形式手法のAからZまで
ーZをどう教えたらよいかー

SIGFM　第5回月例会

2000年7月7日

玉井哲雄

（東京大学大学院情報学環）

形式手法についての個人的な想い

- 世の中は，FMへの「アレルギー反応」と「偏愛」とに二分
- 自分はambivalent（愛憎相半ば)
  - 形式論理のきちんとした修練を経ていないという負い目
    - 野矢茂樹「論理学」
  - 思考の節約に価値
    - 形式と非形式間の自由な飛躍
    - 抽象のレベルの自由な昇降
  - 壷に嵌まった形式化の美しさ

形式化が呼び起こす好悪感

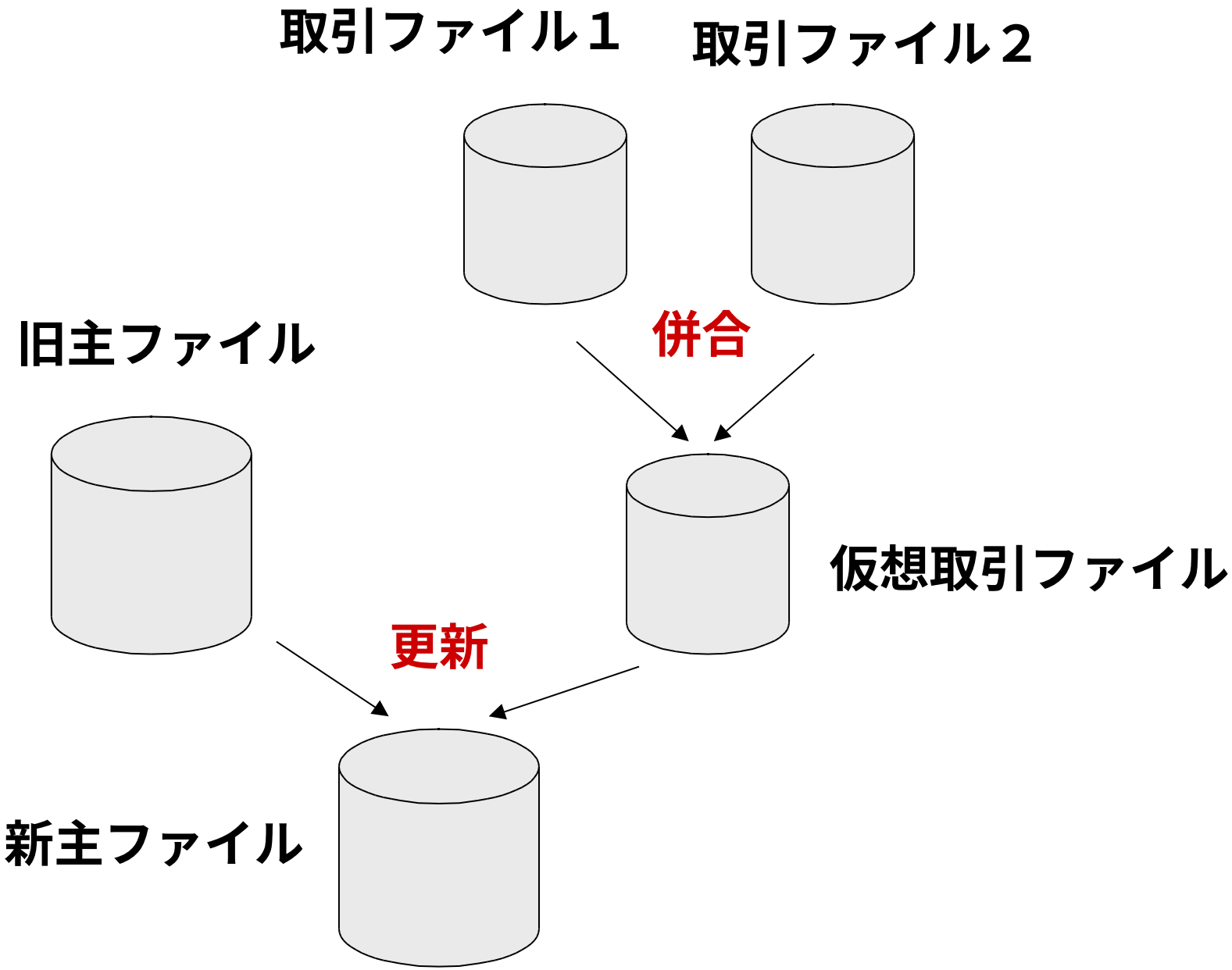
- （とくに日本語で）「形式的」　⇒　中身がない
- 「人の思考は形式的でない」
  - 形式的推論を追うことはできるが面倒
  - モデルに引きつけて理解
- 「抽象化＝形式化ではない」
  - 形式化は抽象化である．
    - 記号化で対象/意味を捨象
  - 非形式的な抽象化もある．

形式手法に関わる回想（１）
ファイル処理プログラムの検証系開発

- 玉井・福永，1977－1980
- 技術的には
  - 半順序に特化した推論機能を持つ定理証明系
  - ループ不変量の自動生成
- より有益だと思われるアイディア
  - 仮想ファイル
    - 同じインターフェースに対する複数の実現
    - 同じ計算対象に対する複数のインターフェース

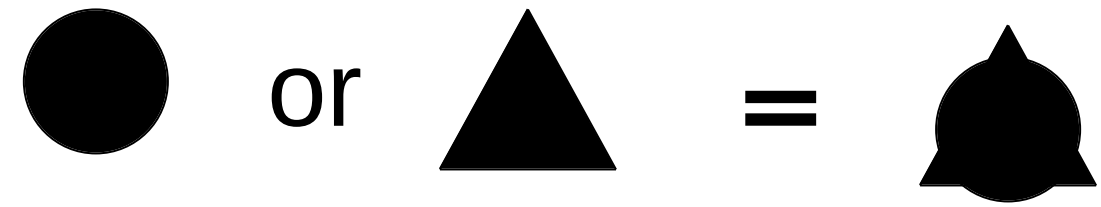
仮想ファイル

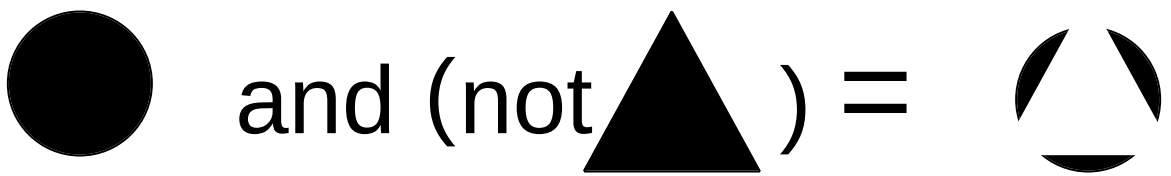
- 背景：n本ファイルの併合，更新，等の処理
  - 例：主ファイルを2本の取引ファイルで更新
    ⇒2本の取引ファイルを併合した仮想ファイルを導入
- ファイルへの操作インタフェース(open, read, eofなど）を定義
- 2本のファイルを実際に併合せず，それへの操作のみを公理を満たすように実現
- 関連する技術
  - ADT
  - ストリーム・プログラミング
  - Javaのimplement句


取引ファイル１
取引ファイル２
旧主ファイル
併合
仮想取引ファイル
更新
新主ファイル

局面志向プログラミング批判

- Aspect Oriented Programmingの例題：画像処理
  - 自然なモジュール化
    - 白黒画像のand, or, not, up, downなどを用意
    - それらの組み合わせでたとえば縁線抽出などの操作を定義
  - そのままでは速度も記憶容量もむだが多いので，性能を上げるための局面プログラムを書く．
- しかし，仮想ファイルと同じアイディアを使えば，and, or, not, up, down のような基本操作を，実際の画面を作らずに，その特定画素のget/put操作として実現し，モジュール性を保ちながら性能を上げることができる．





upedge(X) = and(X, not(down(X)))

複数のインターフェース(視点)

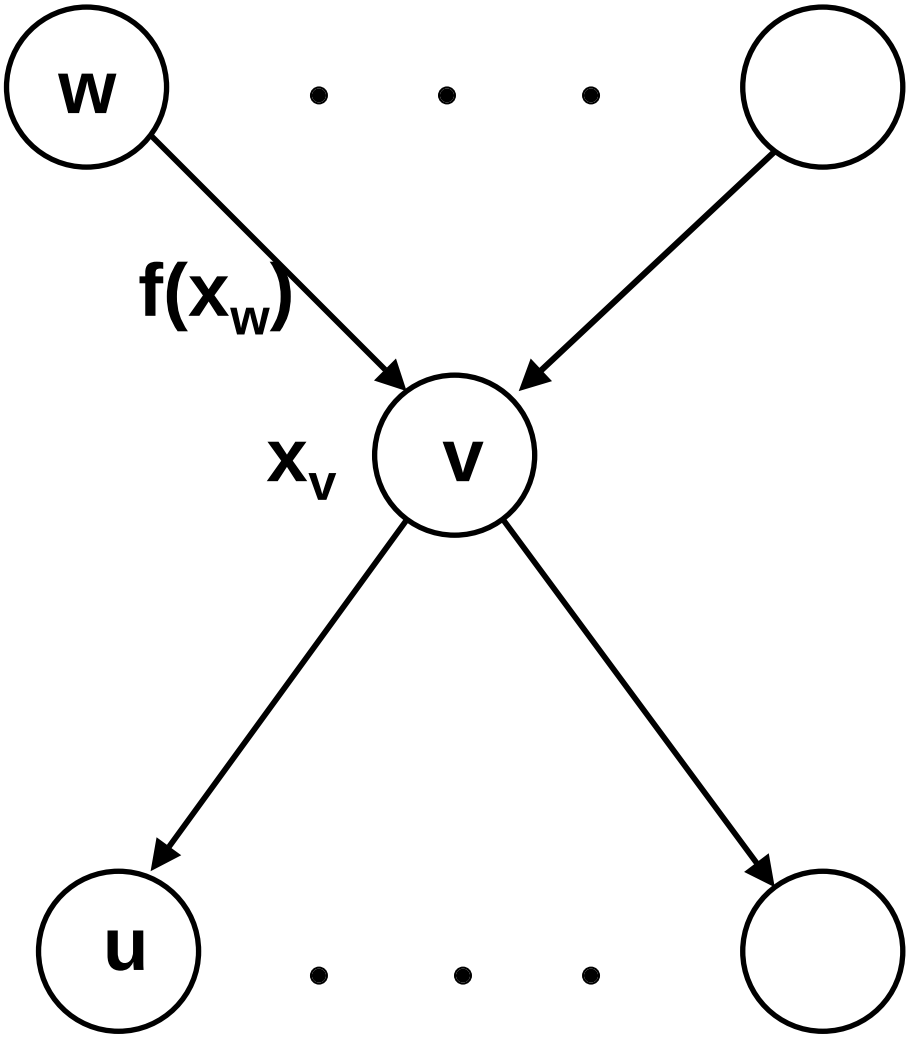
- ファイル＝レコードの並び
  ＝レコード・グループの並び
  ⇒　「キーの切れ目処理」
- テキスト＝文字の並び
  ＝単語の並び
  ＝行の並び
  ⇒　D. Jacksonのエディタの仕様
- ここでも仮想ファイルの概念が使える．

形式手法に関わる回想（２）
多様な問題の統一的な扱い

- 共通構造をもつ問題への遭遇
  - 大規模線形計画問題の構造化
    (行列の構造変換)
  - 整数計画法(分枝限定法)
  - ネットワーク計画問題（最短路，最大流，最小費用流，多種流)
  - プログラムの静的解析(データフロー解析)
  - プログラムの記号実行
  - AIの探索問題(A*アルゴリズム)

グラフ上の不動点問題として統合化

- グラフと束と束の上の関数空間を用い，不動点方程式として定式化
- アルゴリズムの統一的記述と，正当性の証明
- 山本・萩谷等がHOLで記述，検証
- CafeOBJでも記述．パラメータ・モジュールで個別問題領域に写像．


w
f(x_w)
x_v
v
u

モデル検査

- 最近はやり (FM99, ICSE2000)
- もとは1980年代初め．Clarke & Emerson’82など
- モデルが時相論理式を満たすかどうかを自動的に検査
- アルゴリズムはグラフ上の不動点計算
- ハードウェア(論理回路)の検証では日常的に使われている．
- セキュリティ・プロトコルの検査で注目された．

形式手法の意味

- 曖昧性のない厳密な仕様
- 正しさや性質の検証
- “実行”（シミュレーション）
- プログラムの自動合成

本当に実用性があるか？

- 形式手法の利点は広く認識されつつある．
- 実際的な適用例も増えてはきたが，まだ限定的
  - いつも同じ事例 （たとえばCICS）
  - 効果が明らかなのは
    - ハードウェア（論理回路）
    - 通信プロトコル
  - 上の分野ではモデル検査が活躍

ソフトウェアでは

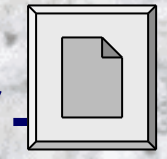
- 従来は高信頼性が要求されるもの
  - 安全性，機密性
- これからは再利用性の高いもの／広い範囲で繰り返し使われるもの
  - ライブラリ，コンポーネント
  - スタンダード
  - 一般性の高い問題と解法

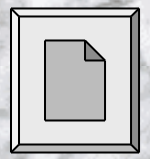
形式的抽象化の方法

- 機能の抽象化
- データの抽象化
- オブジェクトの抽象化
- 時間（順序，並列性）の抽象化

Zの教科書的な導入の仕方

- 状態機械の仕様記述
  - Spiveyの「お誕生日ノート」
- よく使われる表現





Zのくせ

- すべては基本集合の上で作られる．
  - M. Jacksonの批判：　時間的に不変，互いに素という仮定
- 部分関数の多用
  ←　関数や関係の集合的な定義
- スキーマの引用　⇒　簡潔で暗黙な表現　(その意味で分かりにくい)

山崎さんの酒屋問題

ある酒類販売会社の倉庫では，毎日数個のコンテナが搬入されてくる．その内容はビン詰めの酒で，1つのコンテナには10銘柄まで混載できる．扱い銘柄は約200種類ある．倉庫係は，コンテナを受け取りそのまま倉庫に保管し，積荷票を受付係へ手渡す．また受付係からの出庫指示によって内蔵品を出庫することになっている．内蔵品は別のコンテナに詰め替えたり，別の場所に保管することはない．
空になったコンテナはすぐに搬出される．

積荷票: コンテナ番号(5桁)
　　搬入年月，日時
　　内蔵品名，数量(の繰り返し)

酒屋問題(2)

さて受付係は毎日数十件の出庫依頼を受け，その都度倉庫係へ出庫指示書を出すことになっている．出庫依頼は出庫依頼票または電話によるものとし，1件の依頼では，1銘柄のみに限られている．在庫がないか数量が不足の場合には，その旨依頼者に電話連絡し，同時に在庫不足リストに記入する．また空になるコンテナを倉庫係に知らせることになっている．倉庫内のコンテナ数はできる限り最小にしたいと考えているからである．

出庫依頼: 品名，数量
　　送り先名

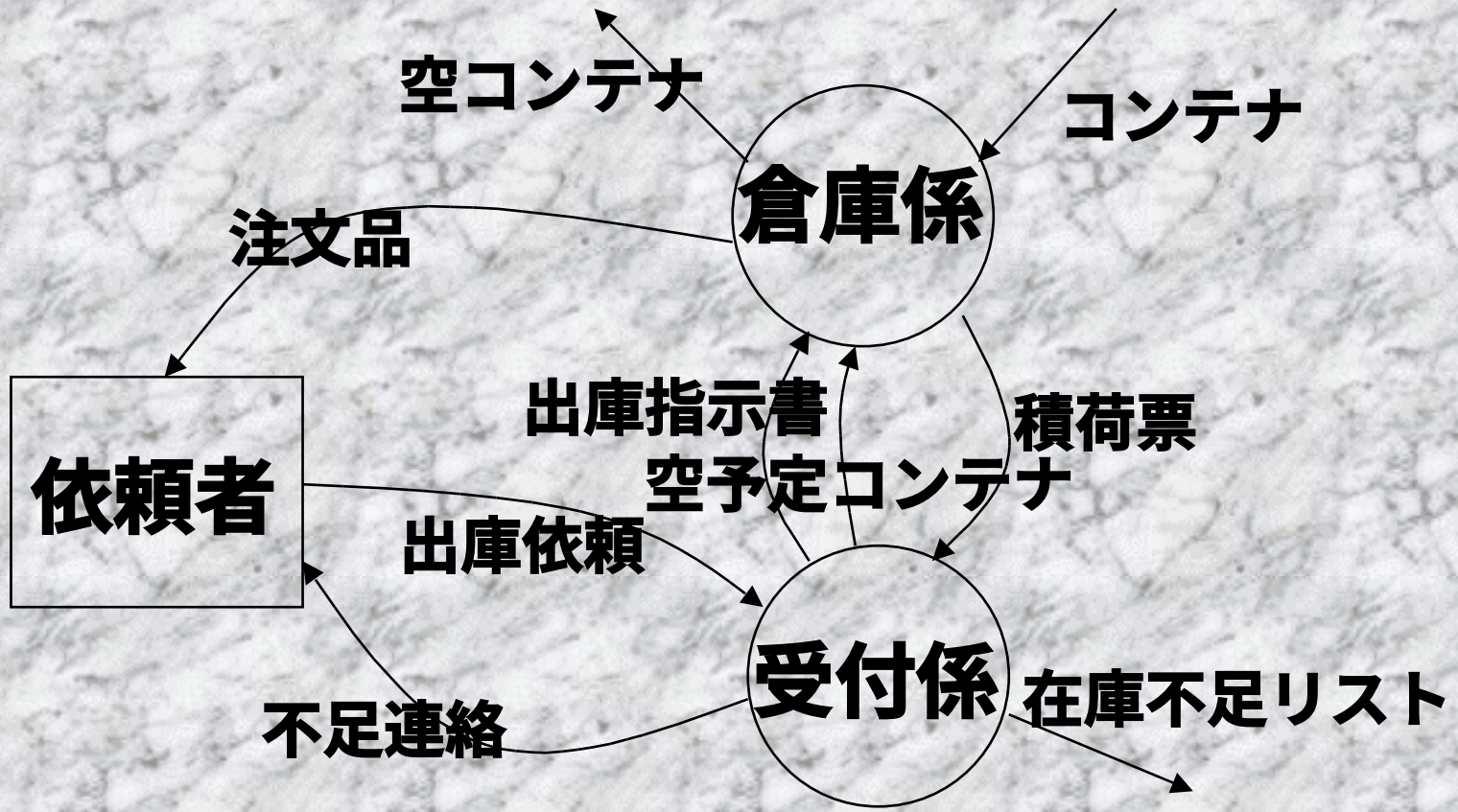
酒屋問題(3)

受付係の仕事(在庫なし連絡，出庫指示書作成および在庫不足リスト作成)のための計算機プログラムを作成せよ．

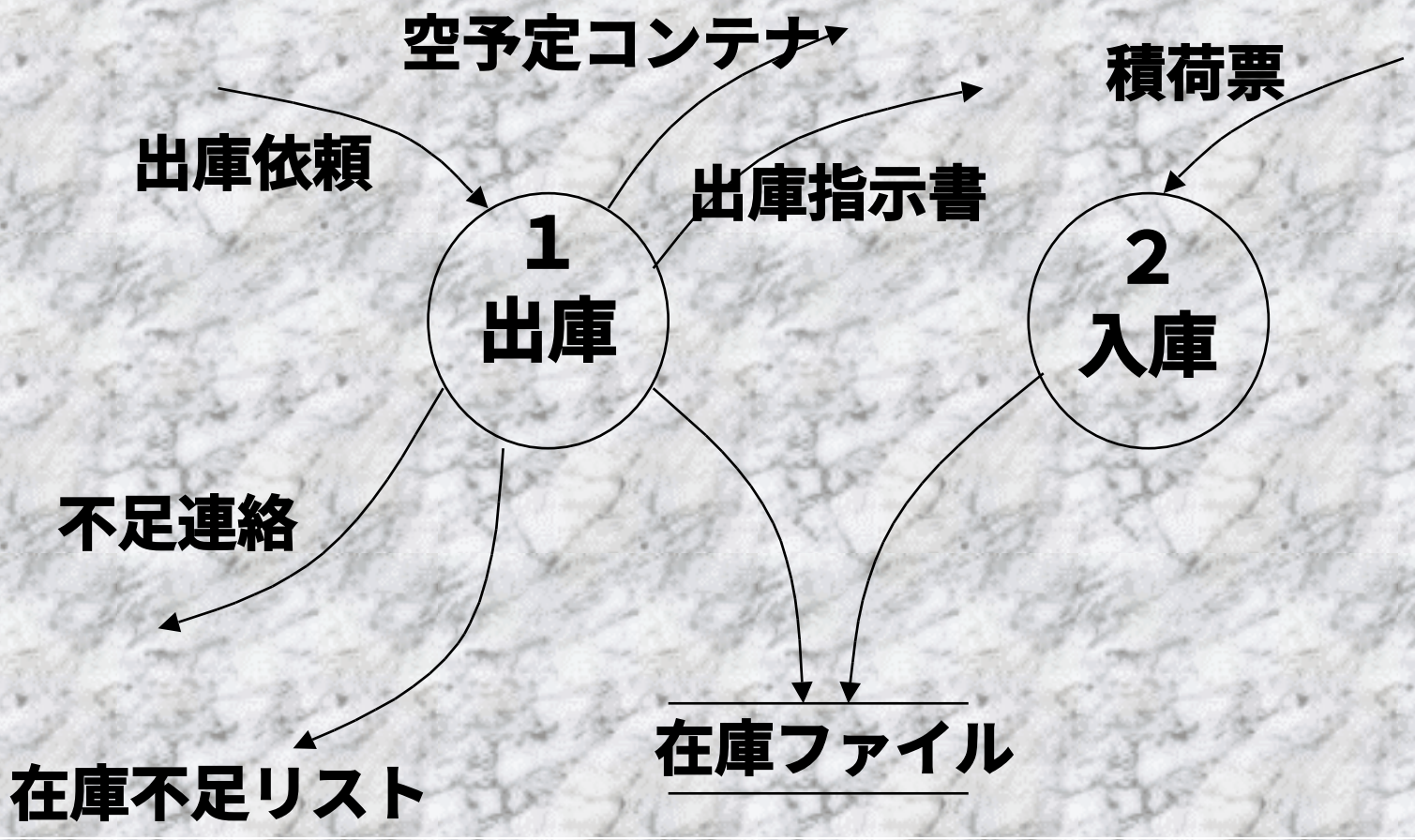
出庫指示書： 注文番号
　　送り先名
　　コンテナ番号
　　品名，数量　　(の繰り返し)
　　空コンテナ搬出マーク
在庫不足リスト: 送り先名
　　品名，数量

- なお移送や倉庫保管中に酒類の損失は生じない．
- この課題は現実的でない部分もあるので，入力データのエラー処理などは簡略に扱ってよい．

酒屋問題の全体文脈図


空コンテナ
コンテナ
倉庫係
注文品
依頼者
出庫指示書
積荷票
空予定コンテナ
出庫依頼
受付係
在庫不足リスト
不足連絡

酒屋問題 トップレベル図



Zによる酒屋問題の記述

- 高階関数の利用
- 非決定性の利用
- 宣言性の利用
  (入力を出力の関数として表す)