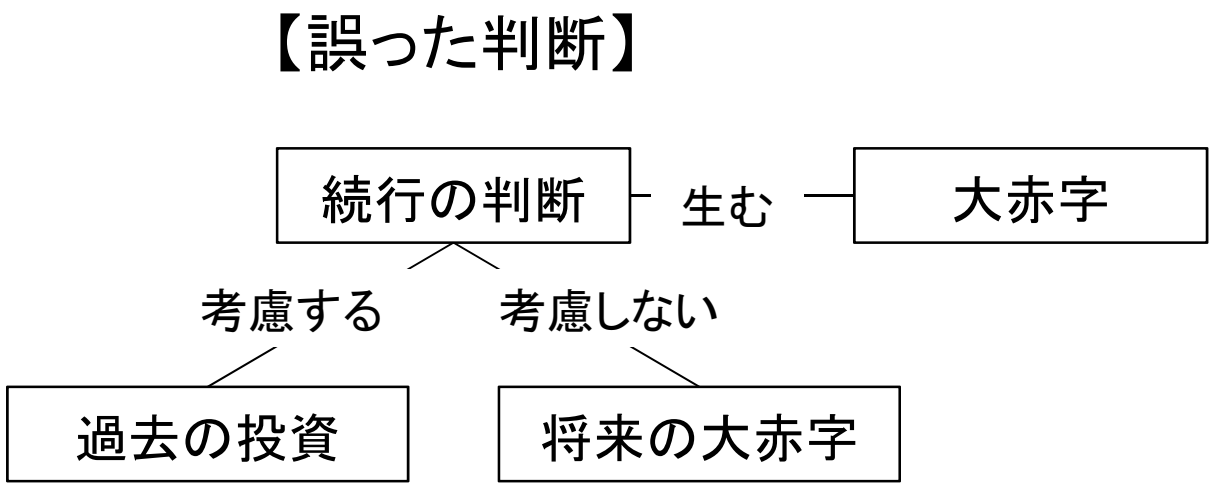
- 改善点⑤: 上から下、左から右



続行の判断を中心に下から上、上から下によませるモデルになっている
下から上がある + 左から右を活用していない

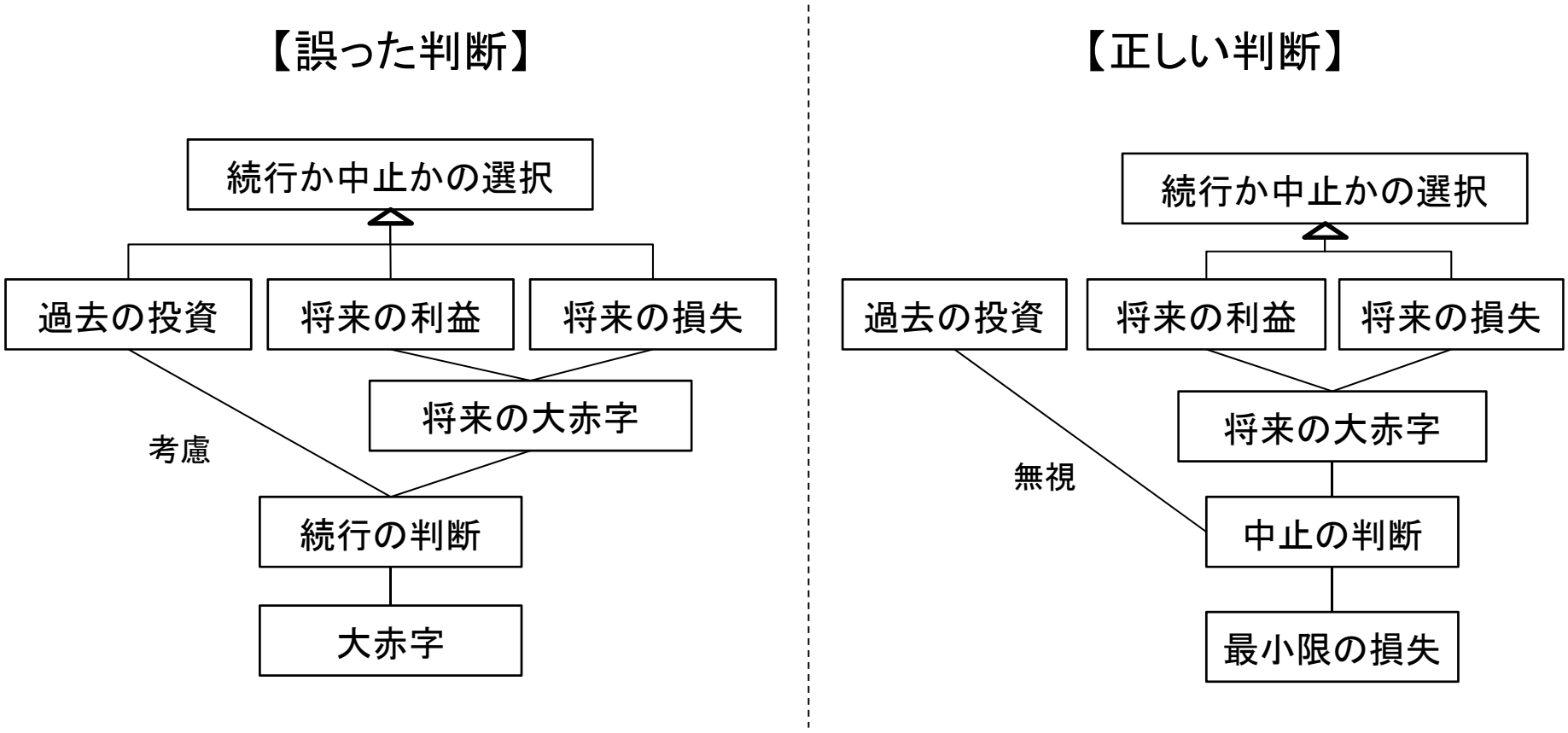
コンコルドの誤り

- 改善点⑤: 上から下、左から右



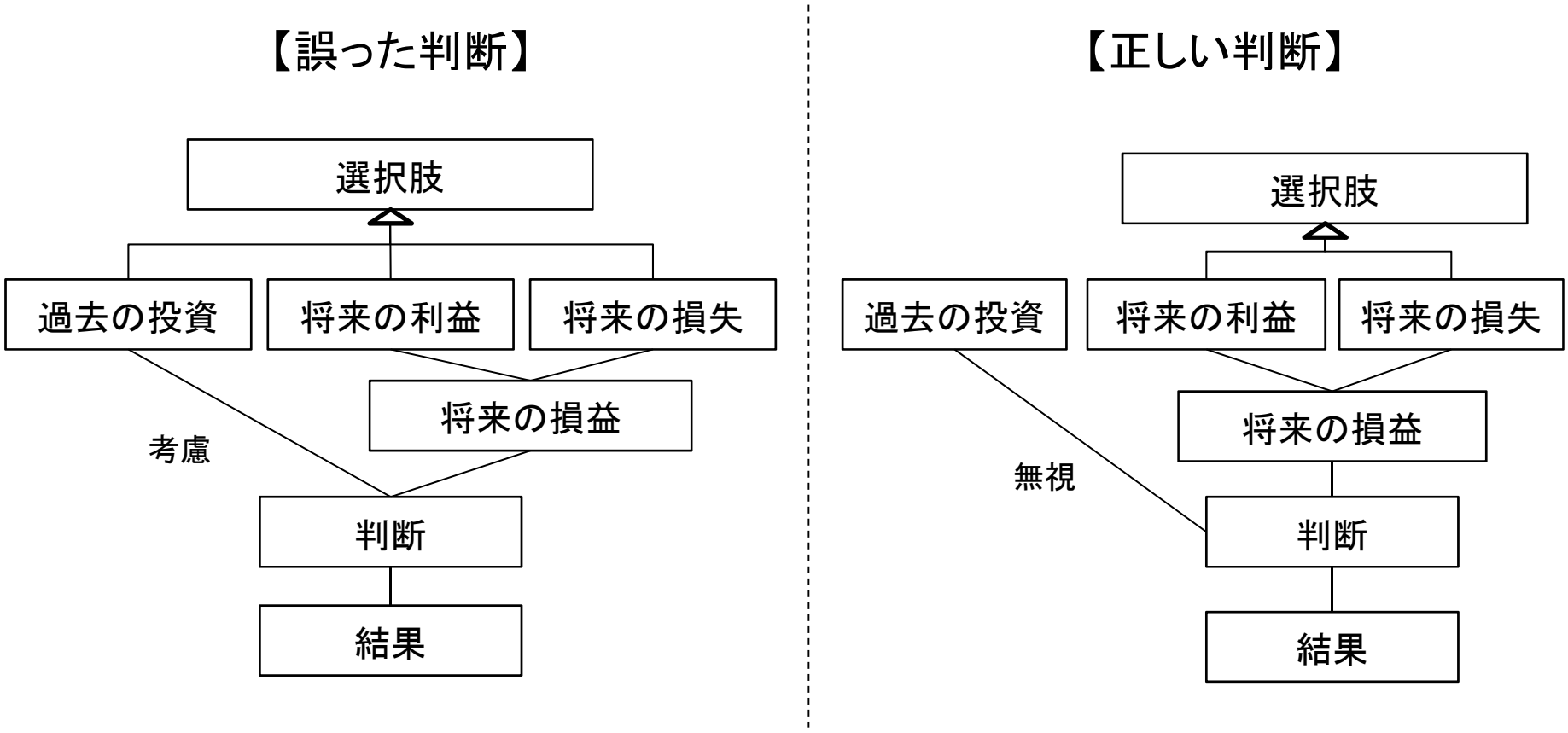
コンコルドの誤り

- 既存モデル（具体）



コンコルドの誤り

- 既存モデル（抽象）

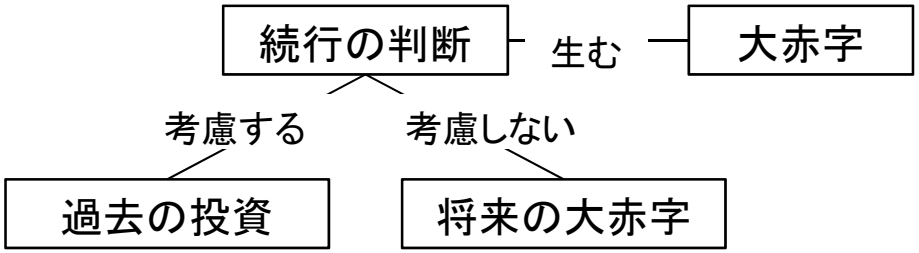


コンコルドの誤り

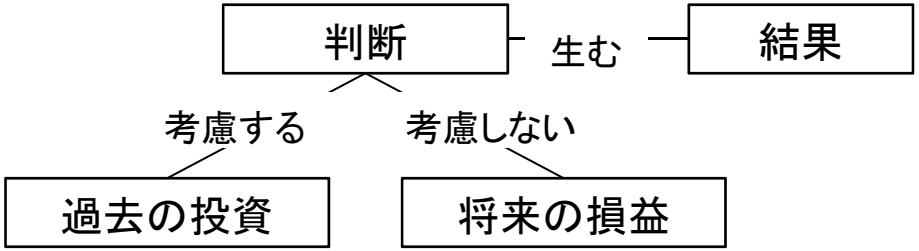
- 改善モデル（具体 + 抽象）

【誤った判断】

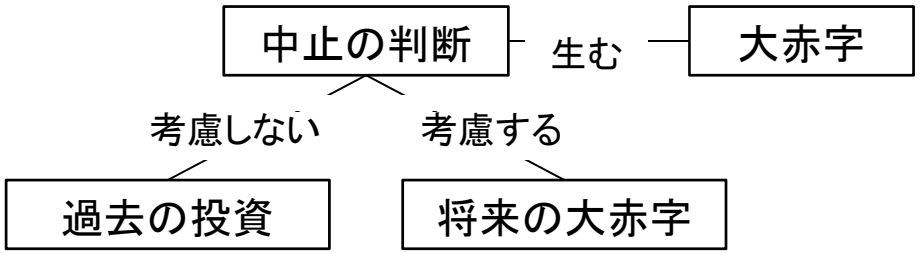
具体

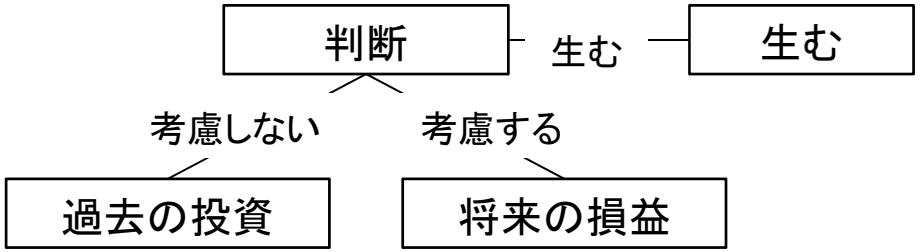


抽象



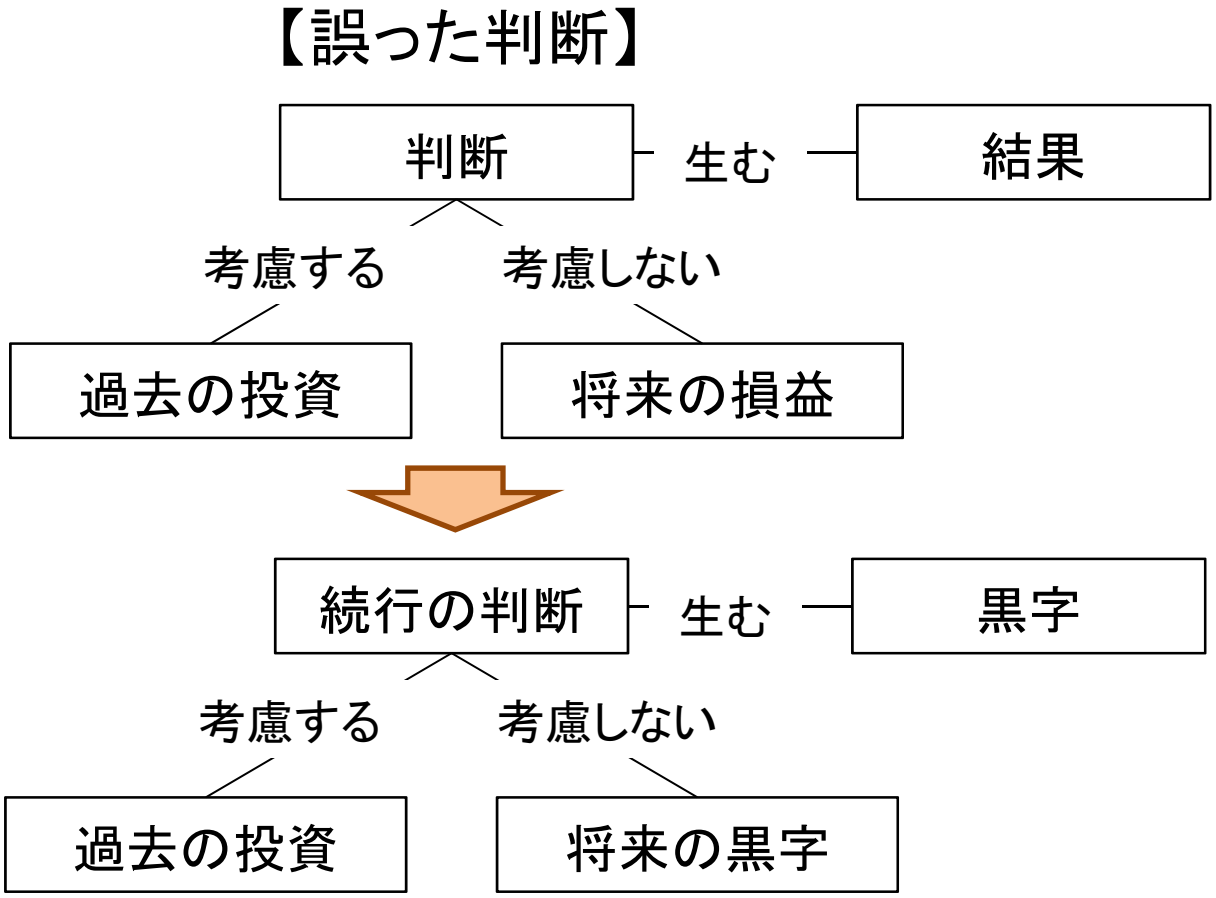
【正しい判断】





コンコルドの誤り

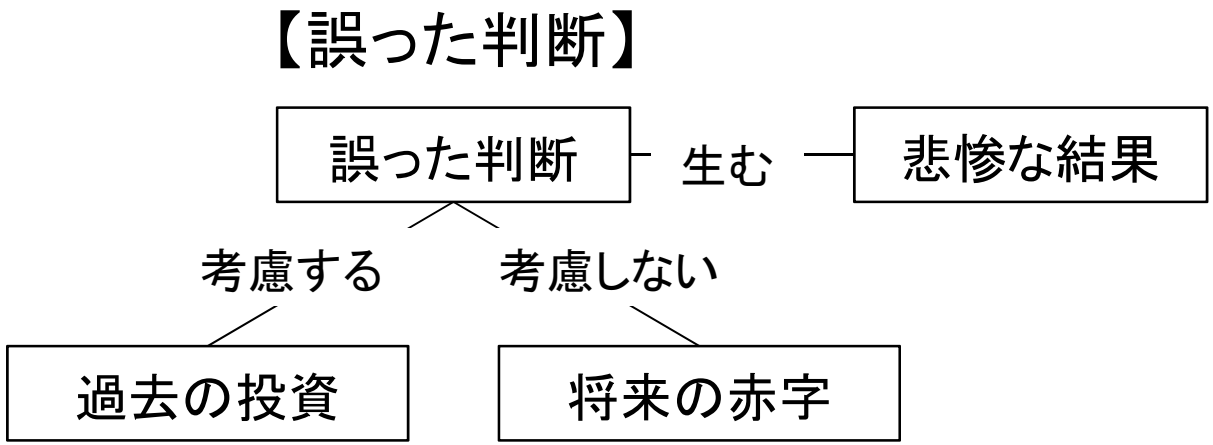
- 改善点⑥: 過度な抽象化をやめる



これも誤った判断をした結果の具体モデルになってしまう

コンコルドの誤り

- 改善点⑥: 過度な抽象化をやめる

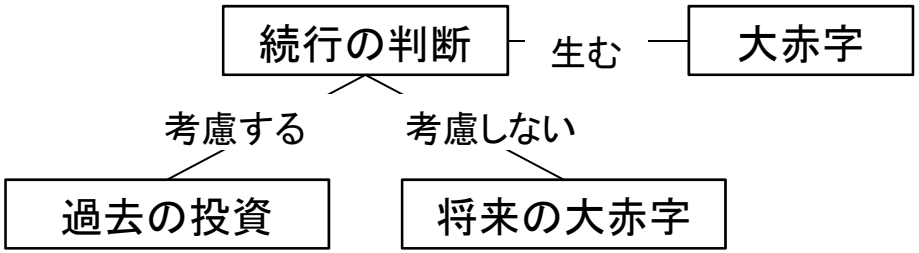


コンコルドの誤り

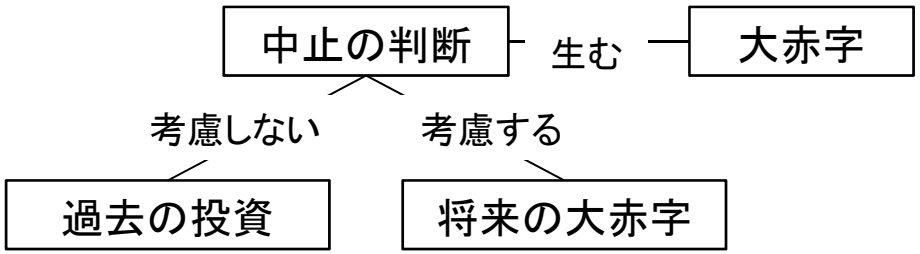
- 改善モデル（具体 + 抽象）

具体

【誤った判断】

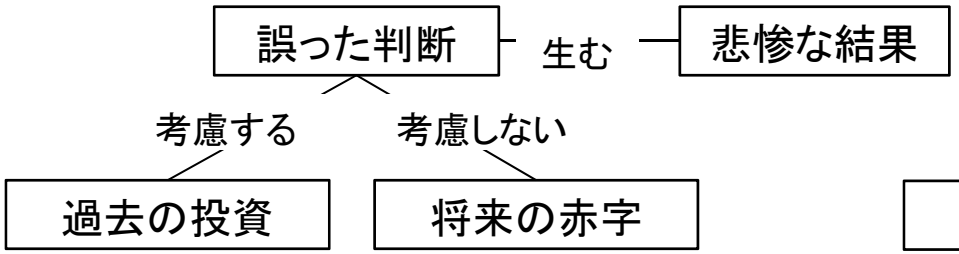


【正しい判断】

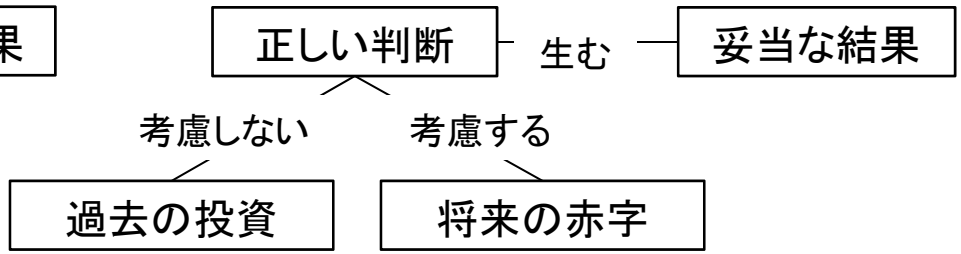


抽象

【コンコルドの誤りに陥っている判断】



【コンコルドの誤りを踏まえた判断】



まとめ

- モデリングは能力を上げる一つの手段として使える
  - Process
    - 情報をまとめる
    - 情報を抽象化し、具体化する
  - Output
    - 情報を表現する
- モデリング力を上げる 2つの方法
  - 0 からモデルを作り上げる
  - 既存のモデルを添削（改善）していく