ワークショップ２
クラシフィケーションツリー技法

2024/5/31

JaSST Tohoku 実行委員会

JaSST Tohoku

タイムテーブル

開始時刻	終了時刻	内容
15:20	15:35	CT技法の説明
15:35	16:15	作り方の説明
16:15	16:40	練習問題1
16:40	17:05	練習問題2
17:05	17:10	まとめ

CT技法の説明：目次

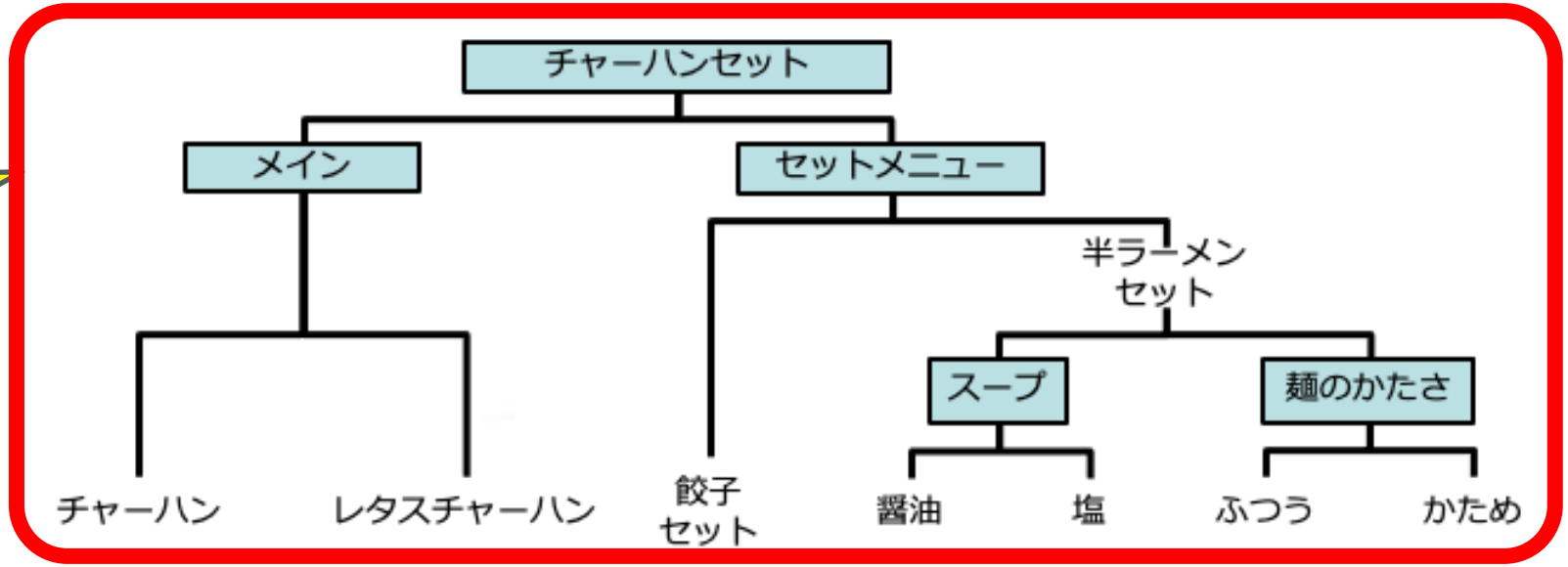
1. クラシフィケーションツリー技法(以下、CT技法)とは
2. 組み合わせテストとは
3. CT技法のメリット
4. クラシフィケーションとクラス
5. CT技法によるテストケースの作り方
6. 組み合わせテストの網羅基準
7. CT技法でのテストケース作成例
8. CT技法が向いているケース、向かないケース

クラシフィケーションツリー技法とは（１）

- **クラシフィケーションツリー技法**(Classification Tree Technique):
  ブラックボックステスト技法のひとつ。クラシフィケーションツリーを使用してテストケースを設計する。

- **クラシフィケーションツリー**(Classification Tree):
  テスト対象のテストデータ領域を表すツリーダイアグラム。

クラシフィケーション ツリー



（出典：ISTQB 用語集 https://glossary.istqb.org/ ）

クラシフィケーションツリー技法とは（２）

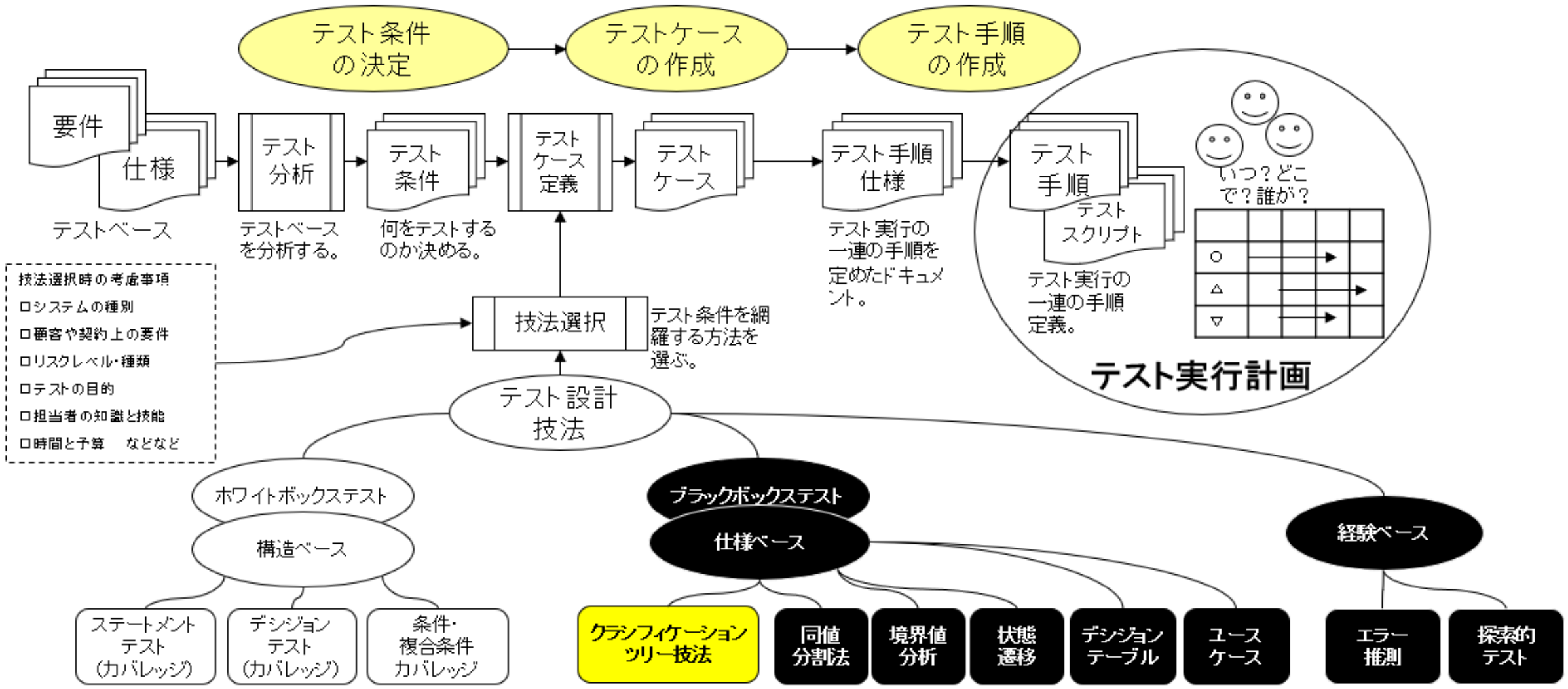
- **ブラックボックステスト設計技法のひとつ**

ブラックボックステスト設計技法：

> コンポーネントやシステムの内部構造を参照することなく、機能仕様や非機能仕様の分析に基づきテストケースを設計、選択する技法。（ISTQB用語集 https://glossary.istqb.org/ ）

クラシフィケーションツリー技法とは（３）

・ブラックボックステスト設計技法とクラシフィケーションツリー技法



※ASTERセミナー標準テキストを引用して修正
https://aster.or.jp/business/seminar_text.html

クラシフィケーションツリー技法とは（４）

- **主に組み合わせテストを作る技法**

組み合わせテスト：

複数の値を持つ複数のパラメータを含むソフトウェアに対して、
組合せに含めるアイテムの数を選択して、作成した組合せを
テストする。（ASTERセミナー標準テキスト　https://aster.or.jp/business/seminar_text.html）

組み合わせテストとは

複数の値を持つ複数のパラメータを含むソフトウェアに対して、組合せに含めるアイテムの数を選択して、作成した組合せをテストする。

（例）様々な環境におけるソフトウェアの動作を確認したい。

種類 ➡ 具体値 ➡

OS
Windows
macOS

ブラウザ
Edge
Chrome

バージョン
ver.2.0
ver.3.0

※テスト対象
ソフトウェアの
バージョン

全組合せの場合 ➡

	OS	ブラウザ	バージョン
1	Windows	Edge	ver.2.0
2	Windows	Edge	ver.3.0
3	Windows	Chrome	ver.2.0
4	Windows	Chrome	ver.3.0
5	macOS	Edge	ver.2.0
6	macOS	Edge	ver.3.0
7	macOS	Chrome	ver.2.0
8	macOS	Chrome	ver.3.0

※ASTERセミナー標準テキストを引用して修正
https://aster.or.jp/business/seminar_text.html

クラシフィケーションツリー技法のメリット（１）

- 組み合わせテストをどう作ればいいか分からない時に、テスト対象をクラシフィケーションツリーで分類して整理し、その後、組合せを検討してテストケースを作成できます。

⇒何をどう組み合わせればよいか分からないとき、分かりにくいときに、クラシフィケーションツリー技法が便利です。

クラシフィケーションツリー技法のメリット（２）

- テスト設計からテストケース作成まで、ひとつの図で表現するため、分かりやすい。
  - 複数人でテストする場合、テストの意図を共有しやすい。
  - ユーザがテスト設計を理解しやすく、参加しやすい。

    ⇒理解容易性が高い。

- 特定の箇所について、テストを増やしたり減らしたりできる。

  ⇒テストの濃淡をつけやすい。

  網羅基準に沿った組み合わせテストがしやすい。

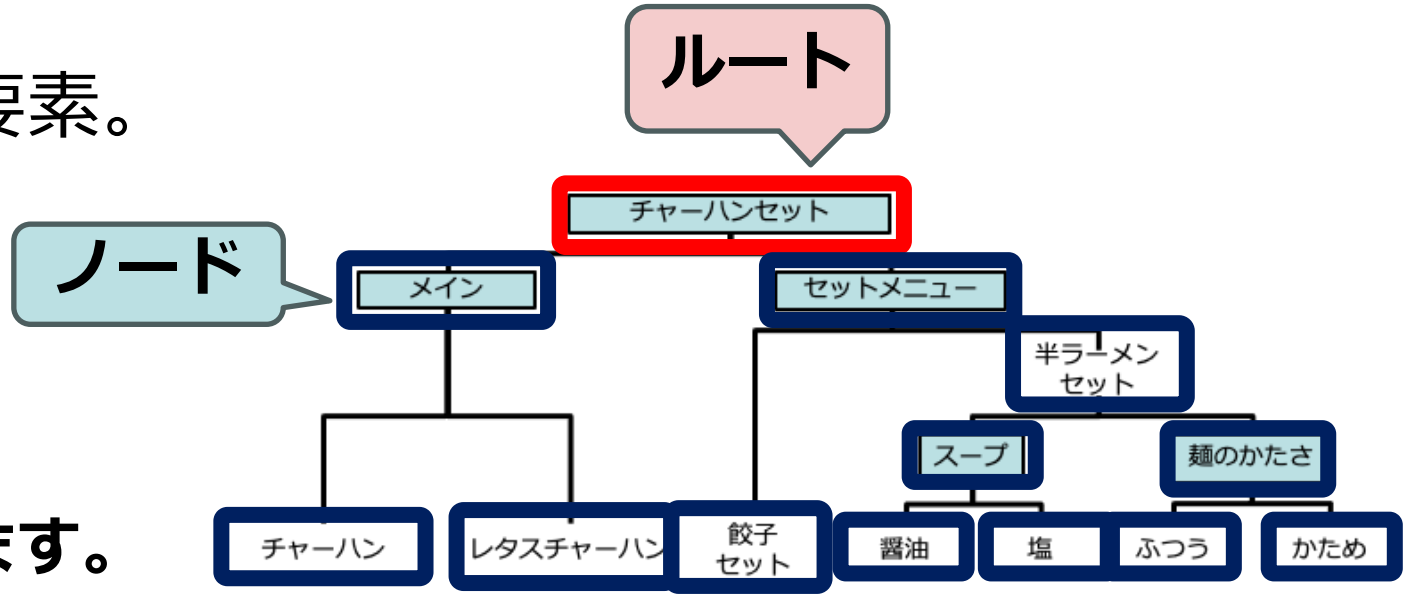
本資料で使用する題材

- 飲食店の注文システムの、チャーハンセットを注文する操作を確認するテストケースを考えましょう。
- メインの種類
  - チャーハン
  - レタスチャーハン
- セットメニュー
  - 餃子セット
  - 半ラーメンセット
    - スープ（醤油、塩）、麺のかたさ（ふつう、かため）

クラシフィケーションツリーに関連する用語・略称

ルート：ツリーの一番上の要素。

ノード：ツリーの要素。



本資料では以下のように表記します。

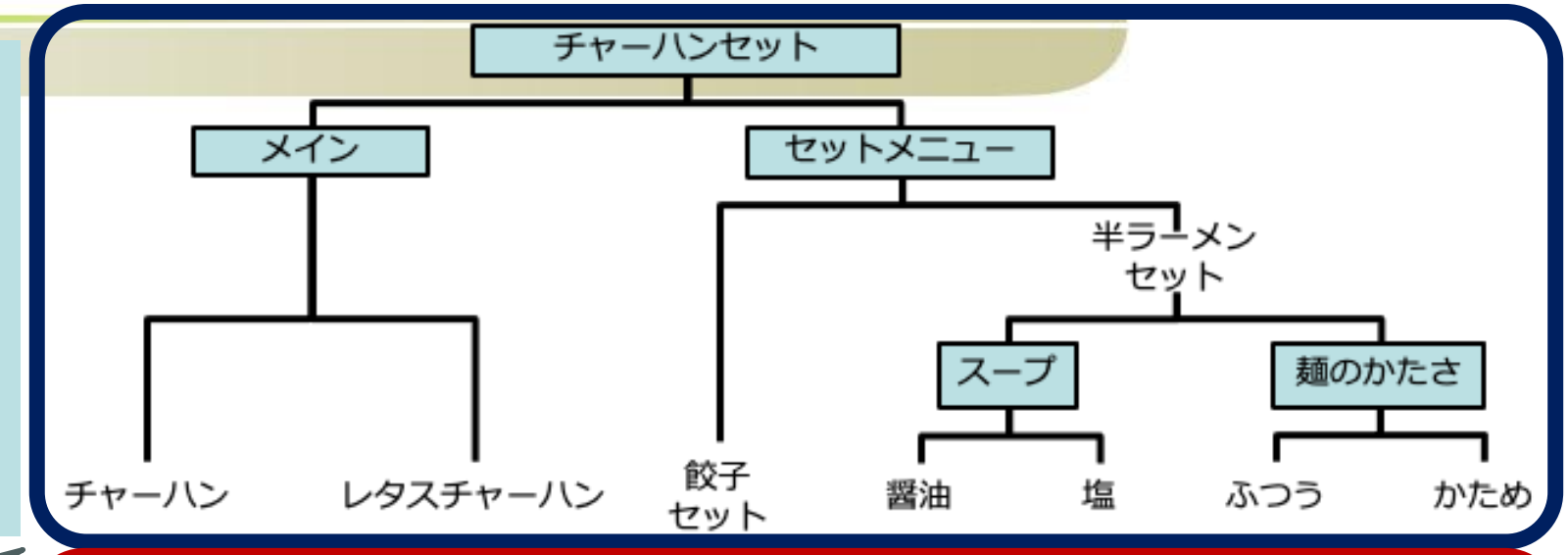
クラシフィケーションツリー技法：「**CT技法**」

クラシフィケーション：「**CLN**」

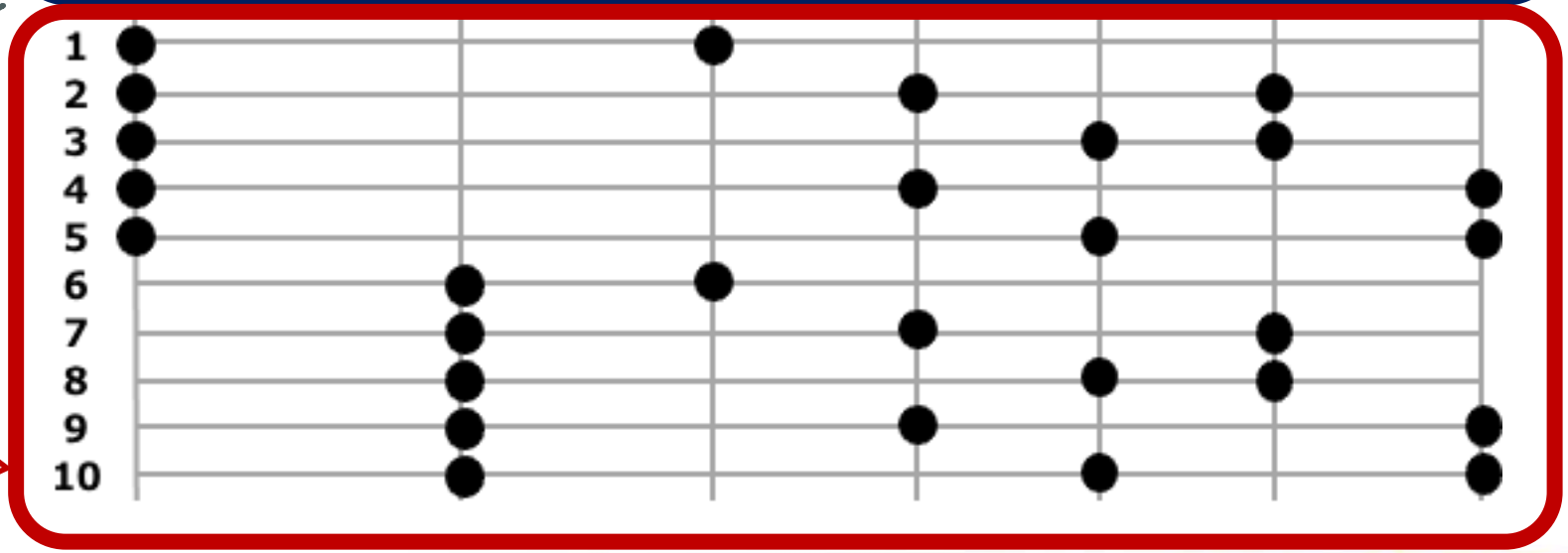
クラス：「**クラス**」（略称なし）

クラシフィケーションツリー技法とは（５）

クラシフィケーションツリーと
組合せテーブルを使用して、
組み合わせテストの
テストケースを作成する技法

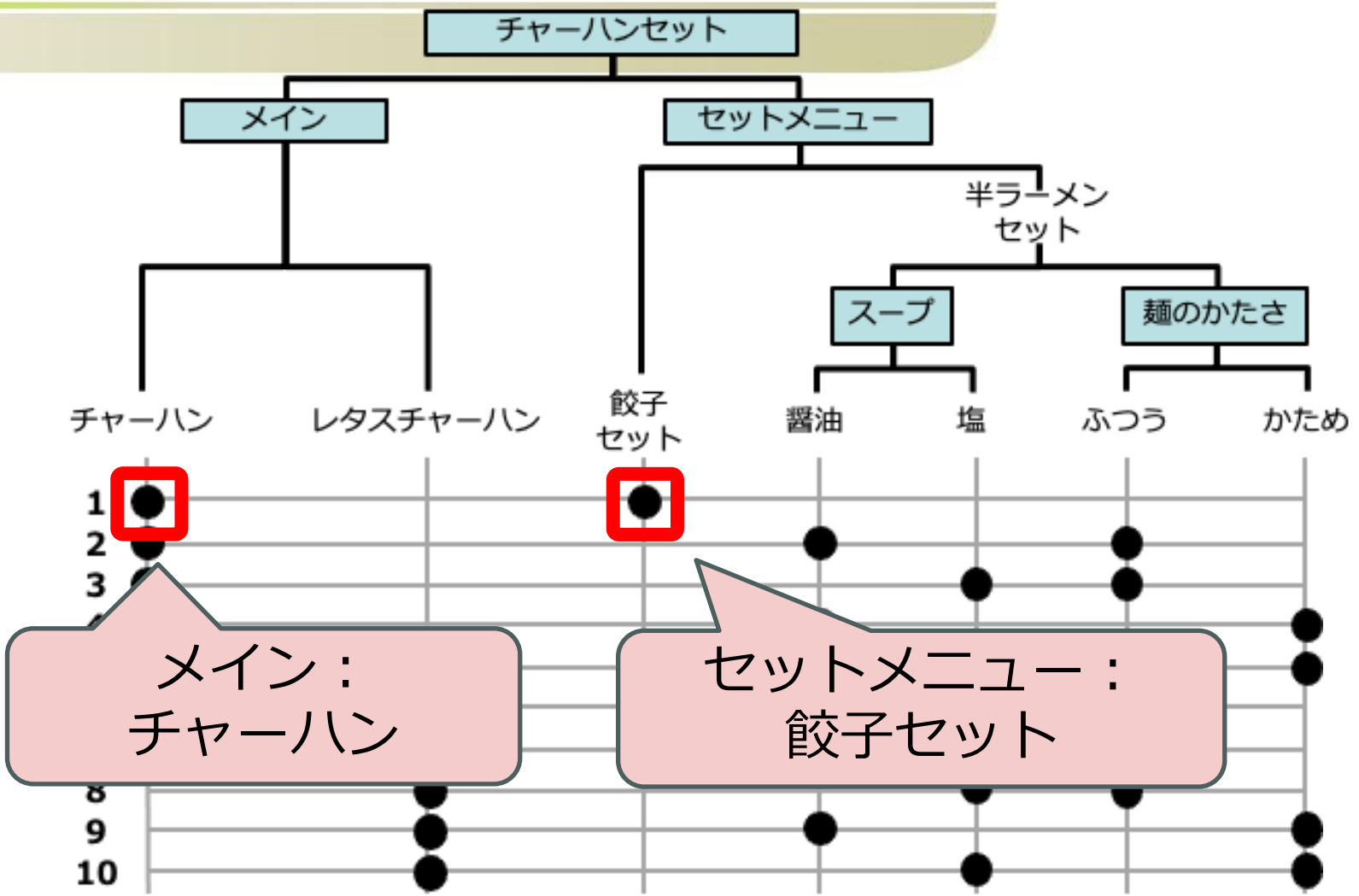


クラシフィケーション
ツリー

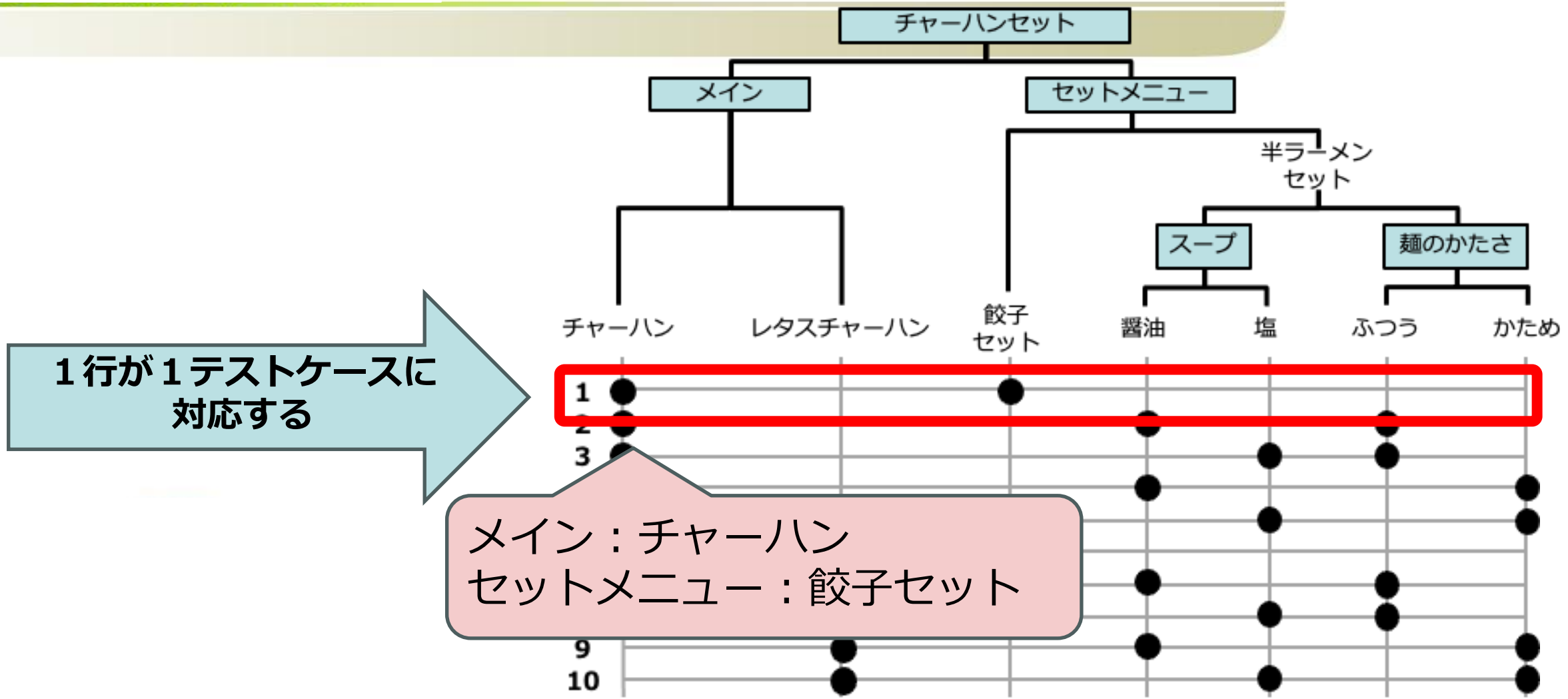


組合せテーブル

クラシフィケーションツリー技法とは（6）



クラシフィケーションツリー技法とは（７）

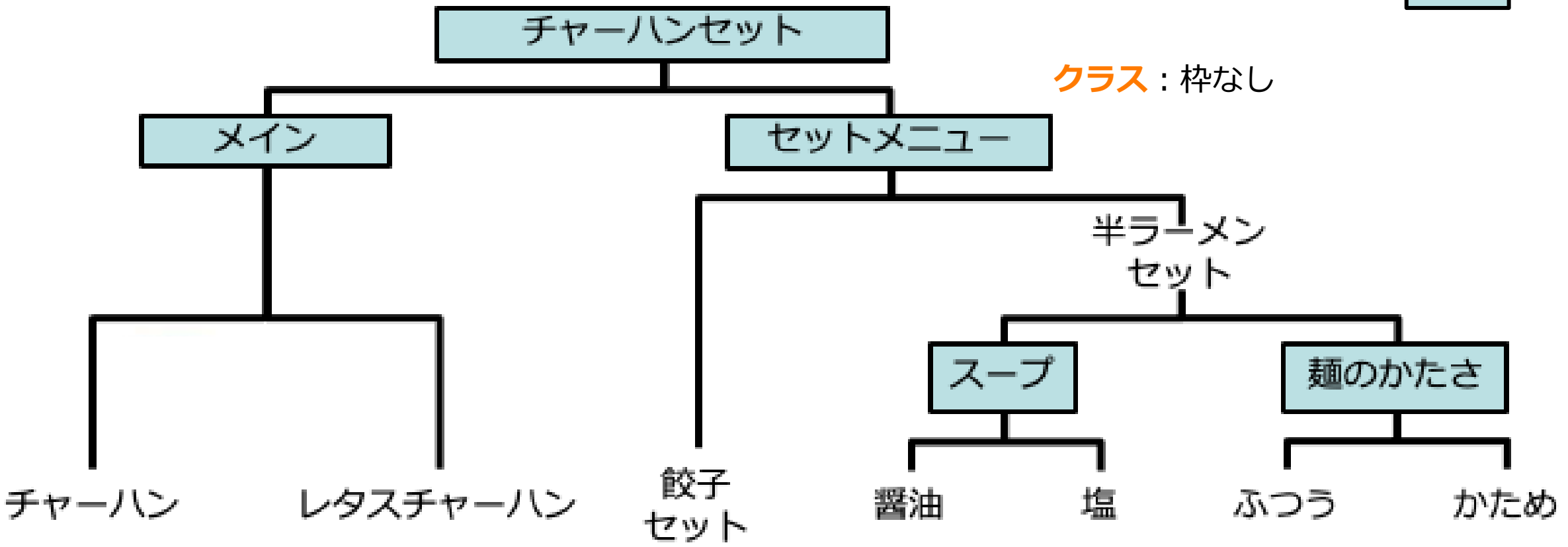


クラシフィケーションとクラス（１）

クラシフィケーションツリーとは：
テスト対象を分類して階層で表したもの

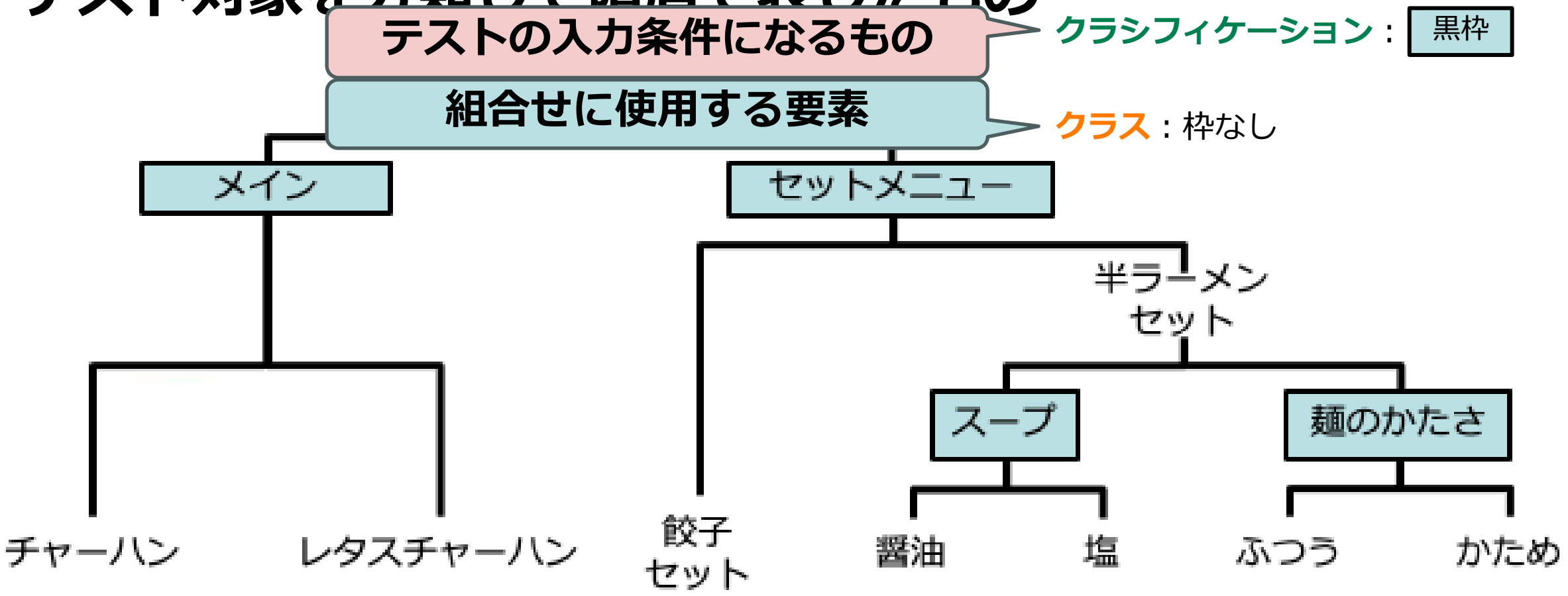
クラシフィケーション：黒枠

クラス：枠なし

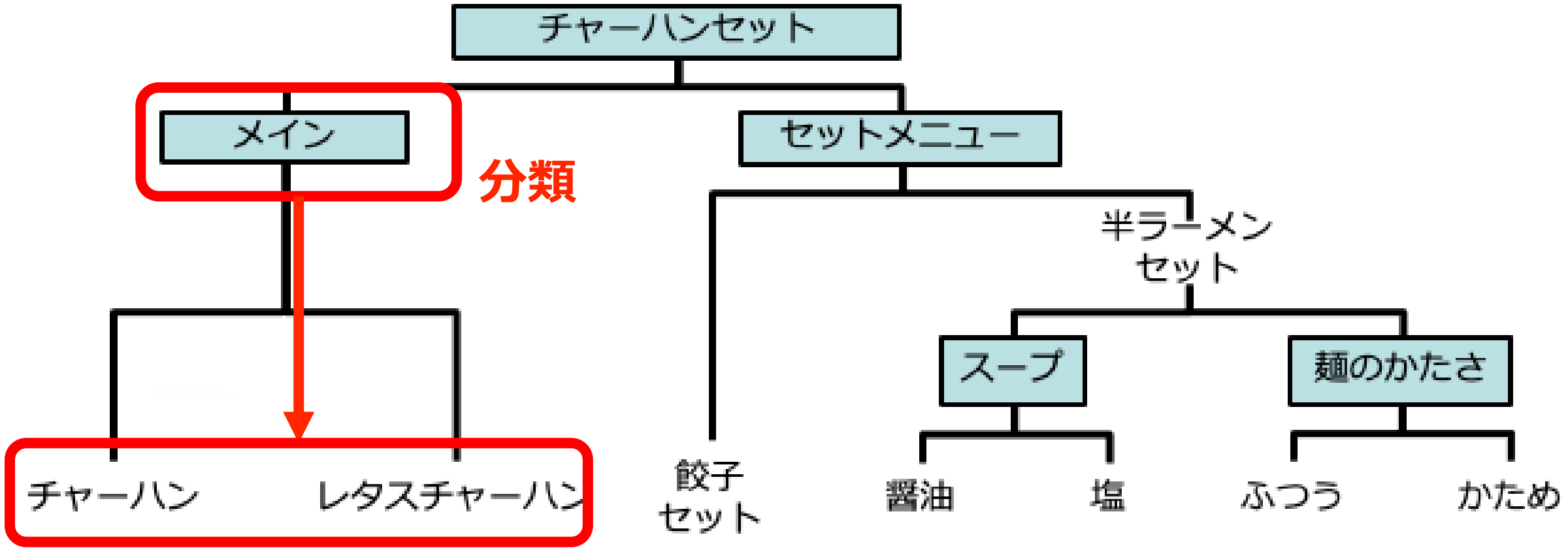


クラシフィケーションとクラス（2）

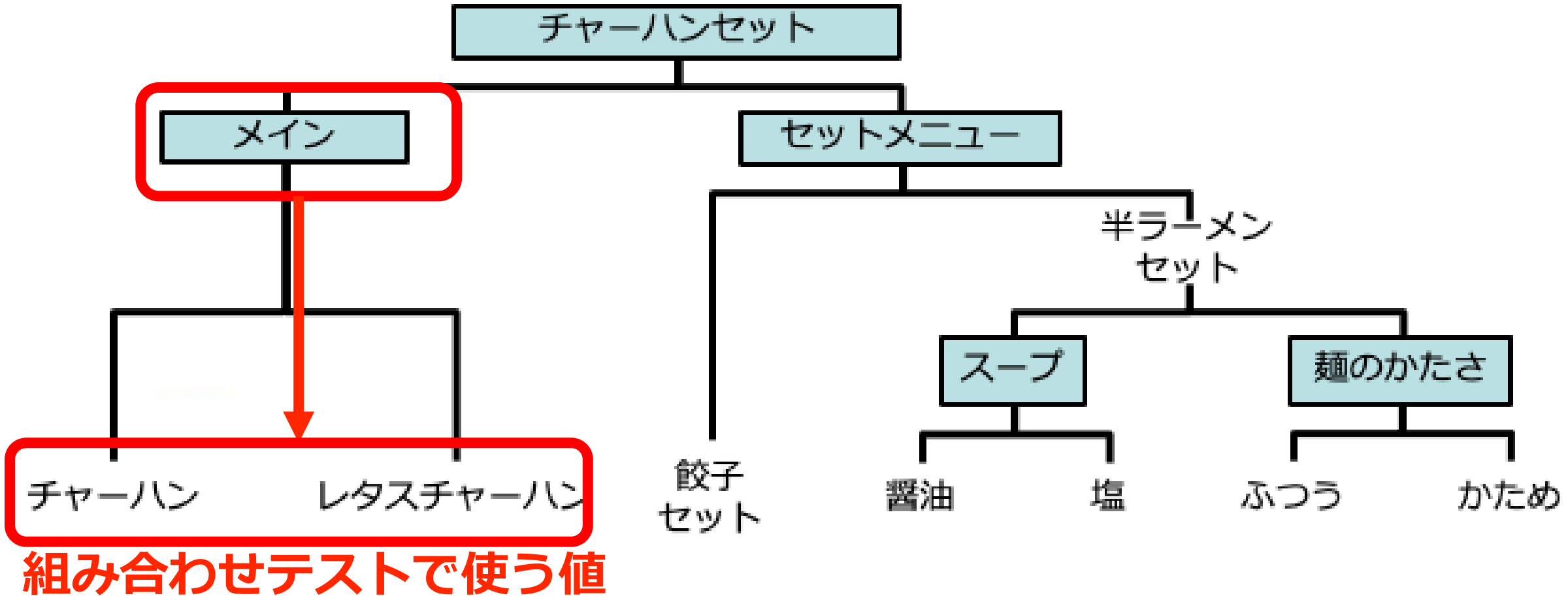
クラシフィケーションツリーとは：
テスト対象を分類して階層で表したもの



クラシフィケーションとクラス（3）

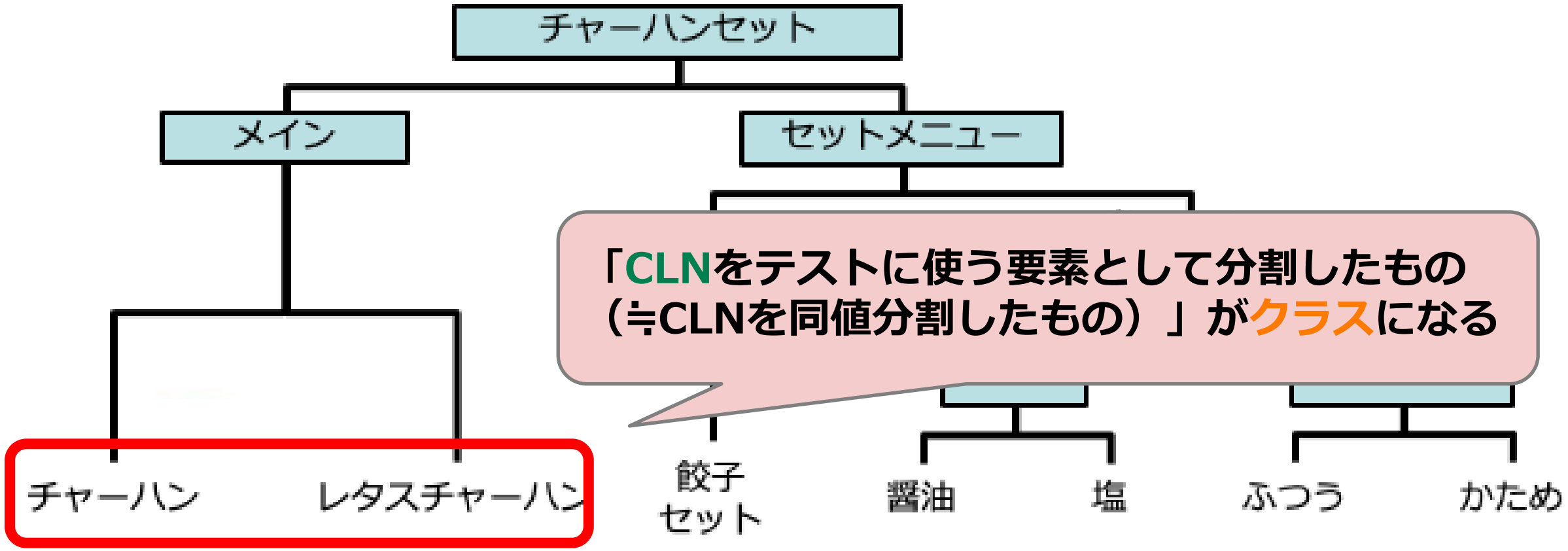


クラシフィケーションとクラス（４）



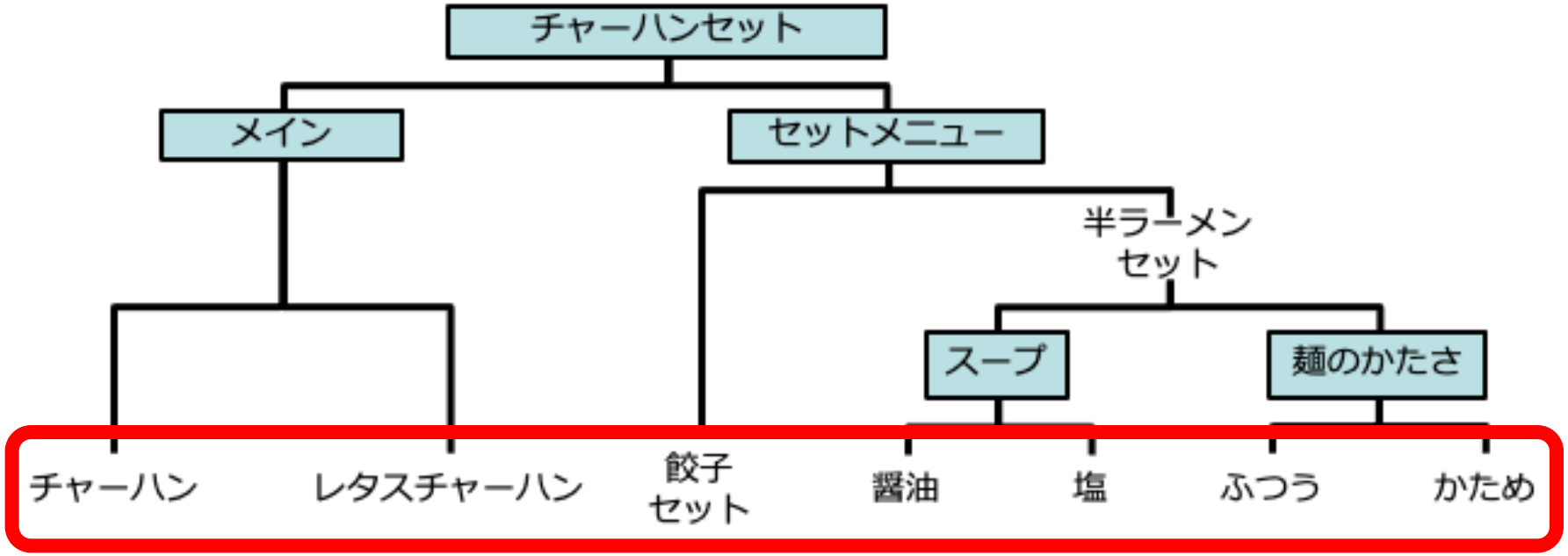
組み合わせテストで使う値

クラシフィケーションとクラス（5）



CT技法によるテストケースの作り方

- テスト対象の構成要素をクラシフィケーション(CLN)として出していく。
- CLNを、木構造として整理する。最下層は組み合わせテストで使う値になる。

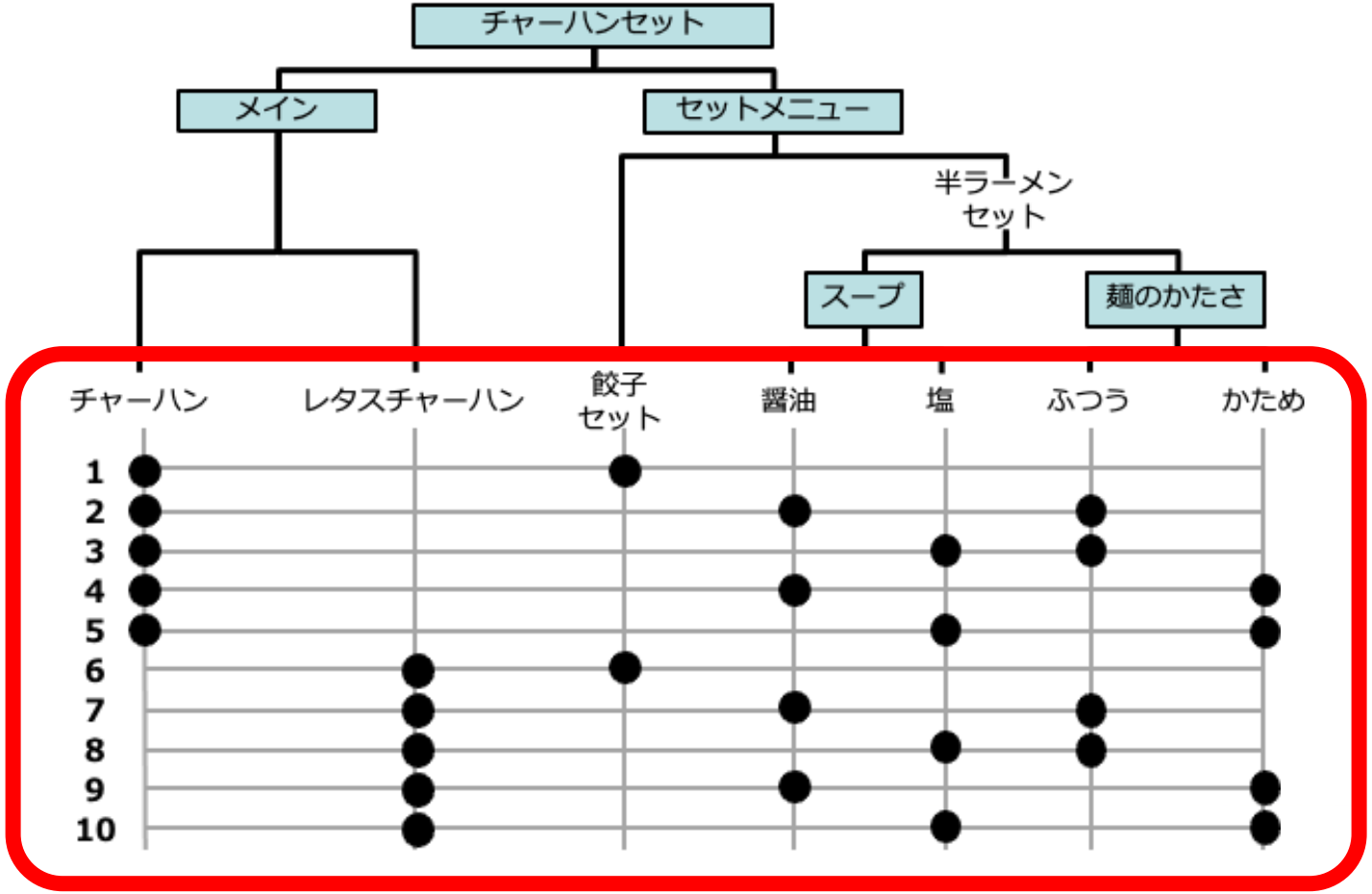


- 木構造として整理
- 最下層は組み合わせテストで使う値

CT技法によるテストケースの作り方

- **ツリーが完成したら組み合わせ方（＝網羅基準）を決めて、組み合わせテストのテストケースを作成する。**

各**クラス**の列に
点を打って
組合せを表現する



組み合わせテストとは（再掲）

※ASTERセミナー標準テキストを引用して修正
https://aster.or.jp/business/seminar_text.html

複数の値を持つ複数のパラメータを含むソフトウェアに対して、組合せに含めるアイテムの数を選択して、作成した組合せをテストする。

（例）様々な環境におけるソフトウェアの動作を確認したい。

種類 ➡

具体値 ➡

OS
Windows
macOS

ブラウザ
Edge
Chrome

バージョン
ver.2.0
ver.3.0

※テスト対象
ソフトウェアの
バージョン

全組合せの場合 ➡

	OS	ブラウザ	バージョン
1	Windows	Edge	ver.2.0
2	Windows	Edge	ver.3.0
3	Windows	Chrome	ver.2.0
4	Windows	Chrome	ver.3.0
5	macOS	Edge	ver.2.0
6	macOS	Edge	ver.3.0
7	macOS	Chrome	ver.2.0
8	macOS	Chrome	ver.3.0

組み合わせテストの網羅基準

- 単純な入力空間網羅基準
  - 末端の各クラスが最低１回存在（Each Choice）
- 組み合わせ網羅基準
  - 2個のCLN間網羅（Pair-Wise）
  - t個のCLN間網羅（t-Wise）
  - 全組合せ（All Combinations）
- 対象ドメイン知識を利用した基準
  - 基本となる組合せを定義する（Base Choice）
  - 基本となる組合せを複数定義する（Multiple Base Choice）

※Ammann & Offutt による分類（Introduction to Software Testing）をもとに作成

組み合わせテストの網羅基準

- **単純な入力空間網羅基準**
  - **末端の各クラスが最低１回存在（Each Choice）**
- **組み合わせ網羅基準**
  - **2個のCLN間網羅（Pair-Wise）**
  - **t個のCLN間網羅（t-Wise）**
  - **全組合せ（All Combinations）**
- **対象ドメイン知識を利用した基準**
  - **基本となる組合せを定義する（Base Choice）**
  - **基本となる組合せを複数定義する（Multiple Base Choice）**

※Ammann & Offutt による分類（Introduction to Software Testing）をもとに作成

組み合わせテストの網羅基準　Each Choice（EC）

	基準	テストケース設定方法	テストケース例
単純な入力空間網羅基準	Each Choice	**未端のクラスが最低１回存在**するようにテストケースを設定する。	

	OS	ブラウザ	バージョン
1	Windows	Edge	ver.2.0
2	macOS	Chrome	ver.3.0

組み合わせテストの設計(PictMaster勉強会)（ https://www.slideshare.net/Bugler/pictmaster-2008717 ）をもとに作成

組み合わせテストの網羅基準

- 単純な入力空間網羅基準
  - 各CLNのクラスが最低１回存在（Each Choice）
- **組み合わせ網羅基準**
  - **2個のCLN間網羅（Pair-Wise）**
  - t個のCLN間網羅（t-Wise）
  - 全組合せ（All Combinations）
- 対象ドメイン知識を利用した基準
  - 基本となる組合せを定義する（Base Choice）
  - 基本となる組合せを複数定義する（Multiple Base Choice）

※Ammann & Offutt による分類（Introduction to Software Testing）をもとに作成

組み合わせテストの網羅基準

	基準	テストケース設定方法	テストケース例
組み合わせ網羅基準	Pair-Wise	**任意の２つのCLN間で全てのクラスの組み合わせが存在する**ようにテストケースを設定する。 ※右記の例では、 Windows と Edge macOS と Edge Windows と Chrome macOS と Chrome Edge と ver.2.0 Edge と ver.3.0 Chrome と ver.2.0 Chrome と ver.3.0 ver.2.0 と Windows ver.2.0 と macOS ・・・ といった２CLN間の組合せが網羅されている。	

	OS	ブラウザ	バージョン
1	Windows	Edge	ver.2.0
2	macOS	Edge	ver.3.0
3	Windows	Chrome	ver.2.0
4	macOS	Chrome	ver.3.0
5	Windows	Edge/ Chrome	ver.3.0
6	macOS	Edge/ Chrome	ver.2.0

組み合わせテストの設計(PictMaster勉強会)（ https://www.slideshare.net/Bugler/pictmaster-2008717 ）をもとに作成

組み合わせテストの網羅基準

- 単純な入力空間網羅基準
  - 各CLNの状態値が最低１回存在（Each Choice）
- **組み合わせ網羅基準**
  - 2個のCLN間網羅（Pair-Wise）
  - **t個のCLN間網羅（t-Wise）**
  - **全組合せ（All Combinations）**
- 対象ドメイン知識を利用した基準
  - 基本となる組合せを定義する（Base Choice）
  - 基本となる組合せを複数定義する（Multiple Base Choice）

※Ammann & Offutt による分類（Introduction to Software Testing）をもとに作成

組み合わせテストの網羅基準

<table>
<tr><th></th><th>基準</th><th>テストケース設定方法</th><th>テストケース例(*)</th></tr>
<tr><td rowspan="2">組み合わせ
網羅基準</td><td>t-Wise</td><td><u>t個のCLN間でクラスの組み合わせが全て存在する</u>ようにテストケースを設定する。</td><td></td></tr>
<tr><td>All
Combinations</td><td><u>全ての末端のクラスの組み合わせが存在する</u>ようにテストケースを設定する。</td><td>２×２×２の８通り
<table>
<tr><th></th><th>OS</th><th>ブラウザ</th><th>バージョン</th></tr>
<tr><td>1</td><td>Windows</td><td>Edge</td><td>ver.2.0</td></tr>
<tr><td>2</td><td>Windows</td><td>Edge</td><td>ver.3.0</td></tr>
<tr><td>3</td><td>Windows</td><td>Chrome</td><td>ver.3.0</td></tr>
<tr><td>4</td><td>Windows</td><td>Chrome</td><td>ver.2.0</td></tr>
<tr><td>5</td><td>macOS</td><td>Edge</td><td>ver.2.0</td></tr>
<tr><td>6</td><td>macOS</td><td>Edge</td><td>ver.3.0</td></tr>
<tr><td>7</td><td>macOS</td><td>Chrome</td><td>ver.3.0</td></tr>
<tr><td>8</td><td>macOS</td><td>Chrome</td><td>ver.2.0</td></tr>
</table></td></tr>
</table>

組み合わせテストの設計(PictMaster勉強会)（ https://www.slideshare.net/Bugler/pictmaster-2008717 ）をもとに作成

組み合わせテストの網羅基準

- 単純な入力空間網羅基準
  - 各CLNの状態値が最低１回存在（Each Choice）
- 組み合わせ網羅基準
  - 2個のCLN間網羅（Pair-Wise）
  - t個のCLN間網羅（t-Wise）
  - 全組合せ（All Combinations）
- **対象ドメイン知識を利用した基準**
  - **基本となる組合せを定義する（Base Choice）**
  - **基本となる組合せを複数定義する（Multiple Base Choice）**

※Ammann & Offutt による分類（Introduction to Software Testing）をもとに作成

組み合わせテストの網羅基準

<table>
<tr><th></th><th>基準</th><th>テストケース設定方法</th><th>テストケース例</th></tr>
<tr><td rowspan="2">対象ドメイン知識を利用した基準</td><td>Base Choice</td><td><u>各CLNについて基本的なクラス（例えば利用者が通常使うもの）を選択し、これを基本テストケース（base choice）とする</u>。以降、一つのCLNを除く他のCLNのクラスを固定して、対象のCLNのクラスを変えてテストケースを設定する。</td><td>base choice ┌ OS:Windows
ブラウザ:Edge
バージョン:2.0
└
<table><tr><th></th><th>OS</th><th>ブラウザ</th><th>バージョン</th></tr><tr><td>1</td><td>Windows</td><td>Edge</td><td>ver.2.0</td></tr><tr><td>2</td><td>macOS</td><td>Edge</td><td>ver.2.0</td></tr><tr><td>3</td><td>Windows</td><td>Chrome</td><td>ver.2.0</td></tr><tr><td>4</td><td>Windows</td><td>Edge</td><td>ver.3.0</td></tr></table>← base</td></tr>
<tr><td>Multiple Base Choice</td><td>base choiceを<u>二つ以上定義</u>し、各々base choiceと同様にテストケースを設定する。</td><td></td></tr>
</table>

組み合わせテストの設計(PictMaster勉強会)（ https://www.slideshare.net/Bugler/pictmaster-2008717 ）をもとに作成

CT技法によるテストケース作成例（パターン1）

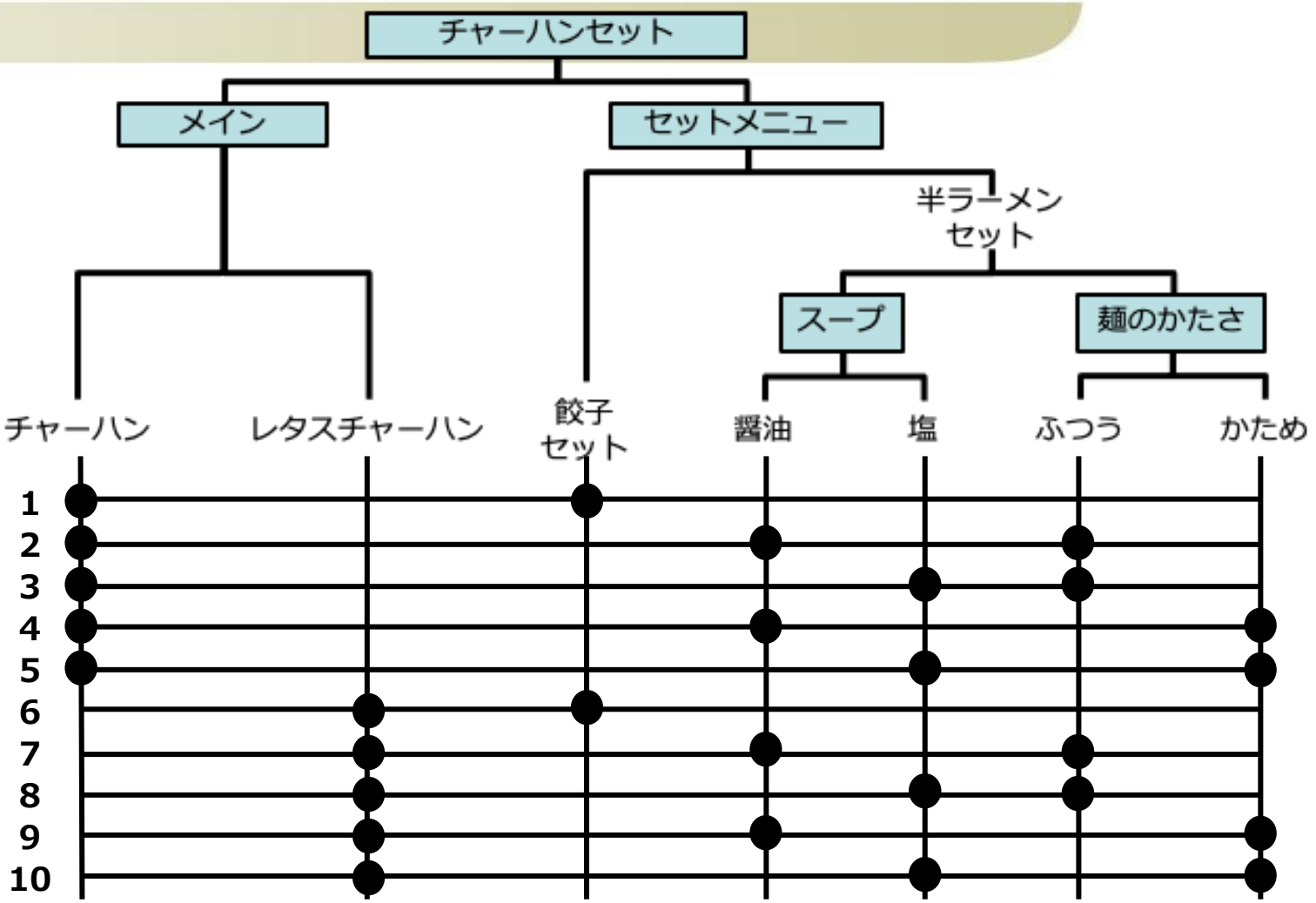
- メイン、セットメニュー、**それぞれの組み合わせで正しく注文の操作ができるか**確認したい。

- **組み合わせ方を決める**
  - 全組合せ
  - Pair-Wise
  - その他

. 新規開発する機能なので、**全組合せ**でテストする。

CT技法によるテストケース作成例（パターン1）

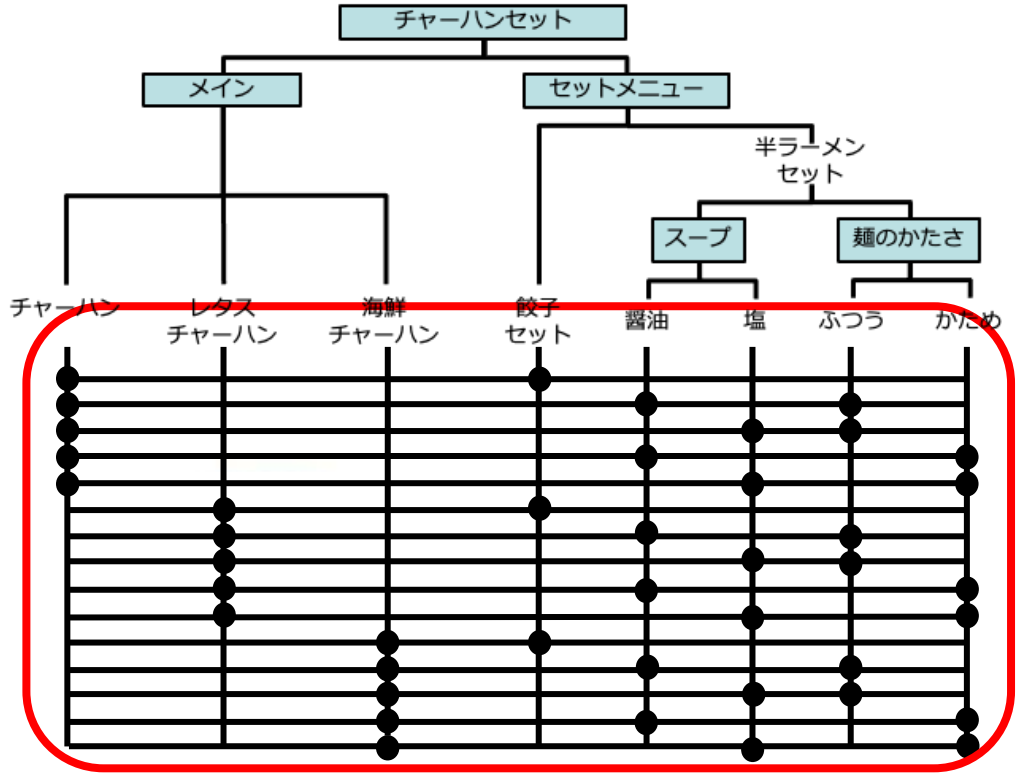
新規開発する機能なので、
全組合せでテストする



CT技法によるテストケース作成例（パターン2）

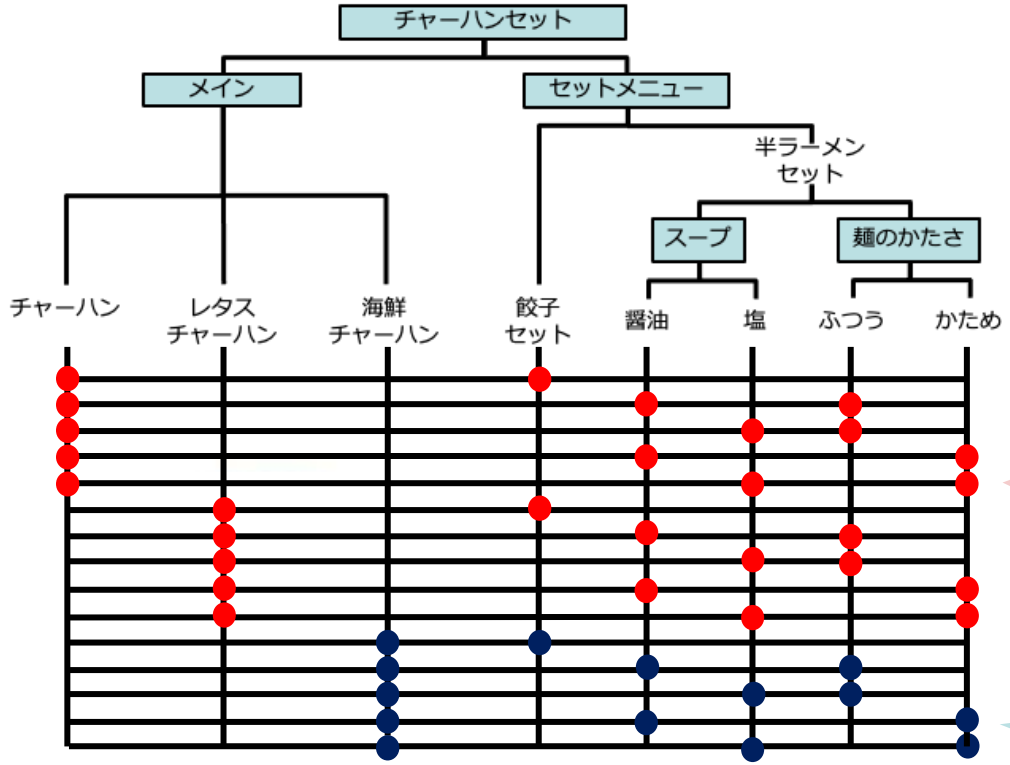
- チャーハンセットに海鮮チャーハンを追加した。
- チャーハンやレタスチャーハンは変更していないので、**1ケースだけ**確認し、海鮮チャーハンを選択した場合は**全組合せ**の動作を確認する。

CT技法によるテストケース作成例（パターン2）



全組合せで作成すると
3×5で15件の
テストケースになる

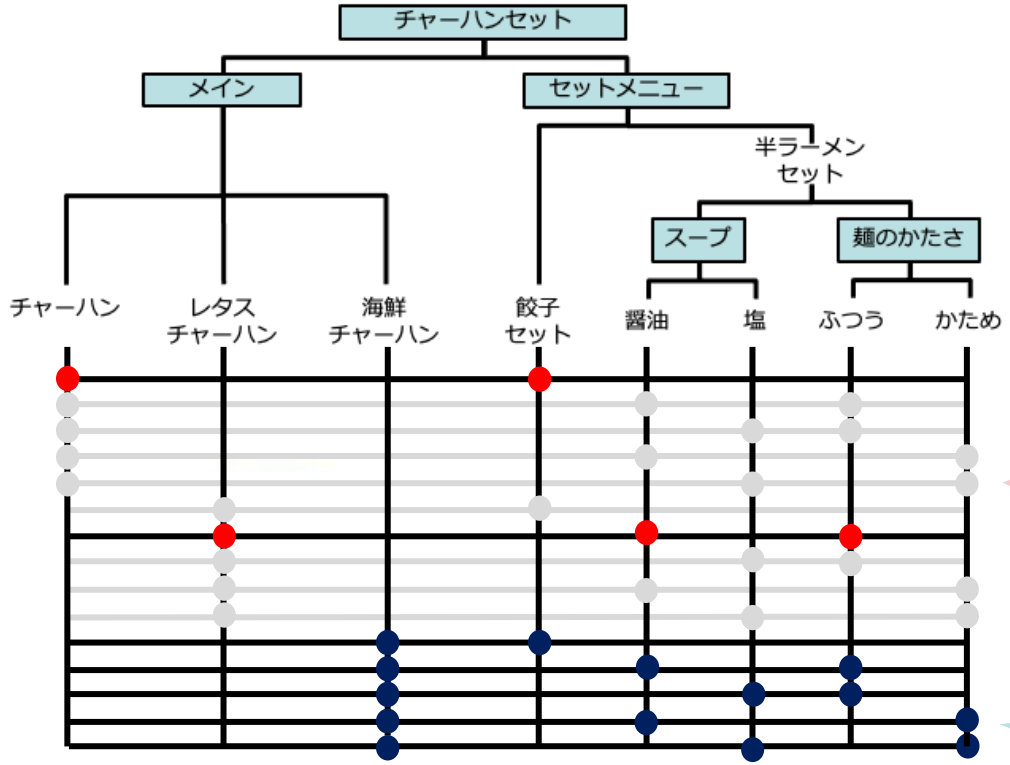
CT技法によるテストケース作成例（パターン2）



「チャーハン」
「レタスチャーハン」は
変更していない

「海鮮チャーハン」は
新規追加

CT技法によるテストケース作成例（パターン2）

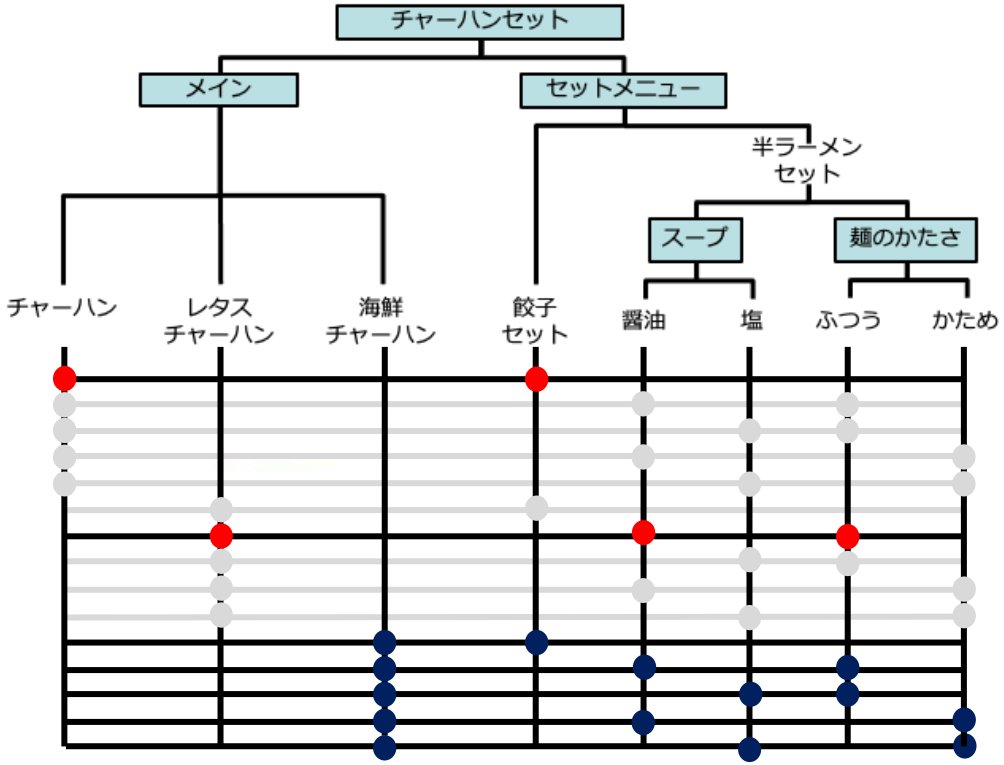


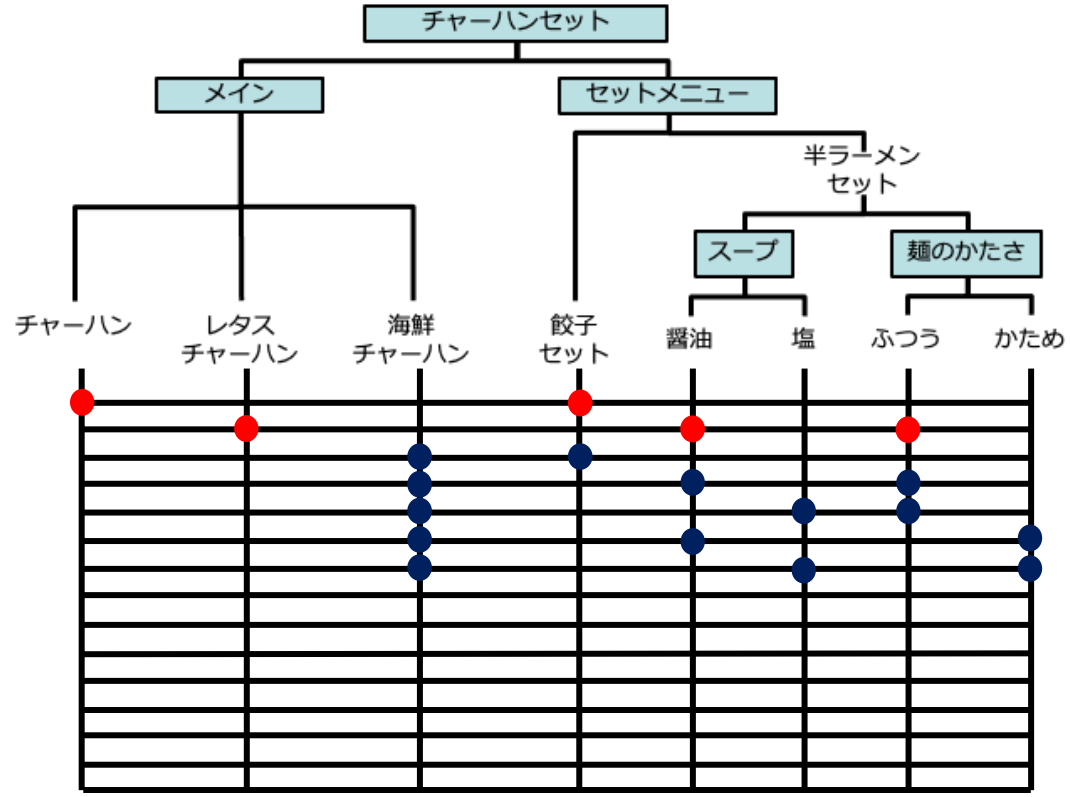
「チャーハン」
「レタスチャーハン」は
変更していない
→**<u>各1ケース</u>**確認する

「海鮮チャーハン」は
新規追加
→**<u>全組合せ</u>**で確認する

CT技法によるテストケース作成例（パターン2）

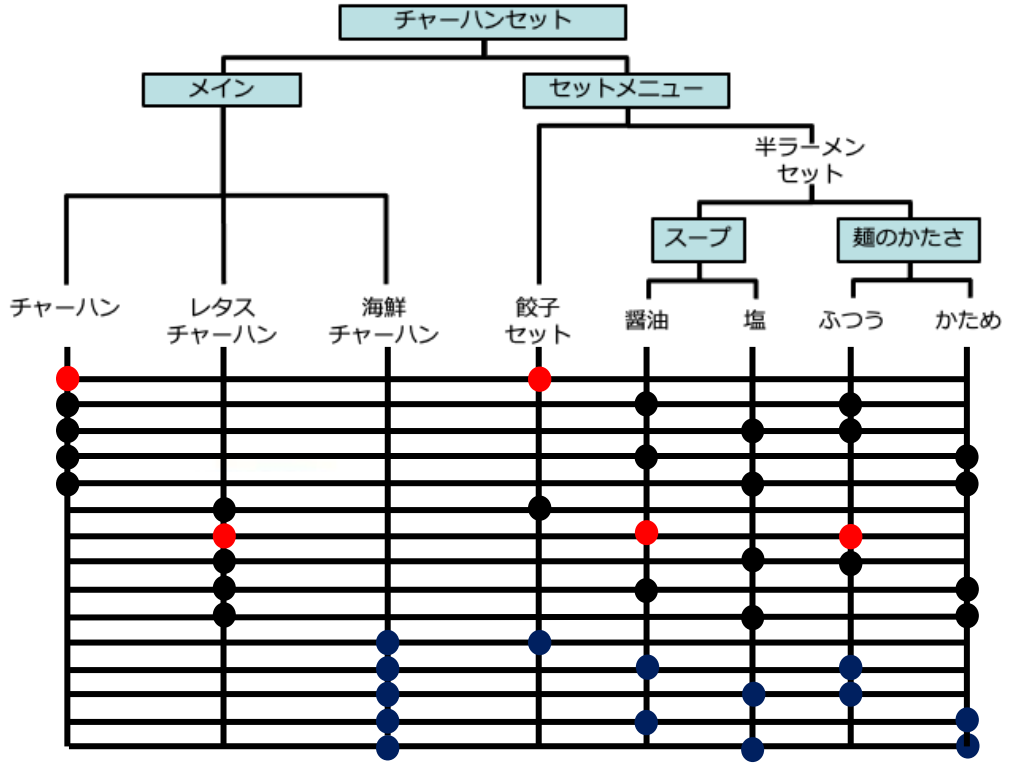
完成したテストケース



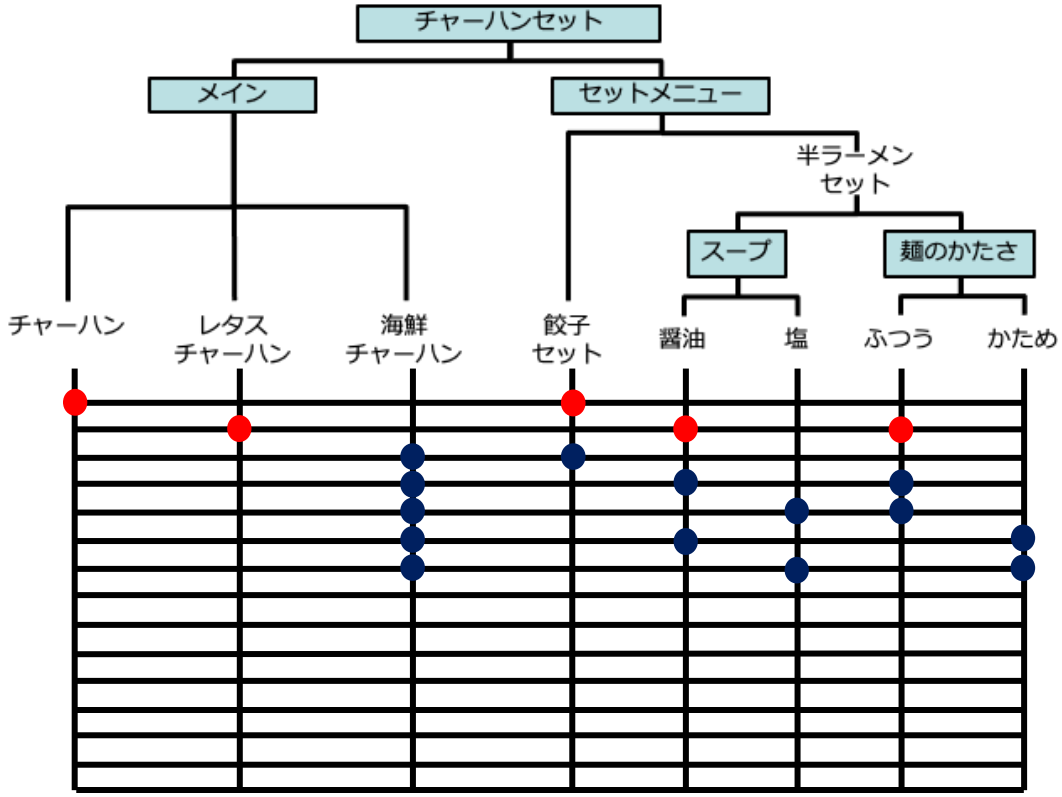


CT技法によるテストケース作成例（パターン2）

変更前のテストケース
（15件）

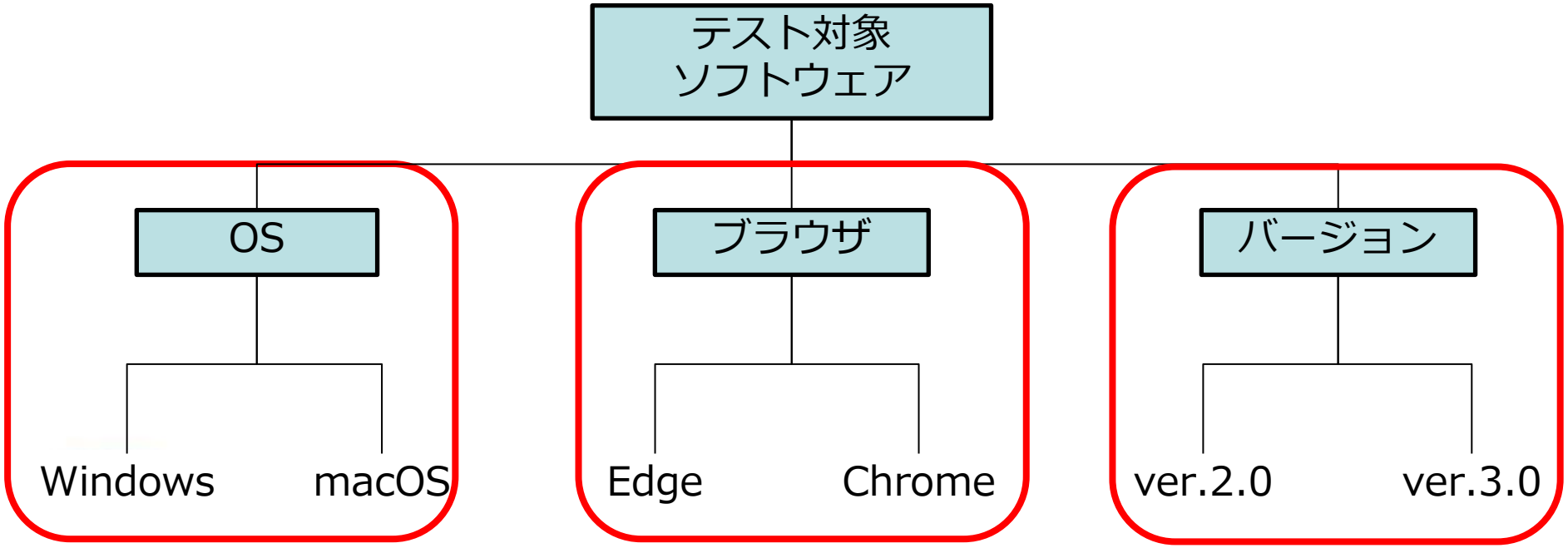


完成したテストケース
（7件）



組み合わせの網羅基準の決め方

CLN（赤枠）ごとに組み合わせの網羅基準を決めることができます



組み合わせの網羅基準の決め方

（例）「OS」「ブラウザ」を全組合せ
「バージョン」をEach Choice（最低１回ずつ登場）にした場合

CLN ➡

クラス ➡

OS
Windows
macOS

ブラウザ
Edge
Chrome

バージョン
ver.2.0
ver.3.0

※テスト対象
ソフトウェアの
バージョン

全組合せ ➡

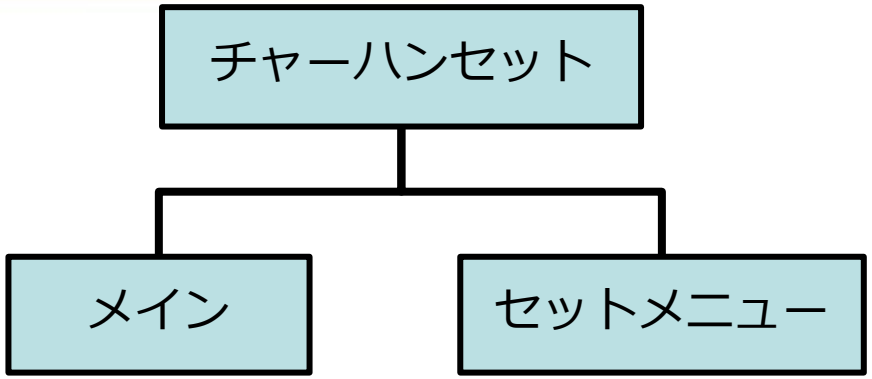
Each Choice ➡

	OS	ブラウザ	バージョン
1	Windows	Edge	ver.2.0
2	Windows	Chrome	ver.3.0
3	macOS	Edge	ver.3.0
4	macOS	Chrome	ver.3.0

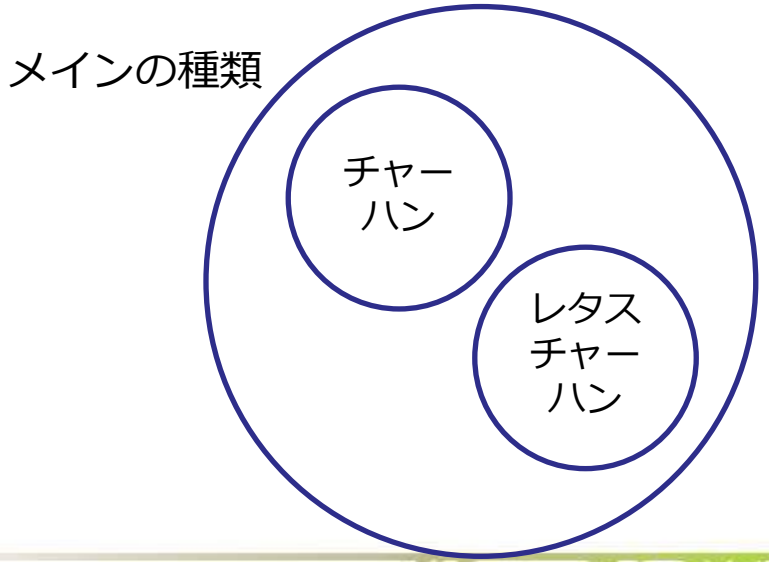
※ASTERセミナー標準テキストを引用して修正
https://aster.or.jp/business/seminar_text.html

クラシフィケーション(CLN)とクラス

- チャーハンセットは、「メイン」と「セットメニュー」から構成されている→ has-a

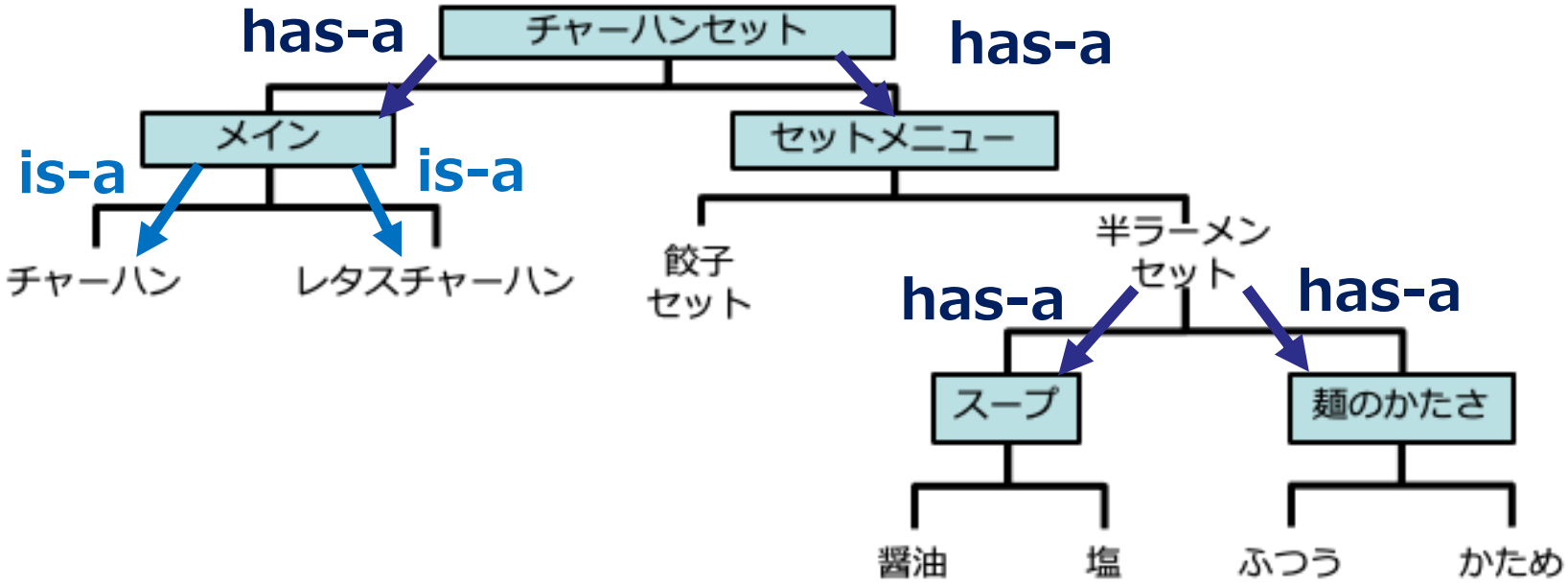


- メインの種類には「チャーハン」と「レタスチャーハン」がある→ is-a



クラシフィケーション(CLN)とクラス

- CLN→CLNは「has-a」の関係
- CLN→クラスは「is-a」の関係
- クラス→CLNは「has-a」の関係



※**ツリーの末端（最下層）は必ずクラスになります**。
クラシフィケーションで終わることはありません

has-aとis-aとは

- 「has-a」とは、構成要素になっている関係（has-a：包含関係）を表したものです
  - 「CLN ― CLN」または「クラス ― CLN」で表します
    - チャーハンセット ー メイン
    - セットメニュー ー 半ラーメンセット

- 「is-a」とは、抽象的なものと具体的なものの関係（is-a：継承関係）を表したものです
  - 「CLN ― クラス」で表します
    - メイン ー レタスチャーハン

CT技法のメリット（１）再掲

- 組み合わせテストをどう作ればいいか分からない時に、テスト対象をクラシフィケーションツリーで分類して整理し、その後、組合せを検討してテストケースを作成できます。

⇒何をどう組み合わせればよいか分からないとき、分かりにくいときに、クラシフィケーションツリー技法が便利です。

CT技法のメリット（２）再掲

- テスト設計からテストケース作成まで、ひとつの図で表現するため、分かりやすい。
  - 複数人でテストする場合、テストの意図を共有しやすい。
  - ユーザがテスト設計を理解しやすく、参加しやすい。

  ⇒理解容易性が高い。

- 特定の箇所について、テストを増やしたり減らしたりできる。

  ⇒テストの濃淡をつけやすい。

  網羅基準に沿った組み合わせテストがしやすい。

CT技法が向いているケース、向かないケース

■ 向いているケース

- テストで何をどう組み合わせればいいか分からない・複雑で整理が必要な場合。
  ⇒クラシフィケーションツリーでテスト対象を分類・整理し、その後、どう組合せを作るかテスト設計できる。

■ 向いていないケース

- テストで何をどう組み合わせればいいか、既に分かっている場合。
  ⇒デシジョンテーブルなど、他のテスト技法を使うと良い場合がある。

（参考）JaSST東北2019のデシジョンテーブルワークショップ資料
https://www.jasst.jp/symposium/jasst19tohoku/pdf/S5-1.pdf

タイムテーブル

開始時刻	終了時刻	内容
15:20	15:35	CT技法の説明
15:35	16:15	作り方の説明
16:15	16:40	練習問題1
16:40	17:05	練習問題2
17:05	17:10	まとめ

CT技法によるテストケースの作り方

例題をもとに、CT技法によるテストケースの作り方を説明

1. テスト対象を選択する
2. クラシフィケーションツリーを作成する
3. 組合せテーブルを作成する
4. テストデータ、期待結果を定義する

例題：肉屋トントンのギュウ丼セット

肉屋トントンの「ギュウ丼セット」の価格（税込）を確認する
テストケースをクラシフィケーションツリーを用いて考えよう

ギュウ丼セット：600円
セットには以下が含まれます。
・ギュウ丼（並盛）
・みそ汁

ギュウ丼は ＋100円 で大盛にできます
みそ汁は＋120円でトン汁に変更できます
ギュウ丼セットを注文の場合、追加でドリンクを注文できます
・コーラ、ウーロン茶、オレンジジュース（＋100円）
・ホットコーヒー、アイスコーヒー（＋150円）

- ギュウ丼

並盛
+0円

大盛
+100円

- 汁物

みそ汁
+0円

トン汁
+120円

- ドリンク

なし
+0円

あり

・100円

コーラ
+100円

ウーロン茶
+100円

オレンジジュース
+100円

・150円

ホットコーヒー
＋150円

アイスコーヒー
＋150円

注文内容:
ギュウ丼セット
・ギュウ丼（並盛）
・みそ汁
・ドリンクなし

金額（税込）:
600 円

数量:
- 1 +

戻る

注文

- **ギュウ丼**

並盛
+0円

大盛
+100円

- **汁物**

みそ汁
+0円

トン汁
+120円

- **ドリンク**

なし
+0円

あり

・**100円**

コーラ
+100円

ウーロン茶
+100円

オレンジジュース
+100円

・**150円**

ホットコーヒー
＋150円

アイスコーヒー
＋150円

注文内容:
ギュウ丼セット
・ギュウ丼(並盛)
・みそ汁
・ドリンク(コーラ)

金額(税込):
700 円

数量:
- 1 +

戻る

注文

CT技法によるテストケースの作り方

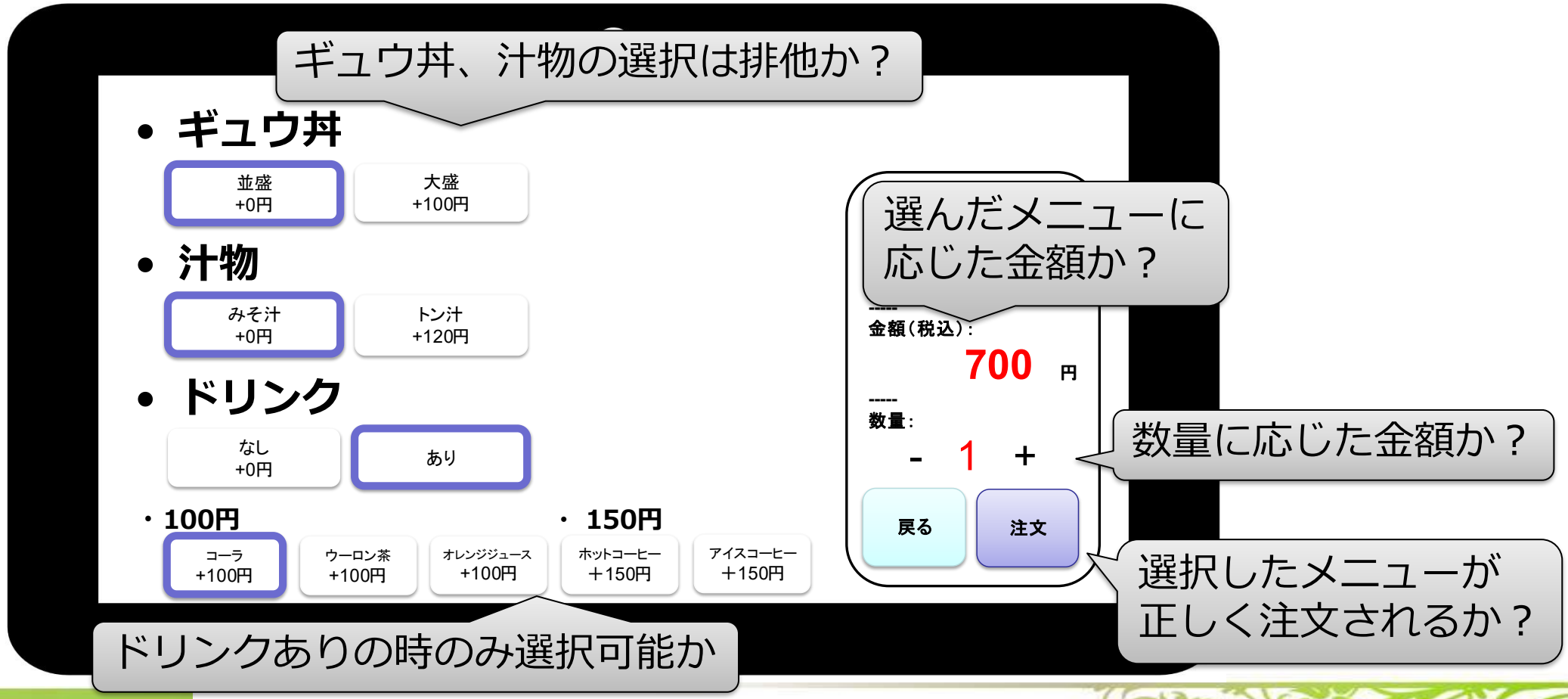
1. テスト対象を選択する
2. クラシフィケーションツリーを作成する
3. 組合せテーブルを作成する
4. テストデータ、期待結果を定義する

CT技法によるテストケースの作り方

1. テスト対象を選択する
2. クラシフィケーションツリーを作成する
3. 組合せテーブルを作成する
4. テストデータ、期待結果を定義する

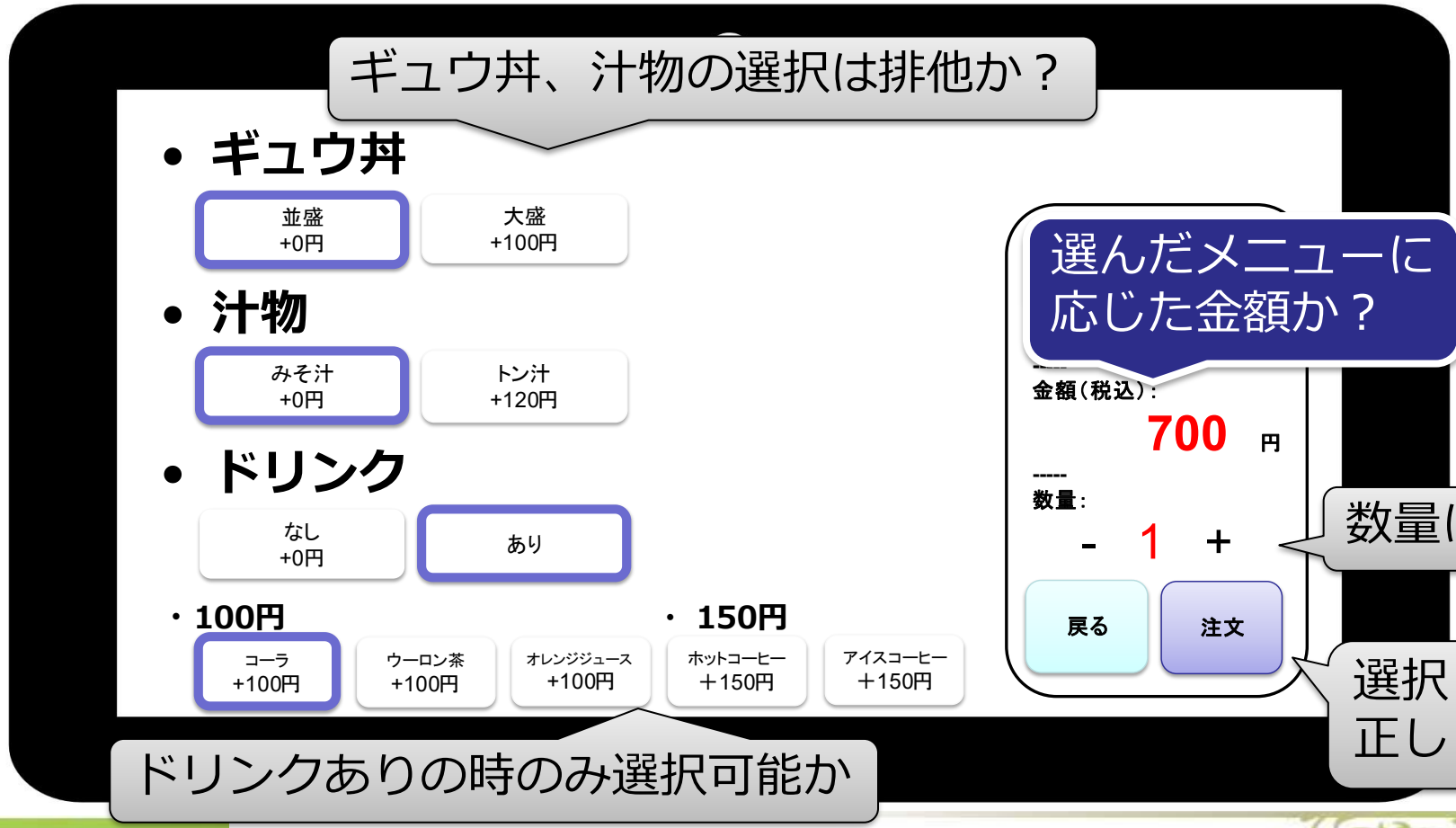
テストで確認したいことは何か？

テストで確認したいことを洗い出します



組み合わせテストで確認したいのはどれ？

組み合わせテストを行いたい（≒CT技法で整理したい）対象を選択



1. テスト対象を選択する

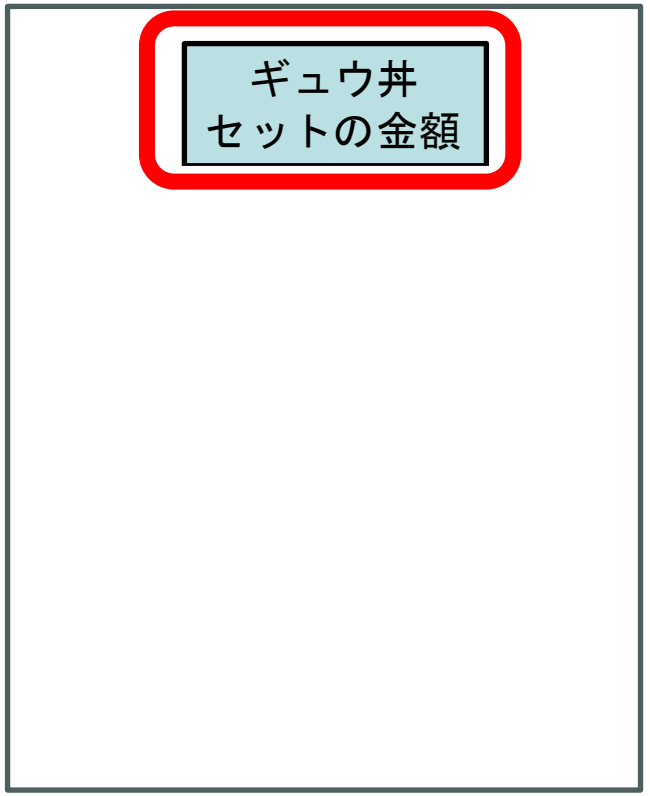
- テスト対象はCT技法で設計したい対象に合わせる
  - 製品／機能／テストで確認したいこと
- 木構造の一番上（ルート、親ノードがないノード）にテスト対象を置く
  - テスト対象は**CLN**となる

【ワーク】1. テスト対象を選択する

模造紙にテスト対象として**「ギュウ丼セットの金額」**付箋を置こう

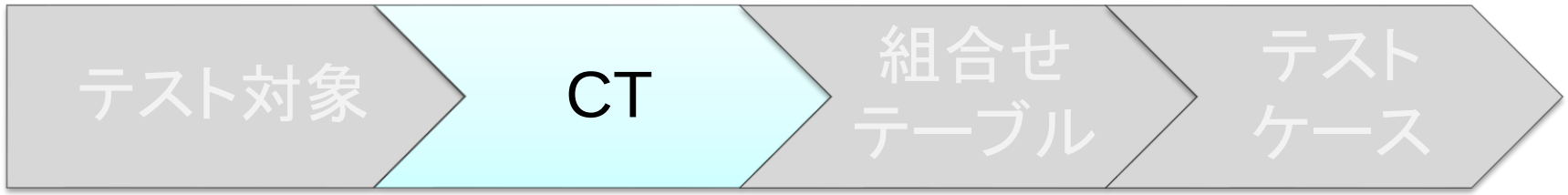
- 模造紙の最上段あたりに**緑色の付箋**を置こう
- 模造紙が丸まる場合は4つ角をテープで止めよう

※ 今回のワークは実施内容を揃えるため、
テスト対象は「ギュウ丼セットの金額」
にしています



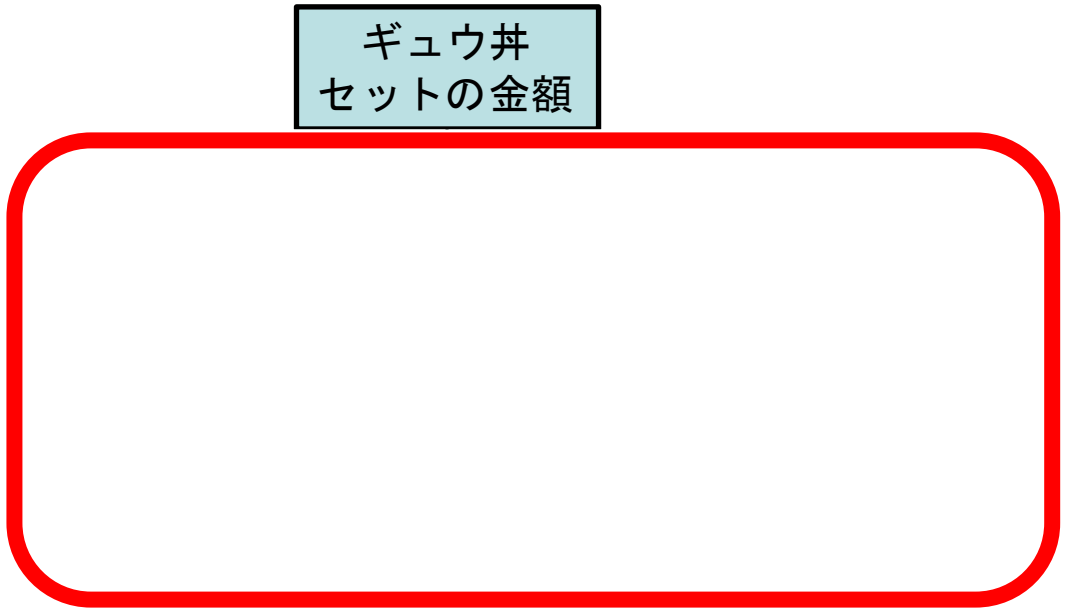
CT技法によるテストケースの作り方

1. テスト対象を選択する
2. クラシフィケーションツリーを作成する
3. 組合せテーブルを作成する
4. テストデータ、期待結果を定義する



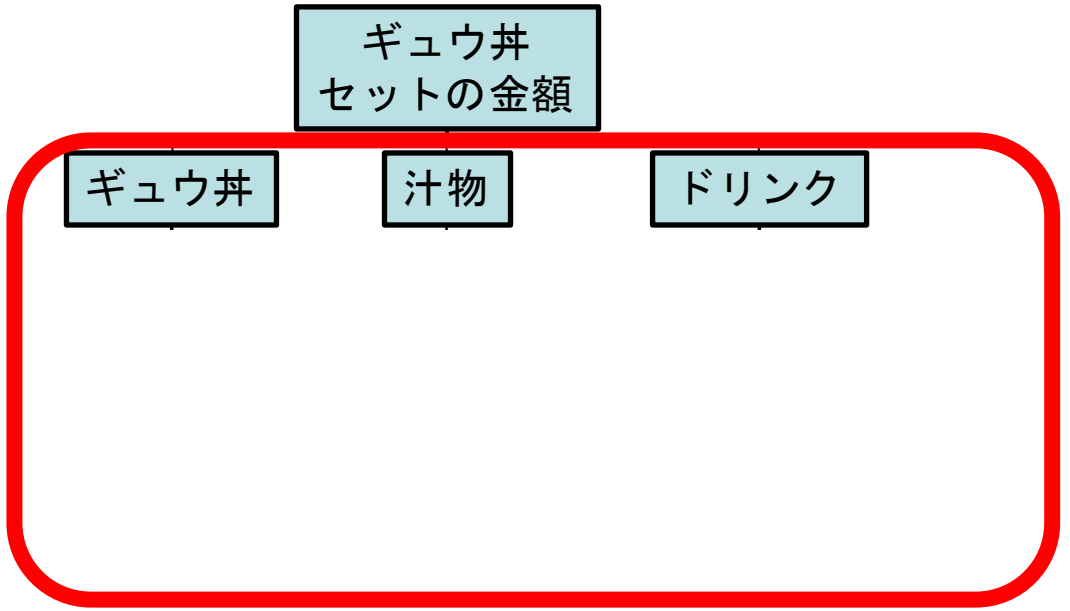
2. クラシフィケーションツリーを作る(1/3)

- **CLN**と**クラス**に分けてノードを書き出す
- **CLN**は「テストしたいこと」を書く
  - 例えば「汁物」「ドリンク」など
- **クラス**は「**CLN**をテストに使う要素として分割したもの」を書く
  - 例えば「みそ汁」「トン汁」
  - **CLN**に対して「不足なく」「重複なく」分割する
- テストの入力に関する項目で考える
  - 出力に関する項目は扱わない



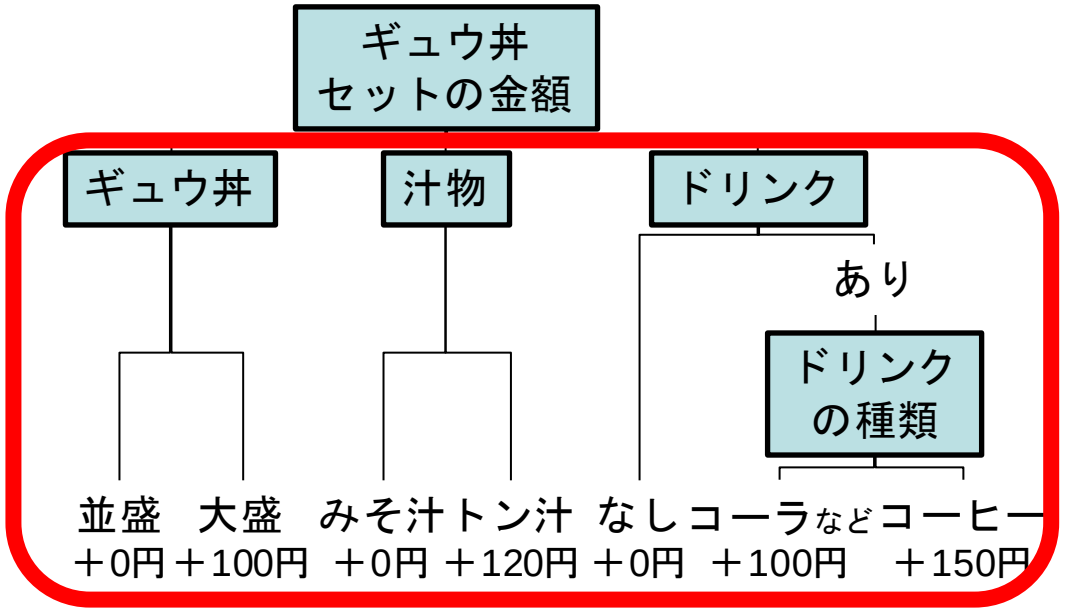
2. クラシフィケーションツリーを作る(1/3)

- CLNとクラスに分けてノードを書き出す
- **CLN**は「テストしたいこと」を書く
  - 例えば「汁物」「ドリンク」など
- クラスは「CLNをテストに使う要素として分割したもの」を書く
  - 例えば「みそ汁」「トン汁」
  - CLNに対して「不足なく」「重複なく」分割する
- テストの入力に関する項目で考える
  - 出力に関する項目は扱わない



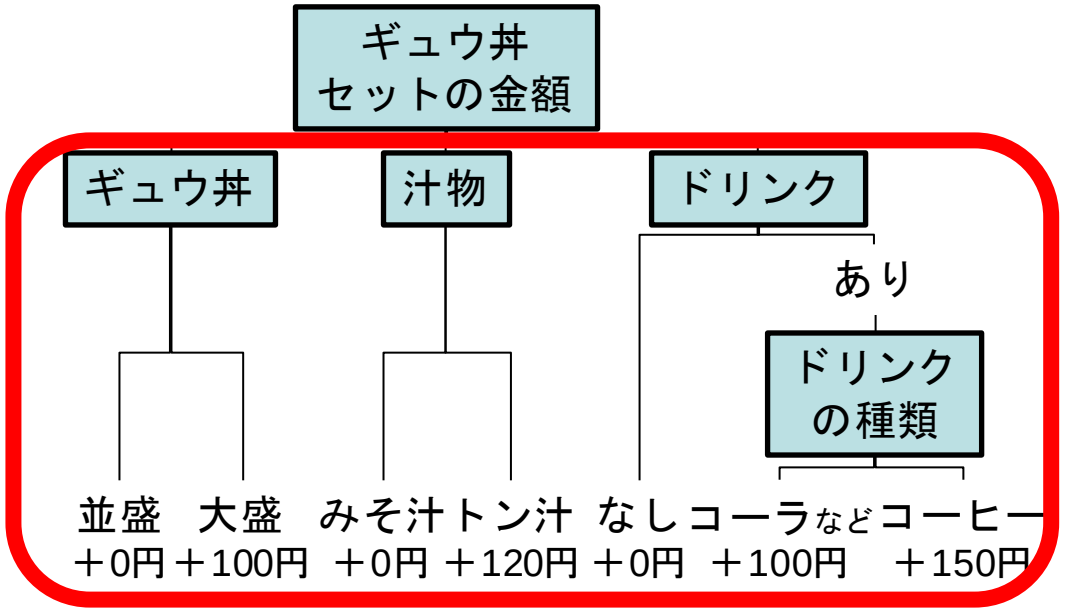
2. クラシフィケーションツリーを作る(1/3)

- CLNとクラスに分けてノードを書き出す
- CLNは「テストしたいこと」を書く
  - 例えば「汁物」「ドリンク」など
- **クラス**は「**CLN**をテストに使う要素として分割したもの」を書く
  - 例えば「みそ汁」「トン汁」
  - **CLN**に対して「不足なく」「重複なく」分割する
- テストの入力に関する項目で考える
  - 出力に関する項目は扱わない



2. クラシフィケーションツリーを作る(1/3)

- **CLN**と**クラス**に分けてノードを書き出す
- **CLN**は「テストしたいこと」を書く
  - 例えば「汁物」「ドリンク」など
- **クラス**は「**CLN**をテストに使う要素として分割したもの」を書く
  - 例えば「みそ汁」「トン汁」
  - **CLN**に対して「不足なく」「重複なく」分割する
- テストの入力に関する項目で考える
  - 出力に関する項目は扱わない



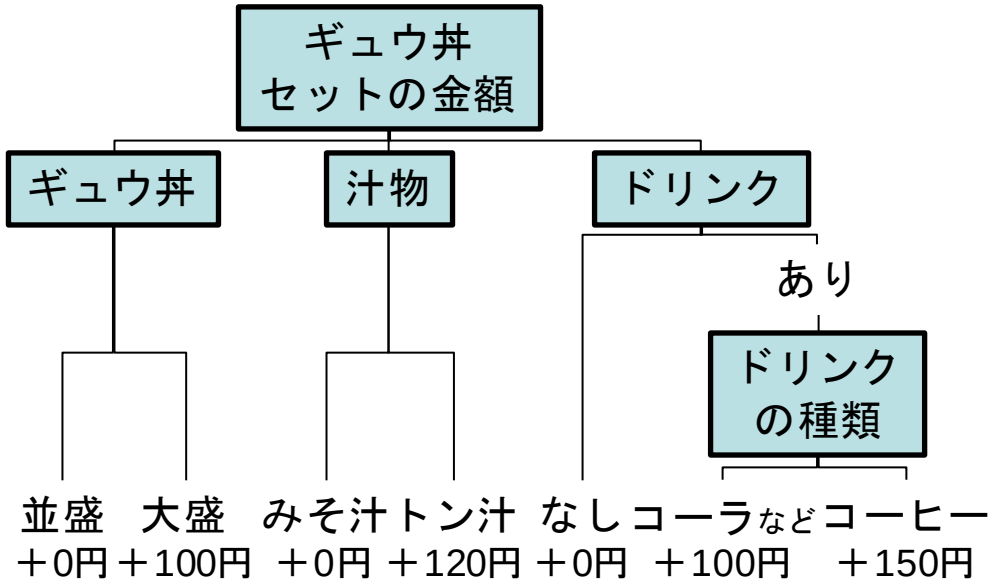
2. クラシフィケーションツリーを作る(2/3)

- **CLN**より下の構造は2パターン取り得る
  - **CLN**
  - **クラス**

  ※ **CLN**は必ず子ノードを持つ
- **クラス**より下の構造は2パターン取り得る
  - **CLN**
  - 子ノード無し

  ※ **クラス**の子ノードに**クラス**は来ない

  ※ あるノードの子ノードで**CLN**と**クラス**は混在しない
- ツリーの末端がすべて**クラス**になるまで検討する



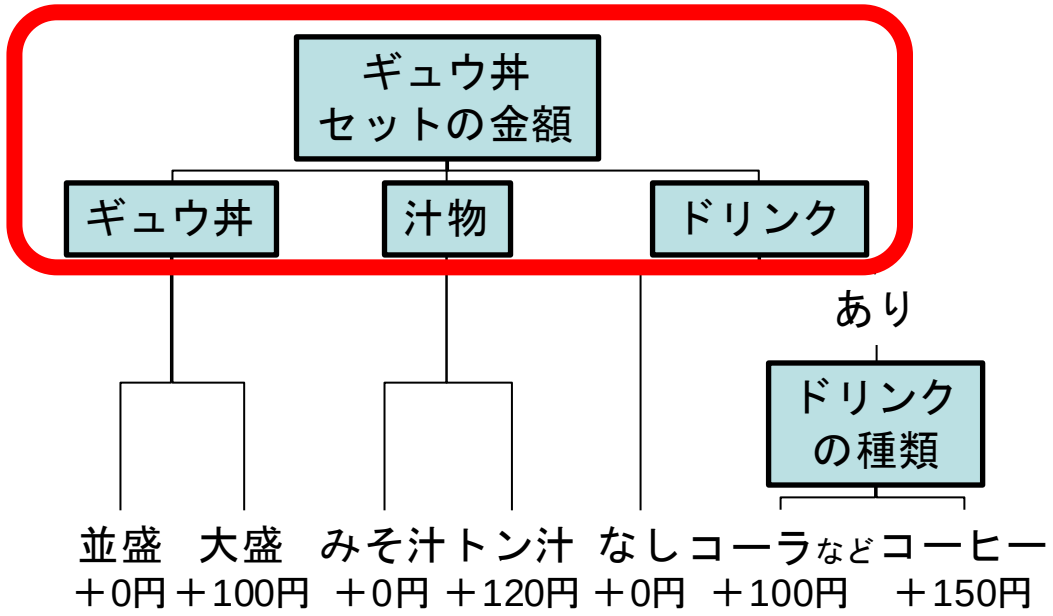
2. クラシフィケーションツリーを作る(2/3)

- **CLN**より下の構造は2パターン取り得る
  - **CLN**
  - クラス

  ※ **CLN**は必ず子ノードを持つ
- クラスより下の構造は2パターン取り得る
  - CLN
  - 子ノード無し

  ※ クラスの子ノードにクラスは来ない

  ※ あるノードの子ノードでCLNとクラスは混在しない
- ツリーの末端がすべてクラスになるまで検討する



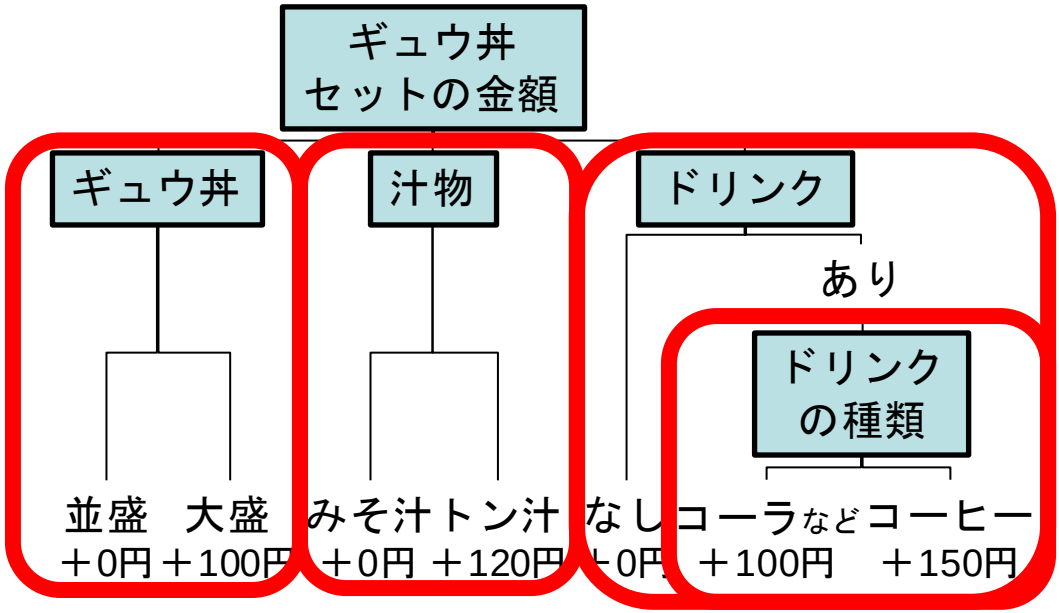
2. クラシフィケーションツリーを作る(2/3)

- **CLN**より下の構造は2パターン取り得る
  - CLN
  - **クラス**

  ※ **CLN**は必ず子ノードを持つ
- クラスより下の構造は2パターン取り得る
  - CLN
  - 子ノード無し

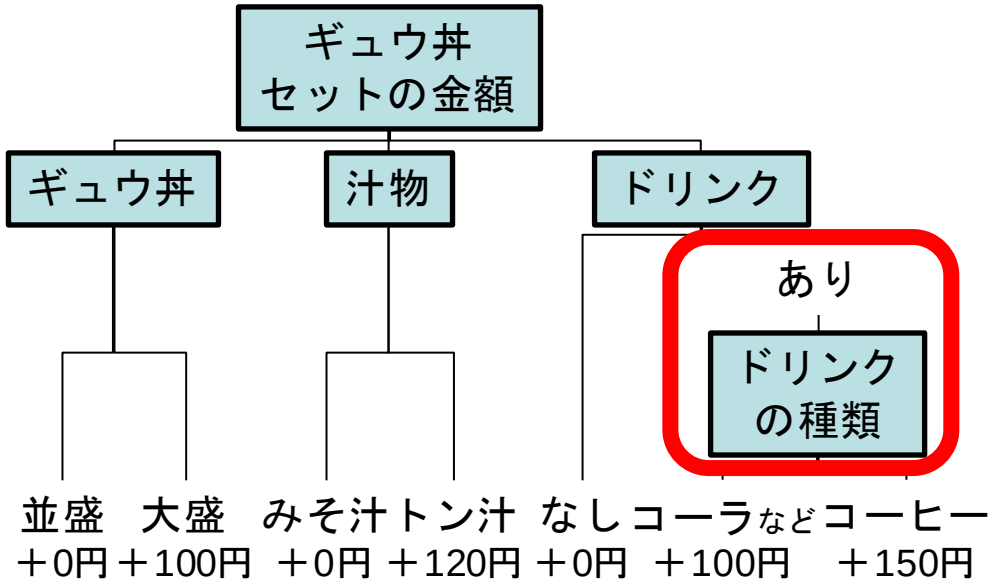
  ※ クラスの子ノードにクラスは来ない

  ※ あるノードの子ノードでCLNとクラスは混在しない
- ツリーの末端がすべてクラスになるまで検討する



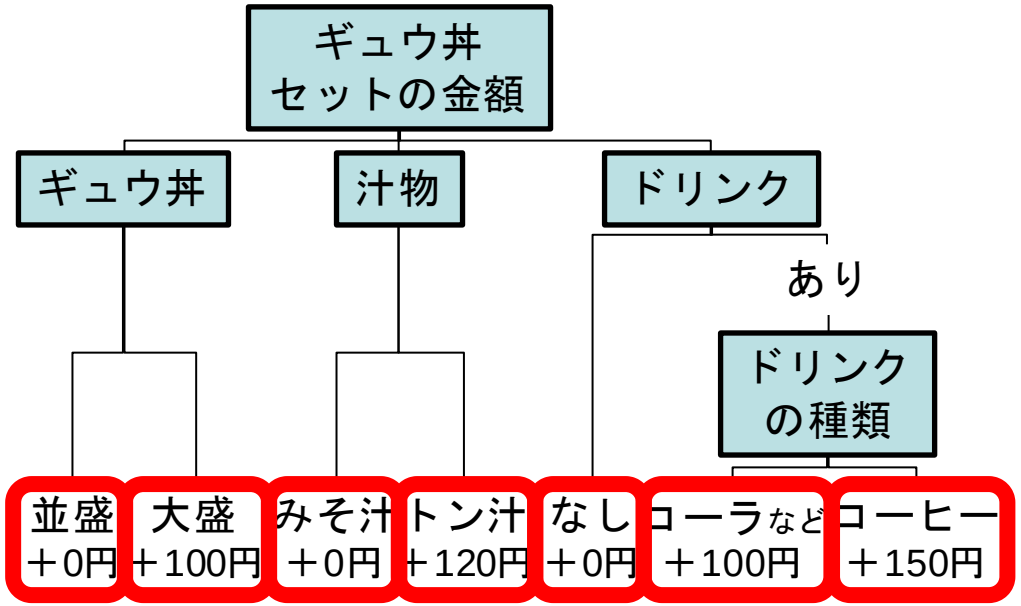
2. クラシフィケーションツリーを作る(2/3)

- CLNより下の構造は2パターン取り得る
  - CLN
  - クラス
  ※ CLNは必ず子ノードを持つ
- **クラス**より下の構造は2パターン取り得る
  - **CLN**
  - 子ノード無し
  ※ **クラス**の子ノードに**クラス**は来ない
  ※ あるノードの子ノードで**CLN**と**クラス**は混在しない
- ツリーの末端がすべてクラスになるまで検討する



2. クラシフィケーションツリーを作る(2/3)

- CLNより下の構造は2パターン取り得る
  - CLN
  - クラス
  ※ CLNは必ず子ノードを持つ
- **クラス**より下の構造は2パターン取り得る
  - CLN
  - 子ノード無し
  ※ **クラス**の子ノードに**クラス**は来ない
  ※ あるノードの子ノードで**CLN**と**クラス**は混在しない
- ツリーの末端がすべて**クラス**になるまで検討する



2. クラシフィケーションツリーを作る(2/3)

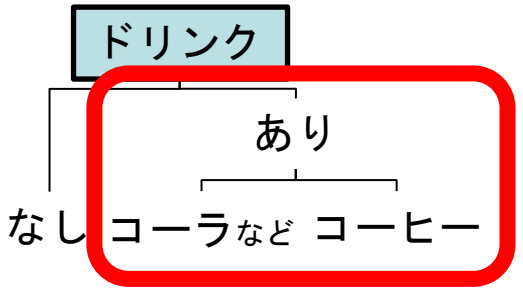
- CLNより下の構造は2パターン取り得る
  - CLN
  - クラス

  ※ CLNは必ず子ノードを持つ
- クラスより下の構造は2パターン取り得る
  - CLN
  - 子ノード無し

  ※ **クラス**の子ノードに**クラス**は来ない

  ※ あるノードの子ノードでCLNとクラスは混在しない
- ツリーの末端がすべてクラスになるまで検討する

× NG



2. クラシフィケーションツリーを作る(2/3)

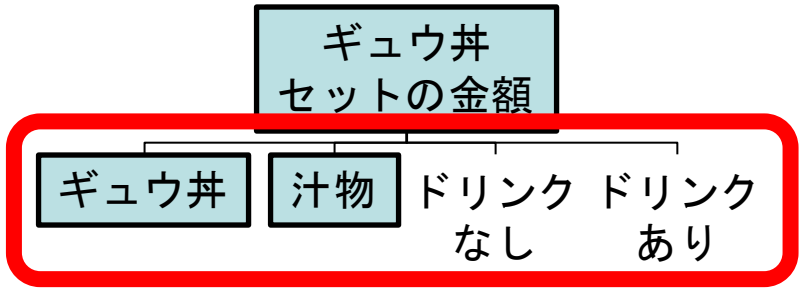
- CLNより下の構造は2パターン取り得る
  - CLN
  - クラス

  ※ CLNは必ず子ノードを持つ
- クラスより下の構造は2パターン取り得る
  - CLN
  - 子ノード無し

  ※ クラスの子ノードにクラスは来ない

  ※ あるノードの子ノードで**CLN**と**クラス**は混在しない
- ツリーの末端がすべてクラスになるまで検討する

× NG



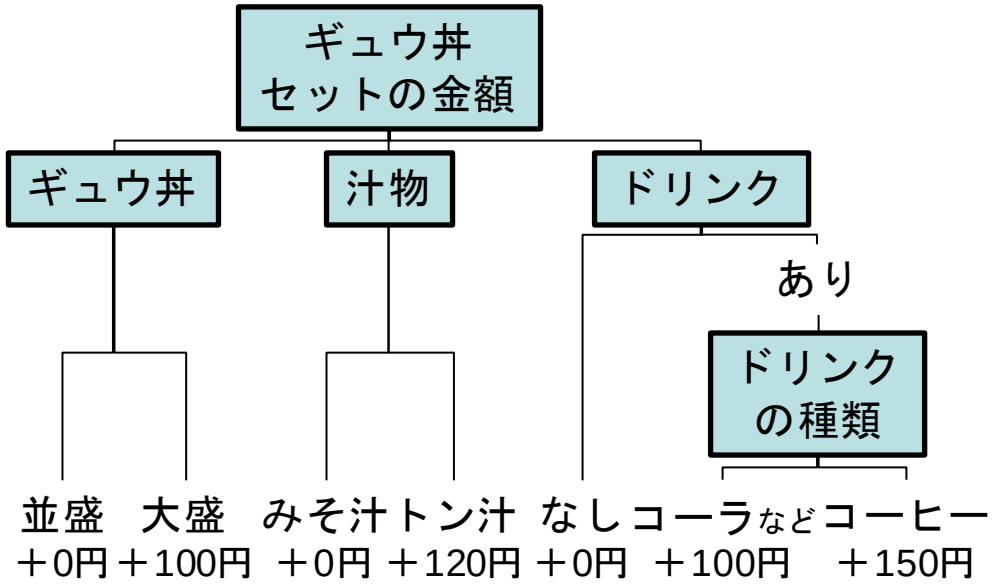
2. クラシフィケーションツリーを作る(2/3)

- **CLN**より下の構造は2パターン取り得る
  - **CLN**
  - **クラス**

  ※ **CLN**は必ず子ノードを持つ
- **クラス**より下の構造は2パターン取り得る
  - **CLN**
  - 子ノード無し

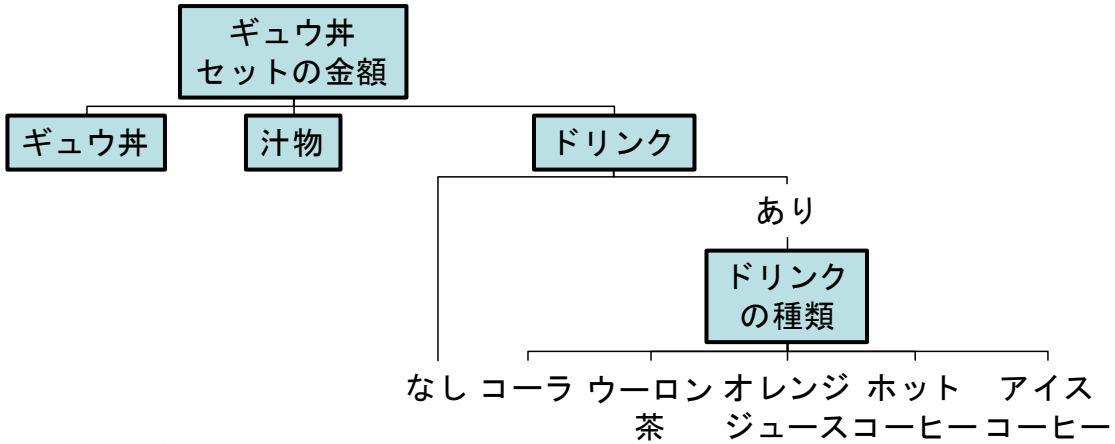
  ※ **クラス**の子ノードに**クラス**は来ない

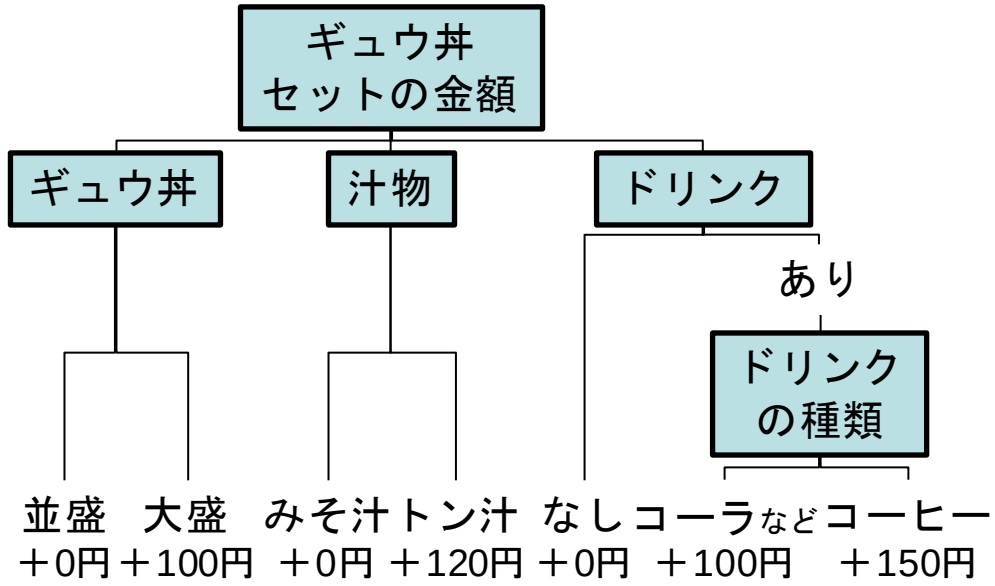
  ※ あるノードの子ノードで**CLN**と**クラス**は混在しない
- ツリーの末端がすべて**クラス**になるまで検討する



2. クラシフィケーションツリーを作る(3/3)

- **クラス**を細かくすると、テストケース数が大きくなりやすい

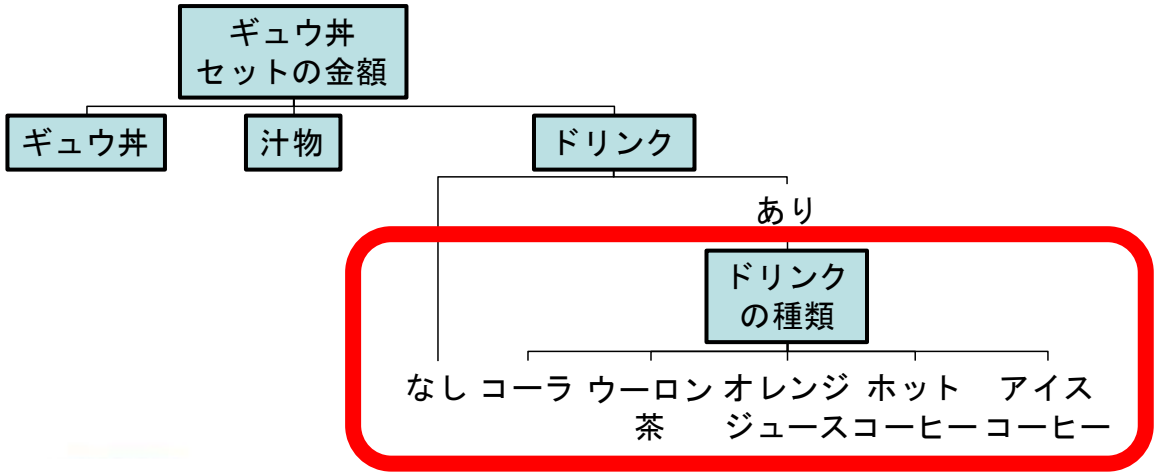


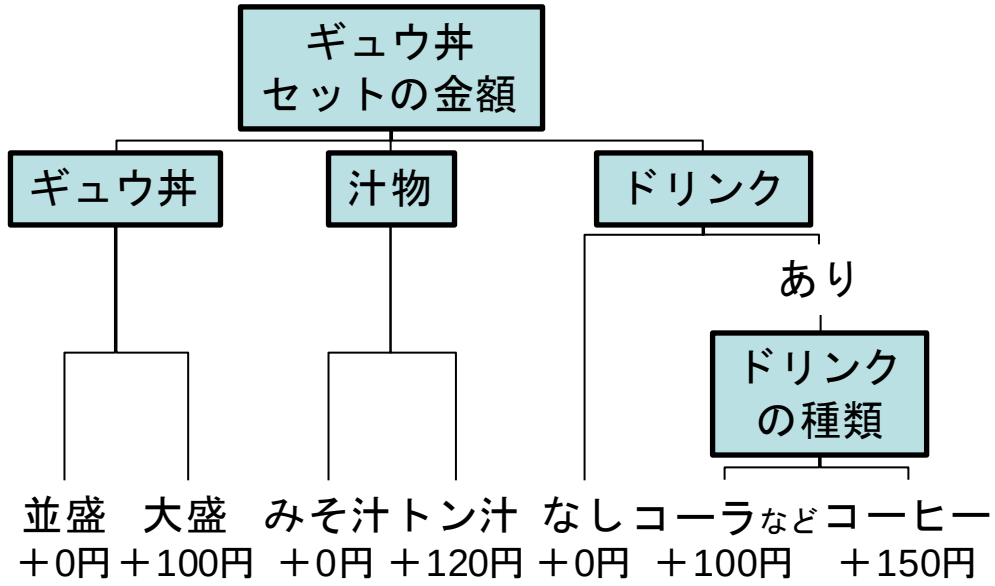


- テストしたいことに合わせて、同値分割法で同じように処理されることを期待する値を１つの**クラス**にまとめる
  - 効率的にテストのために、適切な粒度のCTを目指す
  - 例：「ギュウ丼セットの金額」のテストのため、ドリンク価格が同じものをまとめる

2. クラシフィケーションツリーを作る(3/3)

- **クラス**を細かくすると、テストケース数が大きくなりやすい

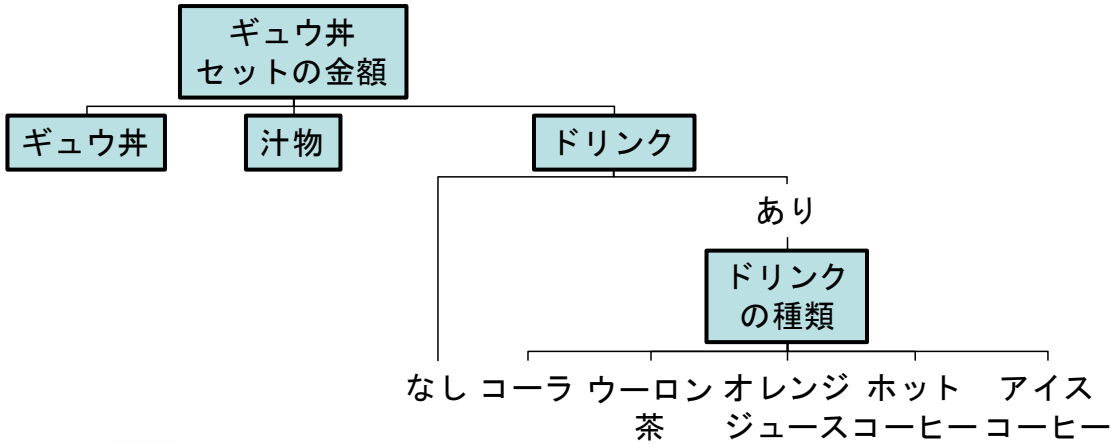


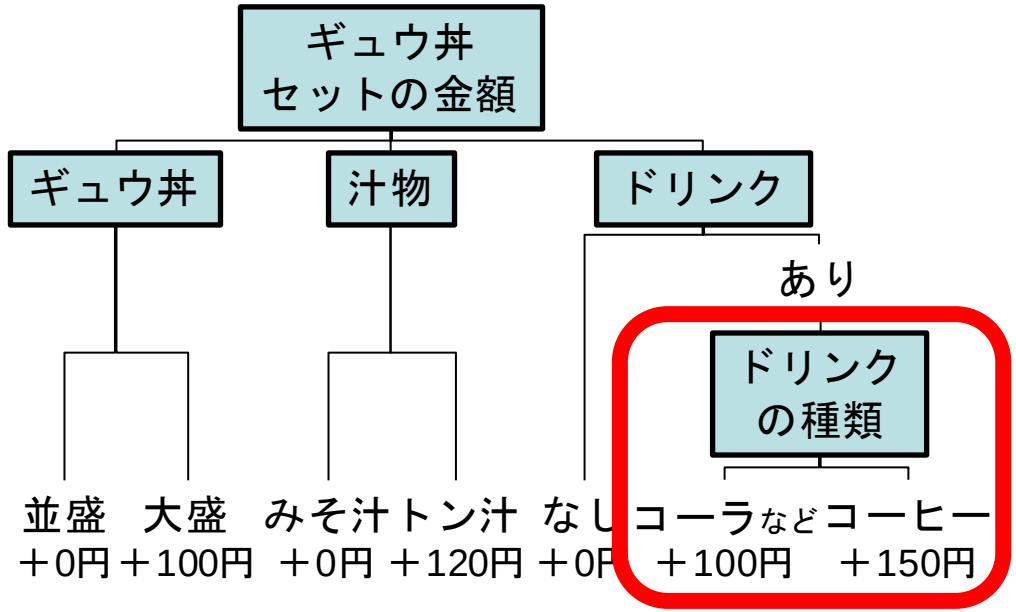


- テストしたいことに合わせて、同値分割法で同じように処理されることを期待する値を１つのクラスにまとめる
  - 効率的にテストのために、適切な粒度のCTを目指す
  - 例：「ギュウ丼セットの金額」のテストのため、ドリンク価格が同じものをまとめる

2. クラシフィケーションツリーを作る(3/3)

- クラスを細かくすると、テストケース数が大きくなりやすい

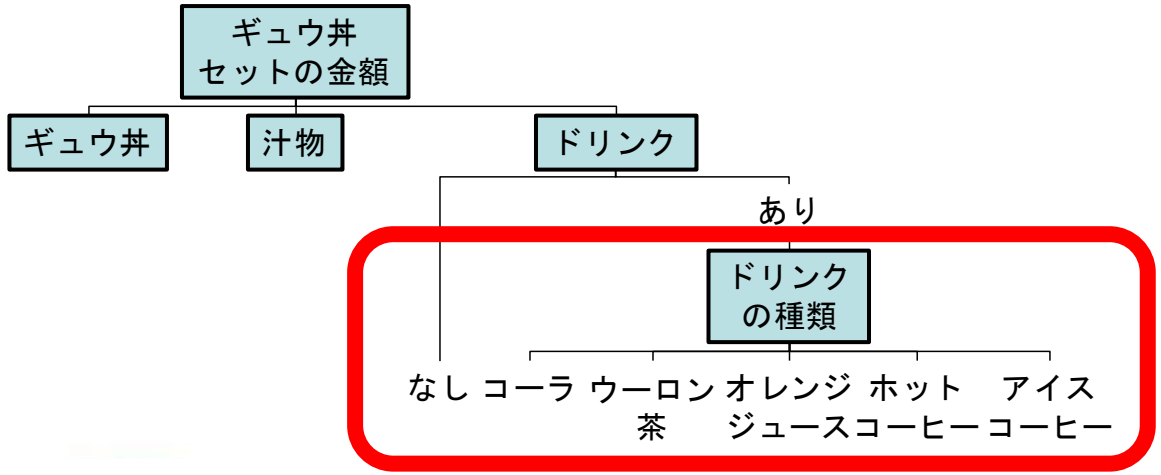


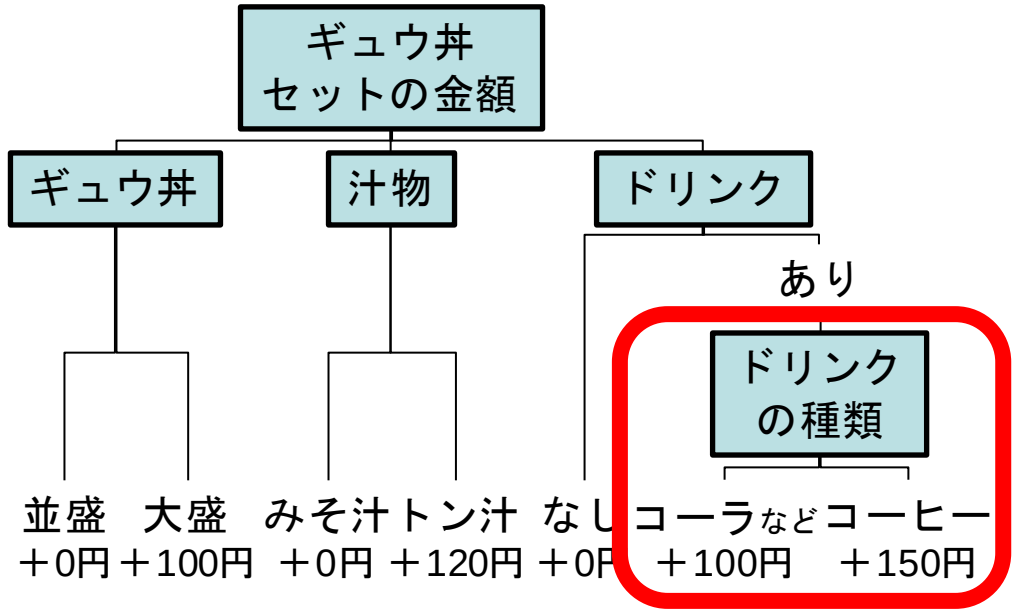


- テストしたいことに合わせて、同値分割法で同じように処理されることを期待する値を１つの**クラス**にまとめる
  - 効率的にテストのために、適切な粒度のCTを目指す
  - 例：「ギュウ丼セットの金額」のテストのため、ドリンク価格が同じものをまとめる

2. クラシフィケーションツリーを作る(3/3)

- **クラス**を細かくすると、テストケース数が大きくなりやすい





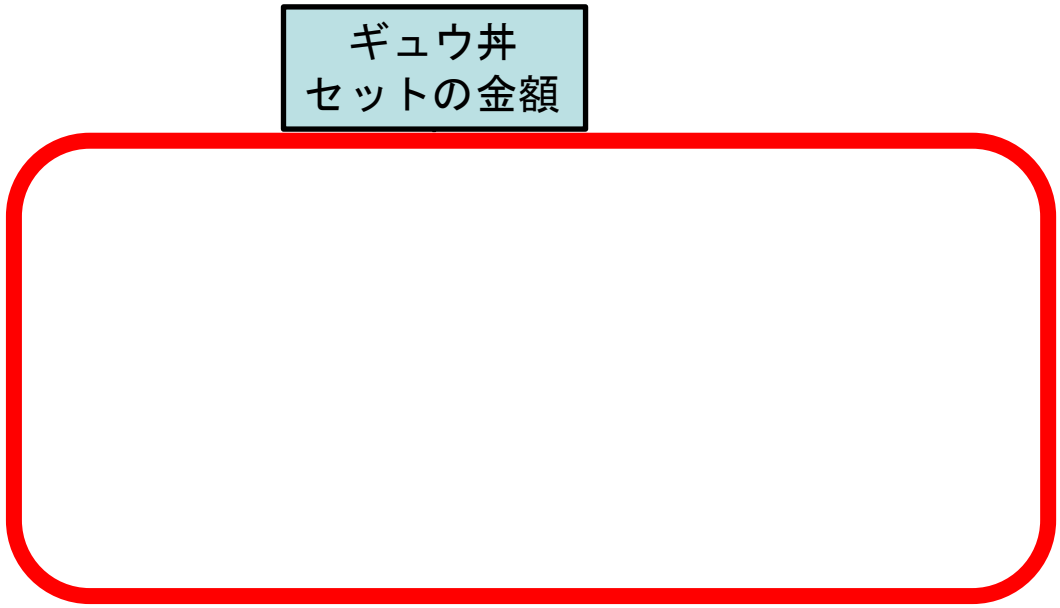
- テストしたいことに合わせて、同値分割法で同じように処理されることを期待する値を１つの**クラス**にまとめる
  - 効率的にテストのために、適切な粒度のCTを目指す
  - 例：「ギュウ丼セットの金額」のテストのため、ドリンク価格が同じものをまとめる

【ワーク】2. クラシフィケーションツリーを作る

CLNや**クラス**の項目ごとに付箋に１つずつ
書き出して、テスト対象の下に並べよう

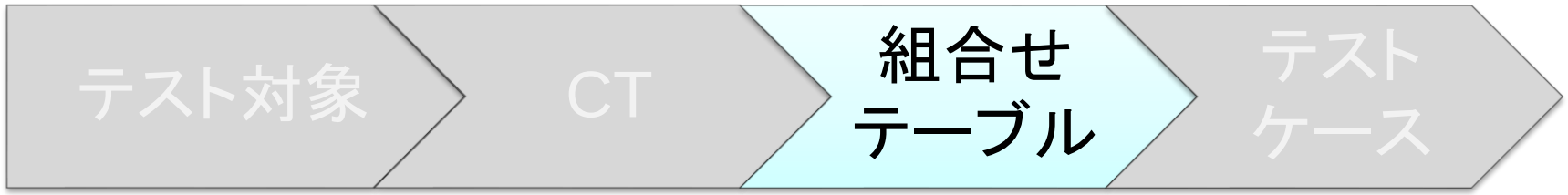
- **CLN**は**緑色の付箋**に書こう
- **クラス**は**黄色の付箋**に書こう

構造を整理してクラシフィケーションツリーを
完成させよう



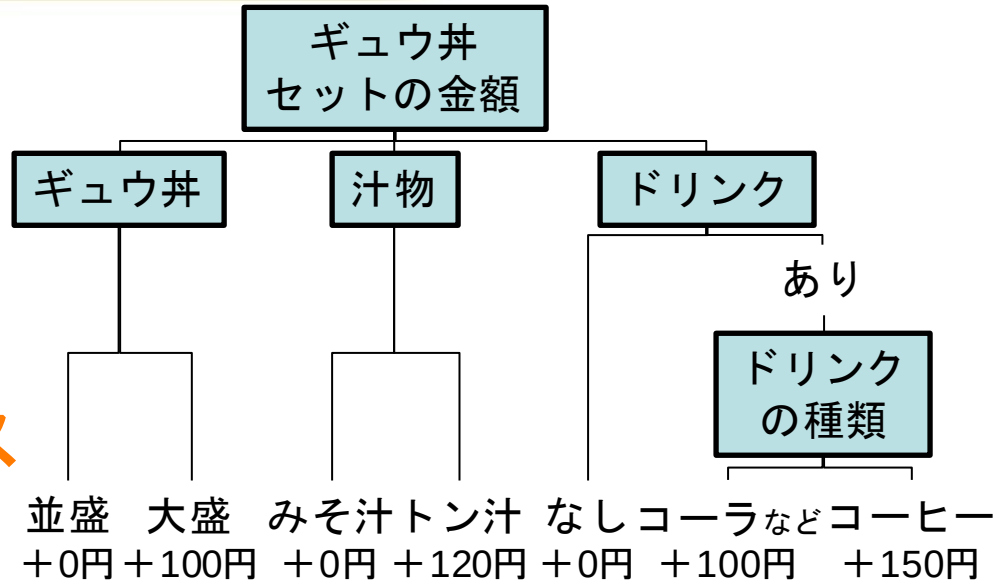
CT技法によるテストケースの作り方

1. テスト対象を選択する
2. クラシフィケーションツリーを作成する
3. 組合せテーブルを作成する
4. テストデータ、期待結果を定義する



3. 組合せテーブルを作る(1/3)

- 作成したクラシフィケーションツリーの末端にある**クラス**ごとに縦線を引く
- 交差するように横線を引く
  - 横線は1つのテストケースに相当
  - 交点の丸はテストケースで使用する**クラス**
- 必要なテスト基準に満たすまで横線（テストケース）を用意する
  - **CLN**、**クラス**の下の**CLN**は、それぞれからひとつが選ばれる
  - **CLN**の下の**クラス**はどれかひとつが選ばれる

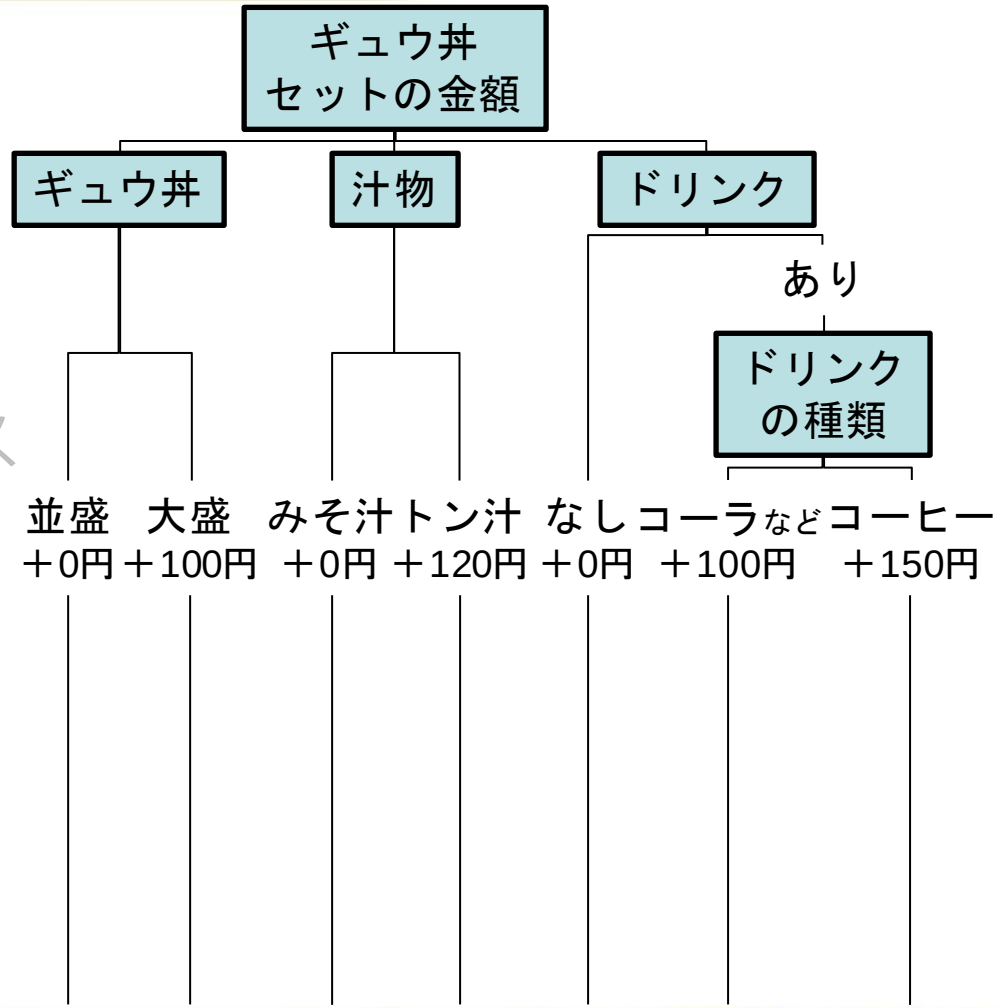


例の網羅基準
:全組合せ

3. 組合せテーブルを作る(1/3)

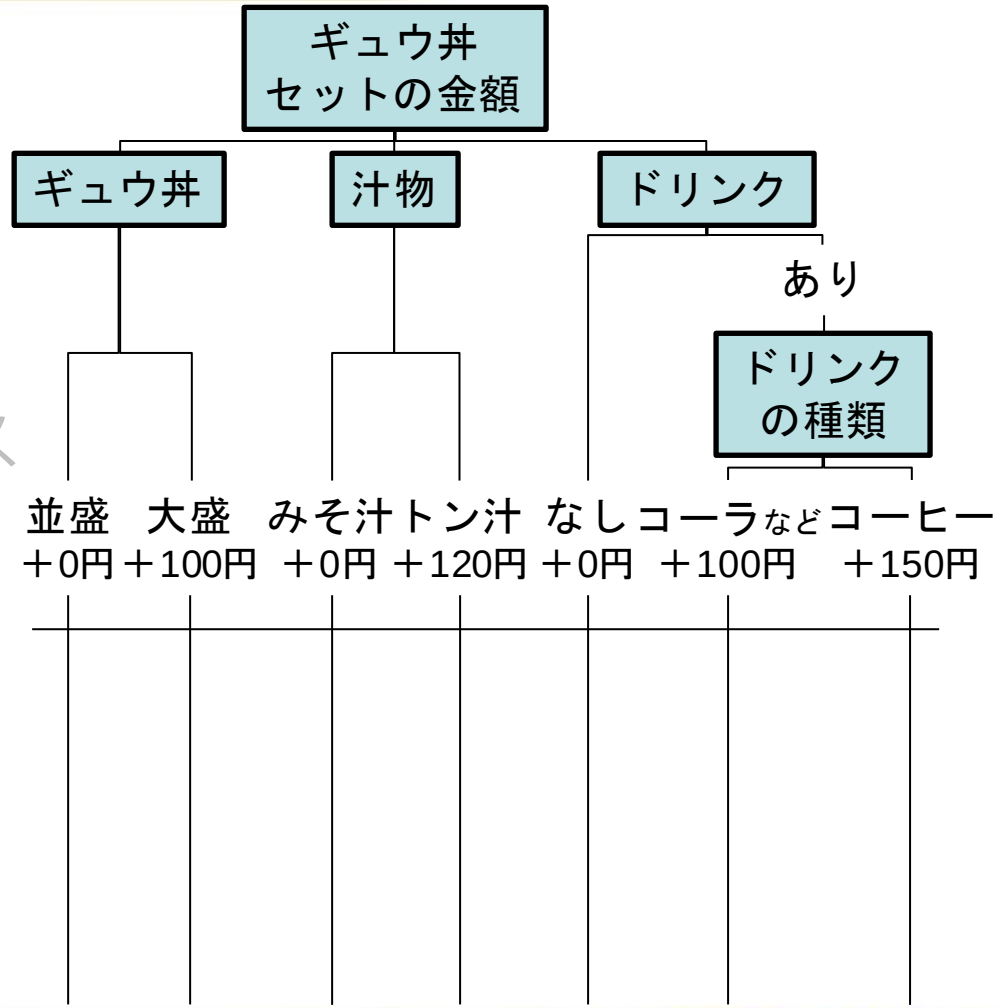
- 作成したクラシフィケーションツリーの末端にある**クラス**ごとに縦線を引く
- 交差するように横線を引く
  - 横線は1つのテストケースに相当
  - 交点の丸はテストケースで使用するクラス
- 必要なテスト基準に満たすまで横線（テストケース）を用意する
  - CLN、クラスの下のCLNは、それぞれからひとつが選ばれる
  - CLNの下のクラスはどれかひとつが選ばれる

例の網羅基準
:全組合せ



3. 組合せテーブルを作る(1/3)

- 作成したクラシフィケーションツリーの末端にあるクラスごとに縦線を引く
- 交差するように横線を引く
  - 横線は1つのテストケースに相当
  - 交点の丸はテストケースで使用するクラス
- 必要なテスト基準に満たすまで横線（テストケース）を用意する
  - CLN、クラスの下のCLNは、それぞれからひとつが選ばれる
  - CLNの下のクラスはどれかひとつが選ばれる



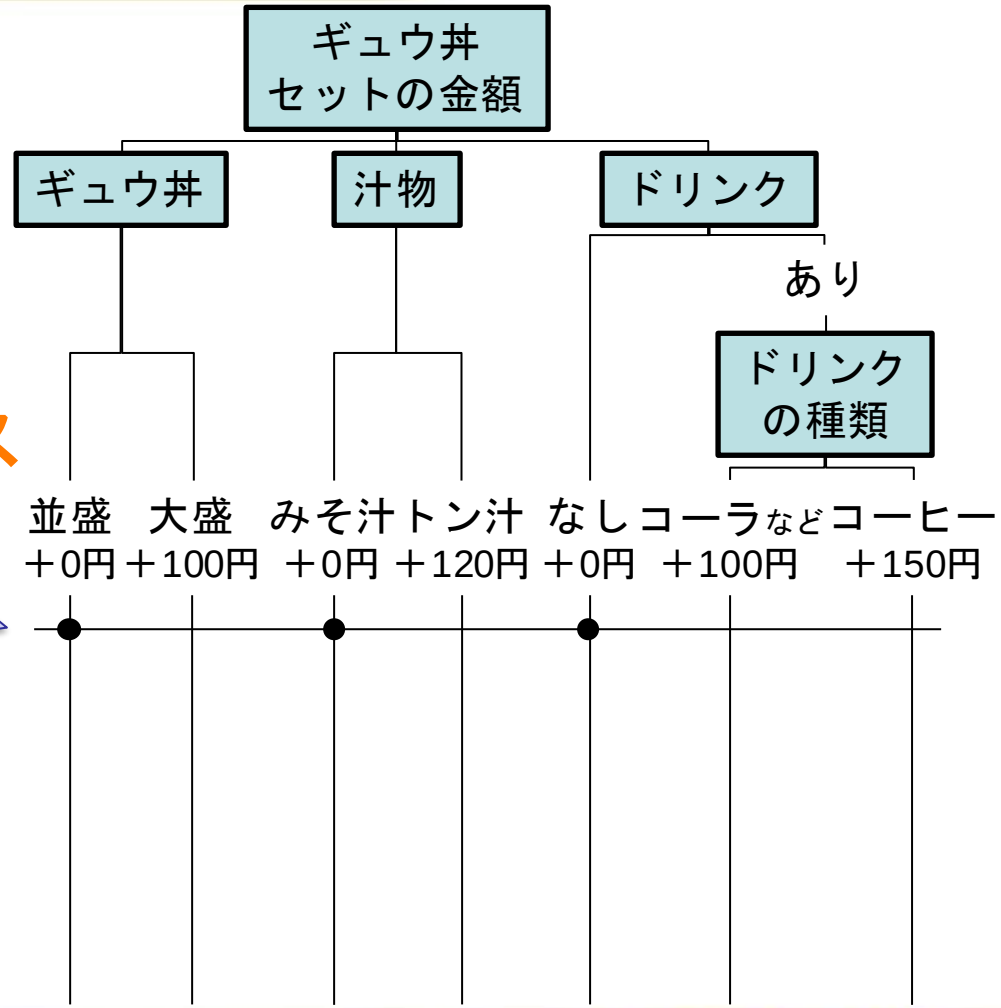
例の網羅基準
:全組合せ

3. 組合せテーブルを作る(1/3)

- 作成したクラシフィケーションツリーの末端にあるクラスごとに縦線を引く
- 交差するように横線を引く
  - 横線は1つのテストケースに相当
  - 交点の丸はテストケースで使用するクラス
- 必要なテ
  横線（テ
  - CLN
    それ
  - CLNの下のクラスは
    どれかひとつが選ばれる

右例は以下のテストケースを表す
ギュウ丼 :並盛
汁物 :みそ汁
ドリンク :なし

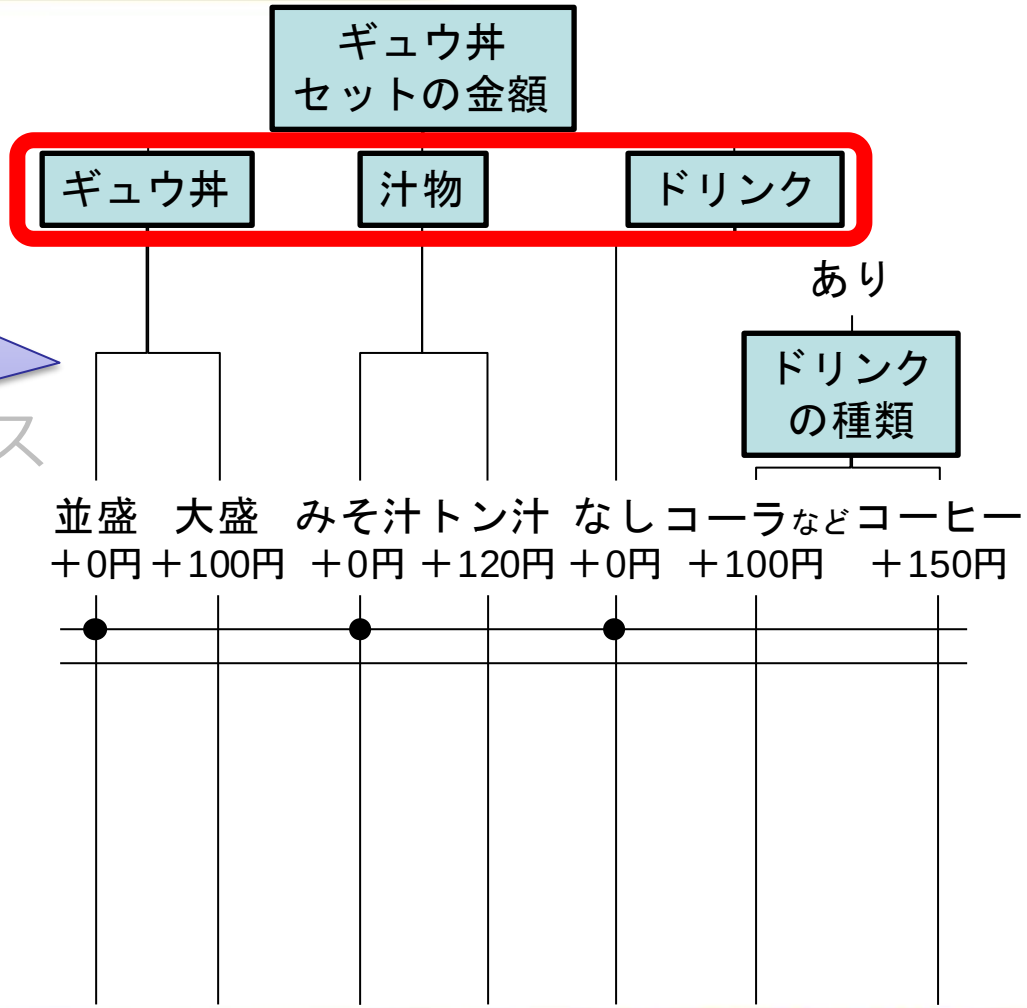
例の網羅基準
:全組合せ



3. 組合せテーブルを作る(1/3)

- 作成したクラシフィケーションツリーの末端にあるクラスごとに縦線を引く
- 交差する
  - 横線は
  - 交点の丸はテストケースで使用するクラス
- 必要なテスト基準に満たすまで横線（テストケース）を用意する
  - **CLN**、**クラス**の下の**CLN**は、それぞれからひとつが選ばれる
  - CLNの下のクラスはどれかひとつが選ばれる

テストケースで使用するデータを「ギュウ丼」「汁物」「ドリンク」からそれぞれ1つずつ選ぶ

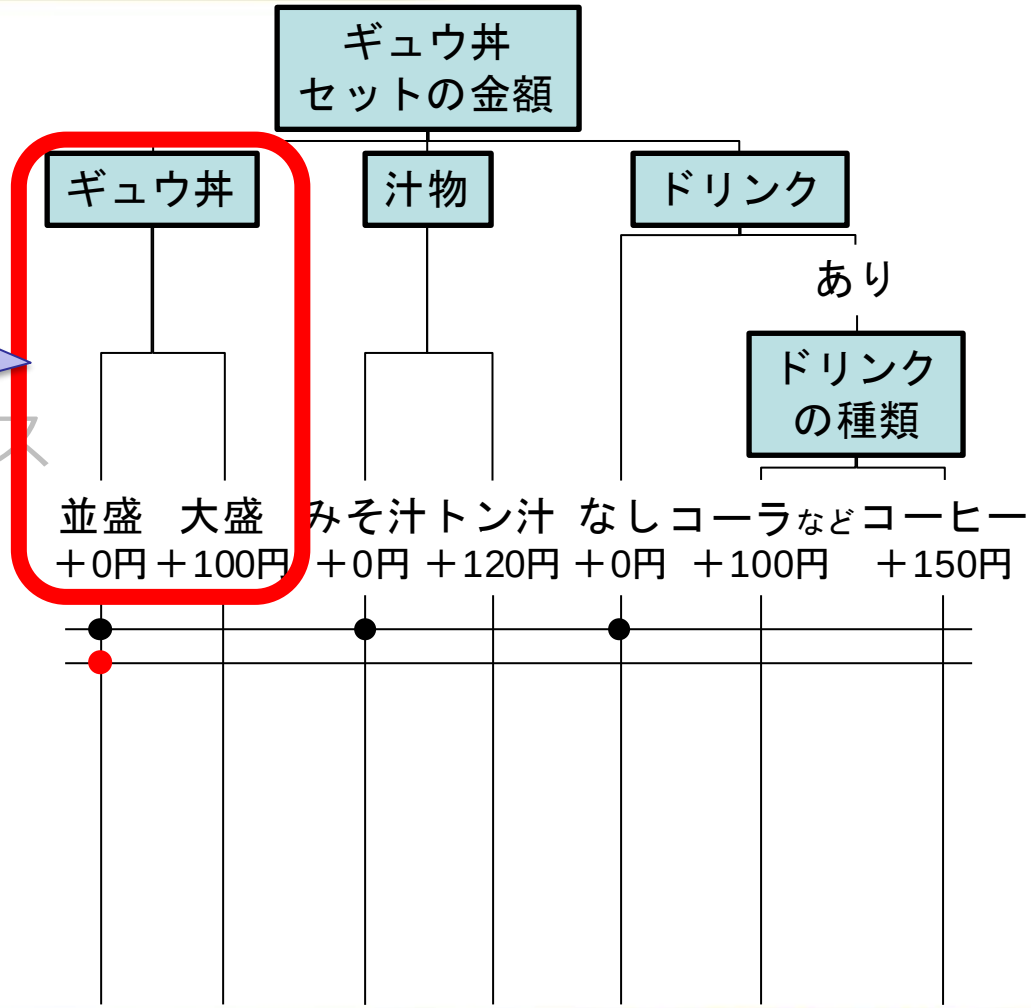


例の網羅基準
:全組合せ

3. 組合せテーブルを作る(1/3)

- 作成したクラシフィケーションツリーの末端にあるクラスごとに縦線を引く
- 交差する
  - 横線は
  - 交点の丸はテストケースで使用するクラス
- 必要なテスト基準に満たすまで横線（テストケース）を用意する
  - CLN、クラスの下のCLNは、それぞれからひとつが選ばれる
  - **CLN**の下の**クラス**はどれかひとつが選ばれる

「ギュウ丼」の「並盛」「大盛」からどれか1つを選ぶ
　ここでは「並盛」を選ぶ

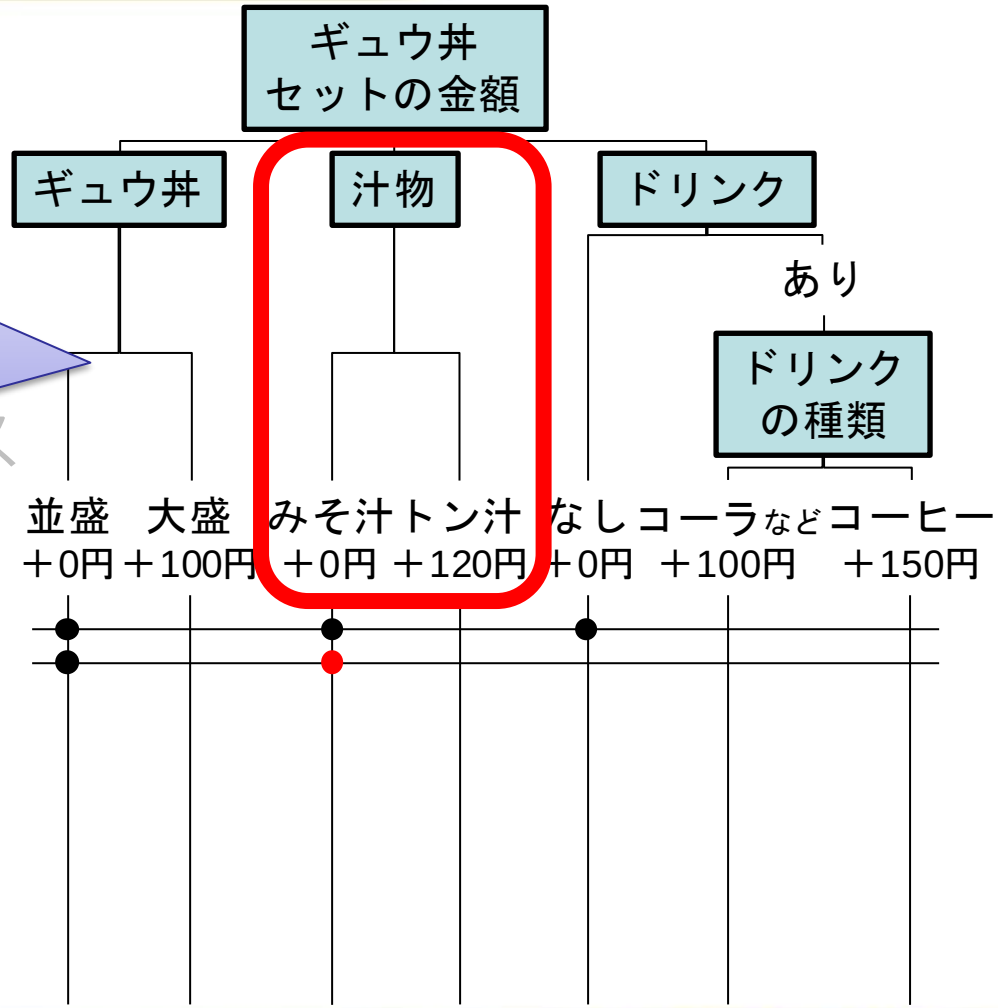


例の網羅基準
:全組合せ

3. 組合せテーブルを作る(1/3)

- 作成したクラシフィケーションツリーの末端にあるクラスごとに縦線を引く
- 交差するよう
  - 横線は1つ
  - 交点の丸はテストケースで使用するクラス
- 必要なテスト基準に満たすまで横線（テストケース）を用意する
  - CLN、クラスの下のCLNは、それぞれからひとつが選ばれる
  - **CLN**の下の**クラス**はどれかひとつが選ばれる

「汁物」の「みそ汁」「トン汁」からどれか1つを選ぶ
ここでは「みそ汁」を選ぶ

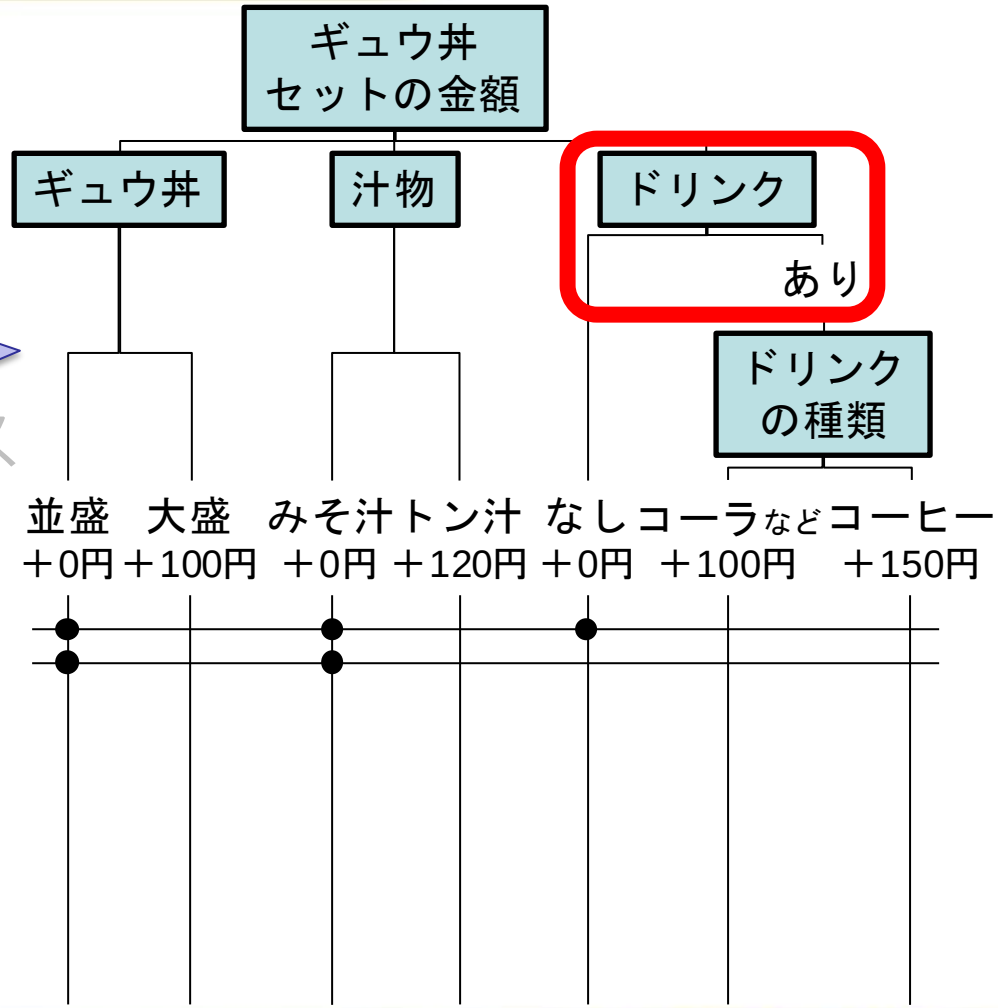


例の網羅基準
:全組合せ

3. 組合せテーブルを作る(1/3)

- 作成したクラシフィケーションツリーの末端にあるクラスごとに縦線を引く
- 交差するよう
  - 横線は1つ
  - 交点の丸はテストケースで使用するクラス
- 必要なテスト基準に満たすまで横線（テストケース）を用意する
  - CLN、クラスの下のCLNは、それぞれからひとつが選ばれる
  - **CLN**の下の**クラス**はどれかひとつが選ばれる

「ドリンク」の「なし」「あり」からどれか1つを選ぶ
ここでは「あり」を選ぶ

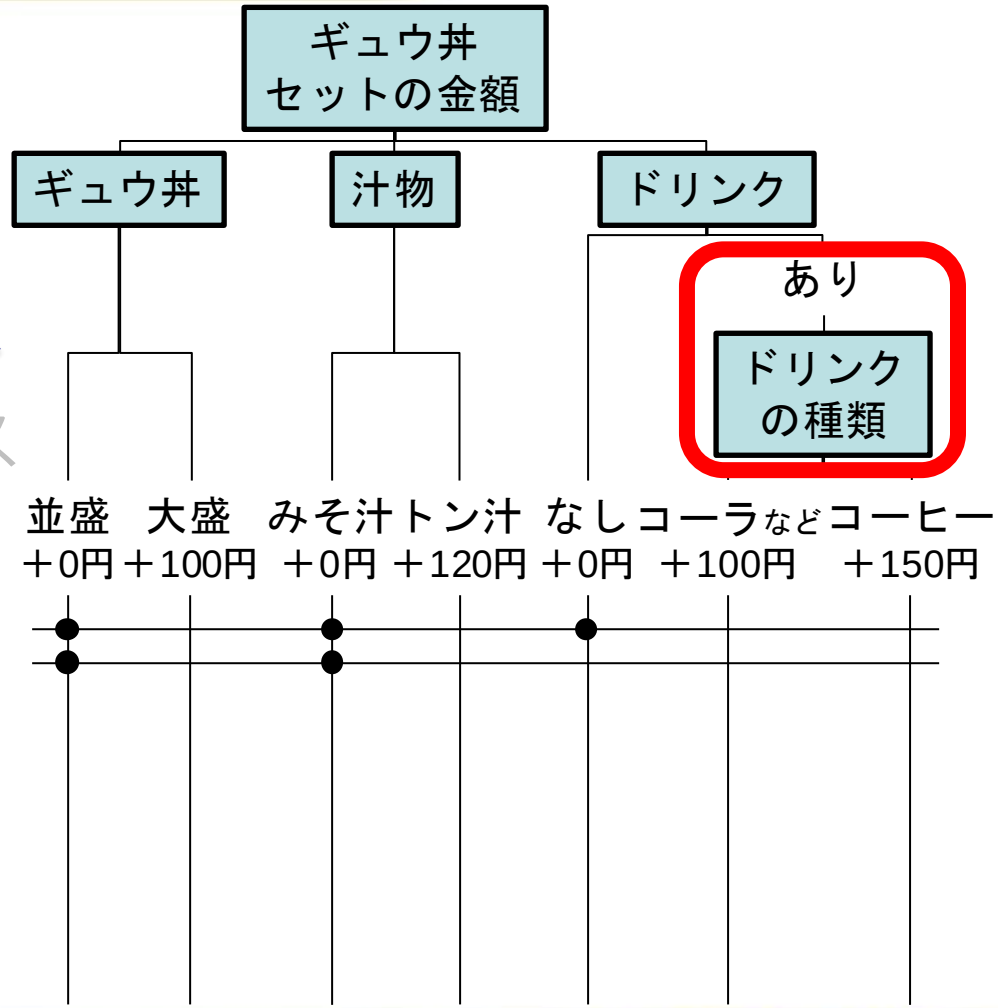


例の網羅基準
:全組合せ

3. 組合せテーブルを作る(1/3)

- 作成したクラシフィケーションツリーの末端にあるクラスごとに縦線を引く
- 交差する
  - 横線は
  - 交点の丸はテストケースで使用するクラス
- 必要なテスト基準に満たすまで横線（テストケース）を用意する
  - **CLN**、**クラス**の下の**CLN**は、それぞれからひとつが選ばれる
  - CLNの下のクラスはどれかひとつが選ばれる

テストケースで使用するデータを「ドリンクの種類」からそれぞれ1つずつ選ぶ

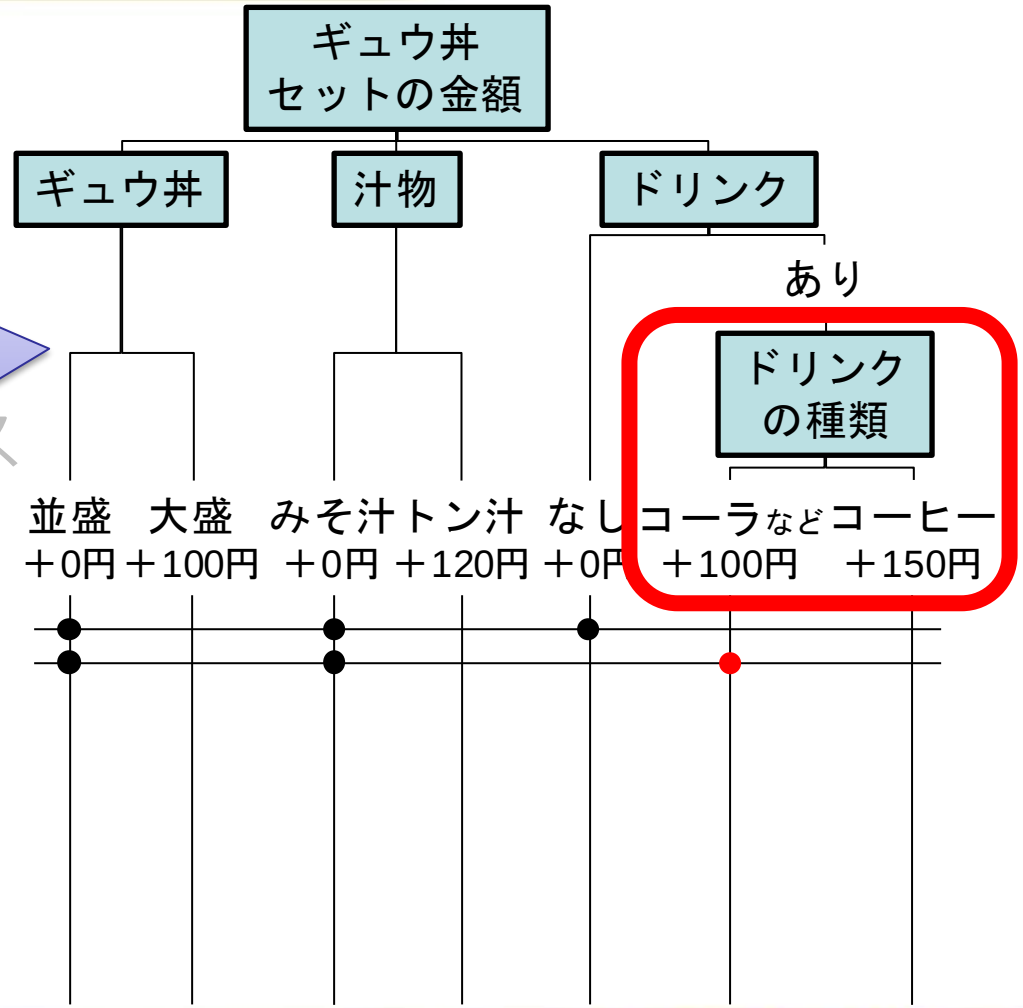


例の網羅基準
:全組合せ

3. 組合せテーブルを作る(1/3)

- 作成したクラシフィケーションツリーの末端にあるクラスごとに縦線を引く
  - 交点の丸はテストケースで使用するクラス
- 必要なテスト基準に満たすまで横線（テストケース）を用意する
  - CLN、クラスの下のCLNは、それぞれからひとつが選ばれる
  - **CLN**の下の**クラス**はどれかひとつが選ばれる

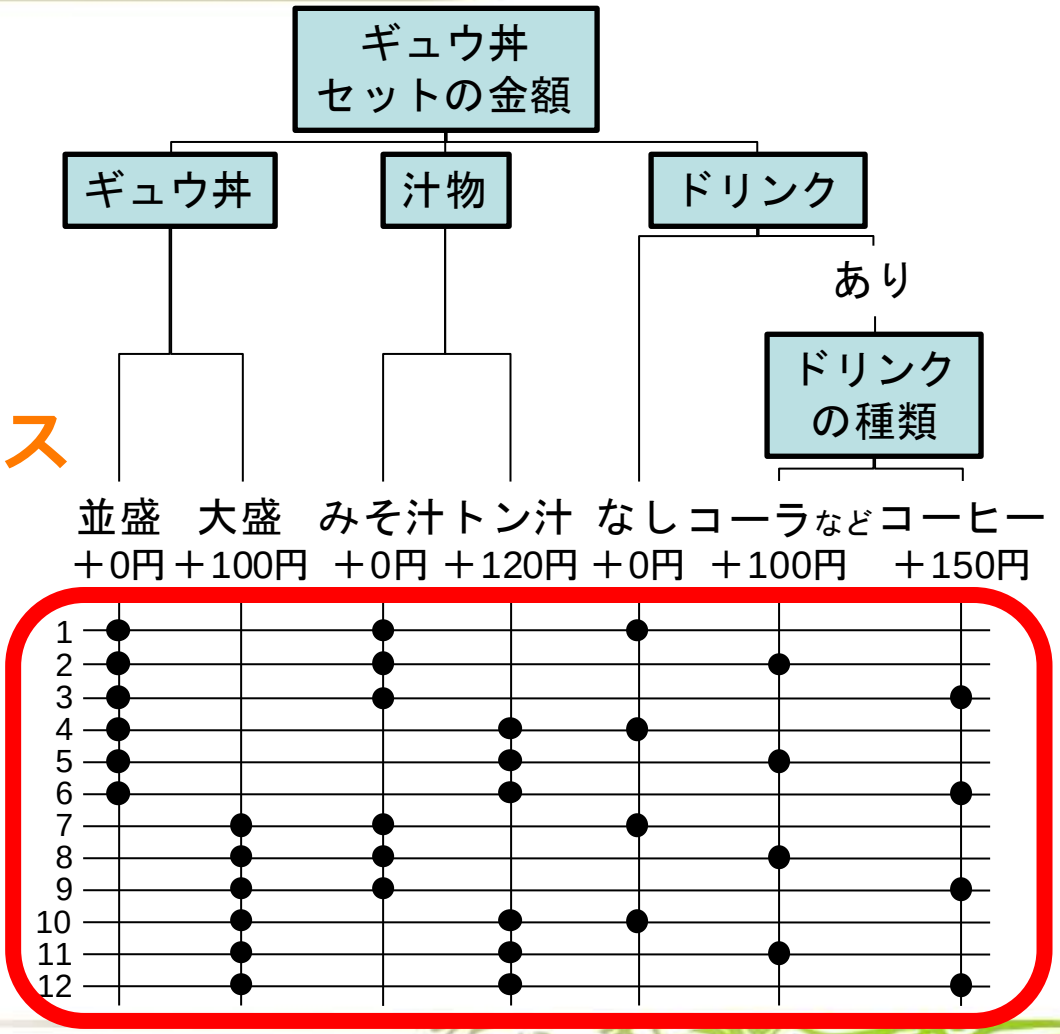
「ドリンクの種類」の「コーラなど」「コーヒー」からどれか1つを選ぶ
ここでは「コーラなど」を選ぶ



例の網羅基準
:全組合せ

3. 組合せテーブルを作る(1/3)

- 作成したクラシフィケーションツリーの末端にある**クラス**ごとに縦線を引く
- 交差するように横線を引く
  - 横線は1つのテストケースに相当
  - 交点の丸はテストケースで使用する**クラス**
- 必要なテスト基準に満たすまで横線（テストケース）を用意する
  - **CLN**、**クラス**の下の**CLN**は、それぞれからひとつが選ばれる
  - **CLN**の下の**クラス**はどれかひとつが選ばれる

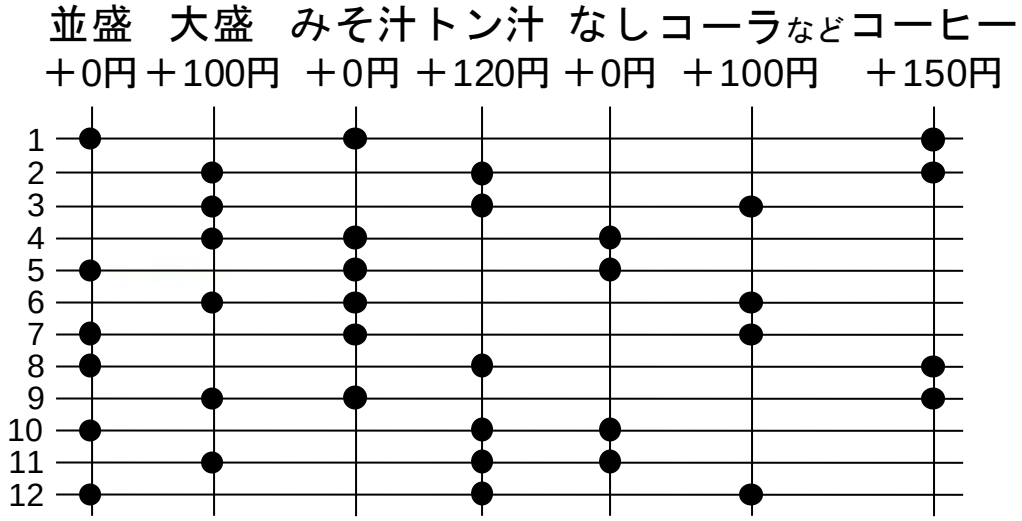


例の網羅基準
:全組合せ

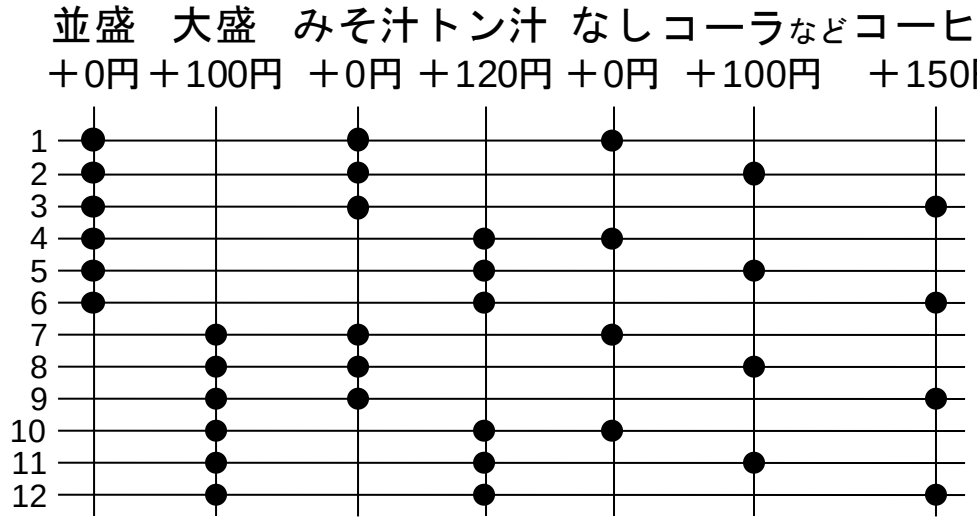
3. 組合せテーブルを作る(2/3)

全組合せのテストケースは規則性を持たせた方が抜けや重複に気づきやすい
網羅基準によって、並び方を工夫することで理解しやすくなる

規則性なし

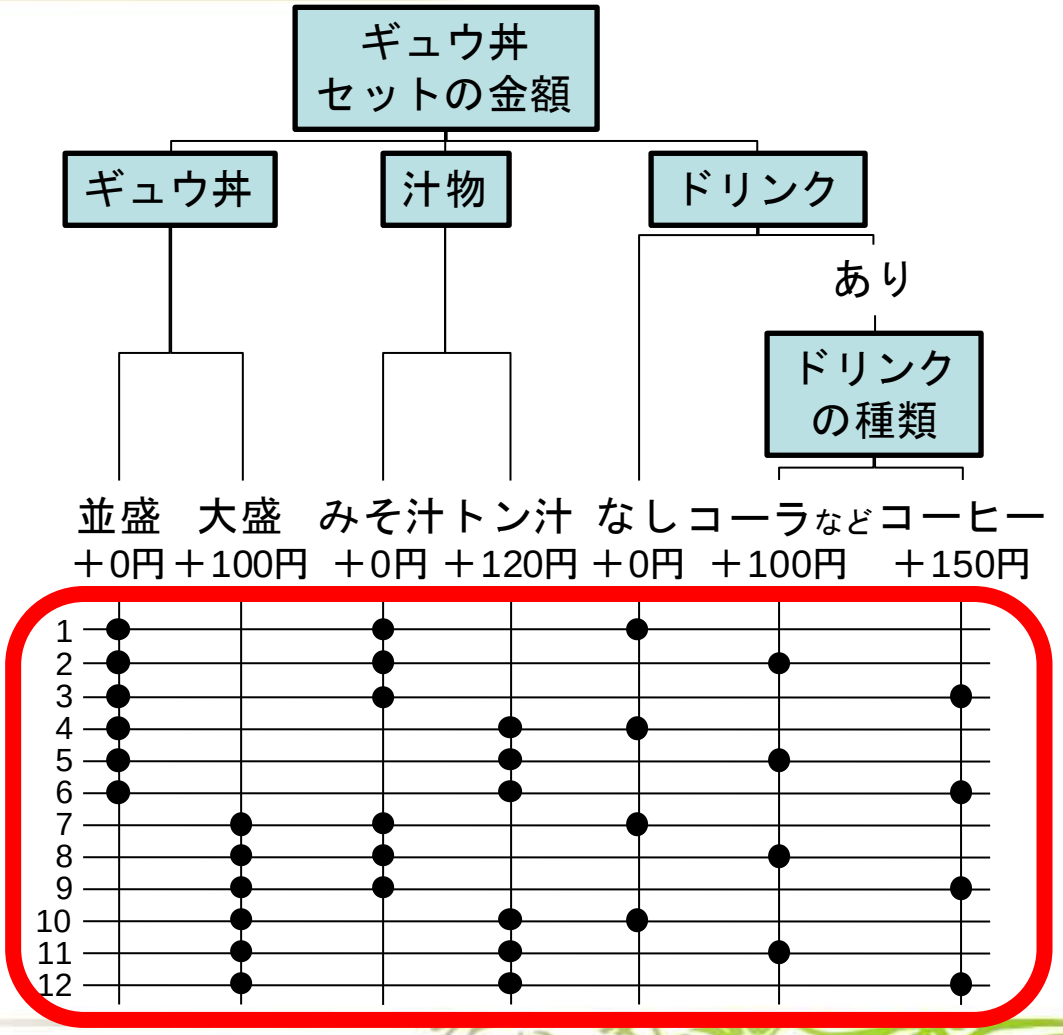


規則性あり



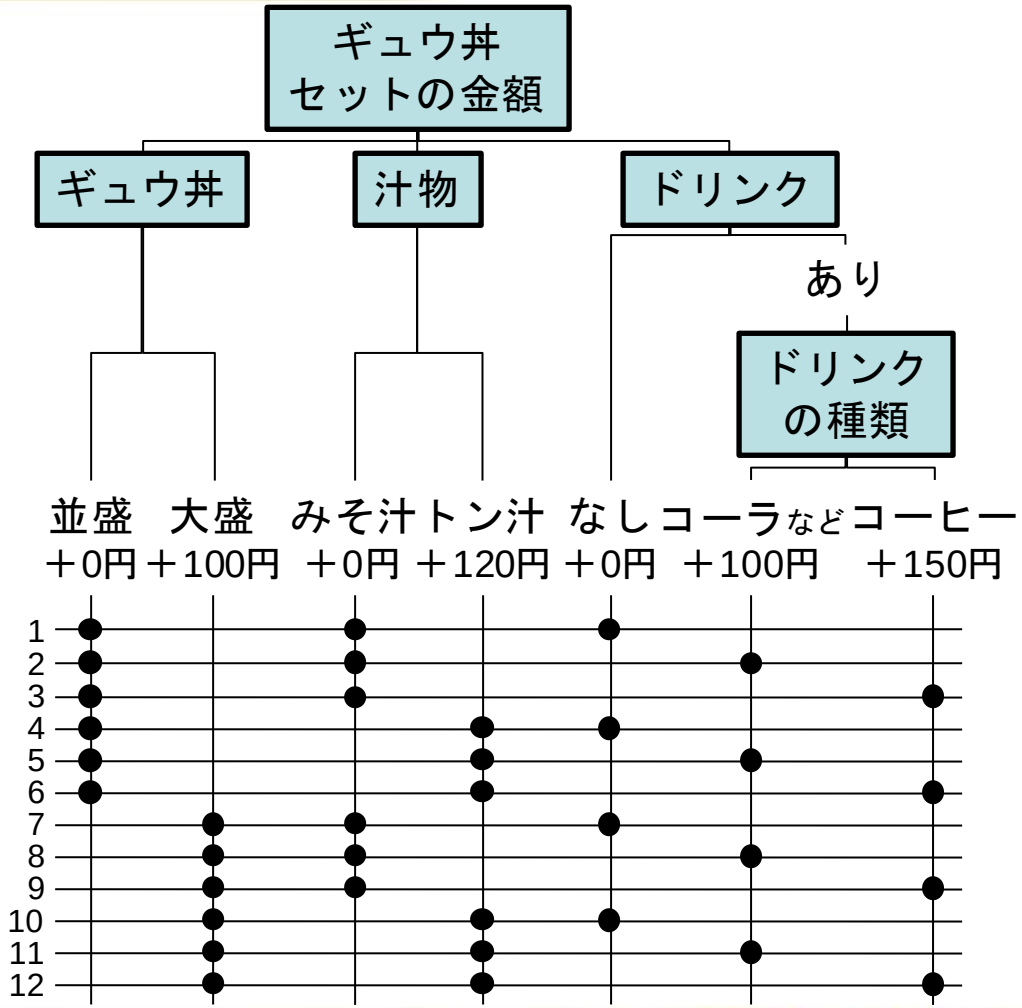
3. 組合せテーブルを作る(3/3)

- テストケースの組合せは
  保証したい範囲（機能や条件など）
  に応じて網羅基準を選定
  - Each Choice
    - 末端の**クラス**を少なくとも１回は使用
  - Pair-Wise
    - ２**CLN**ごとの組合せを網羅
  - All Combinations（全組合せ）
    - 全ての組合せを網羅
- CT技法では**CLN**ごとに網羅基準を
  選定しても良い（p.40参照）



All Combinations(12ケース)

全ての組合せを網羅

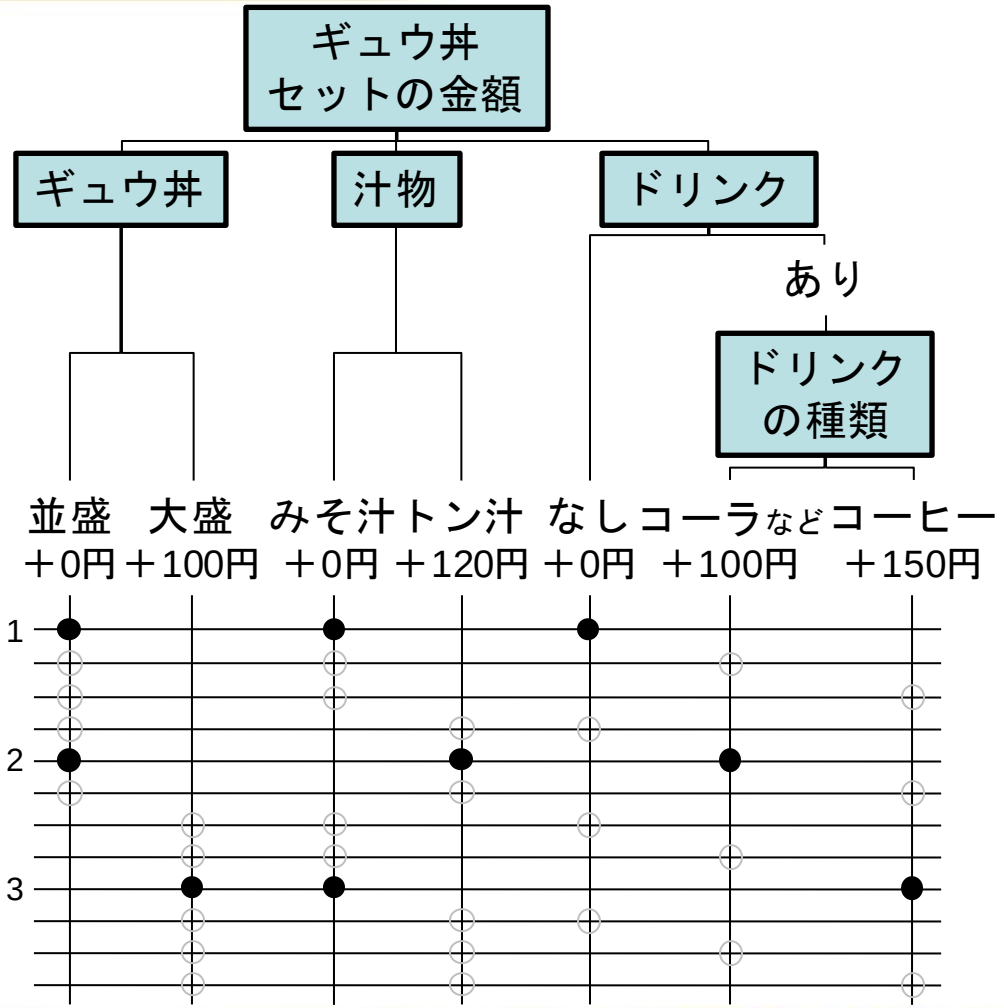


Each Choice(3ケース)

末端の**クラス**を少なくとも１回は使用

右例は「ドリンク」以下の３ケースが最大**クラス**数

→ Each Choiceを満たす
　最小テストケース数は３



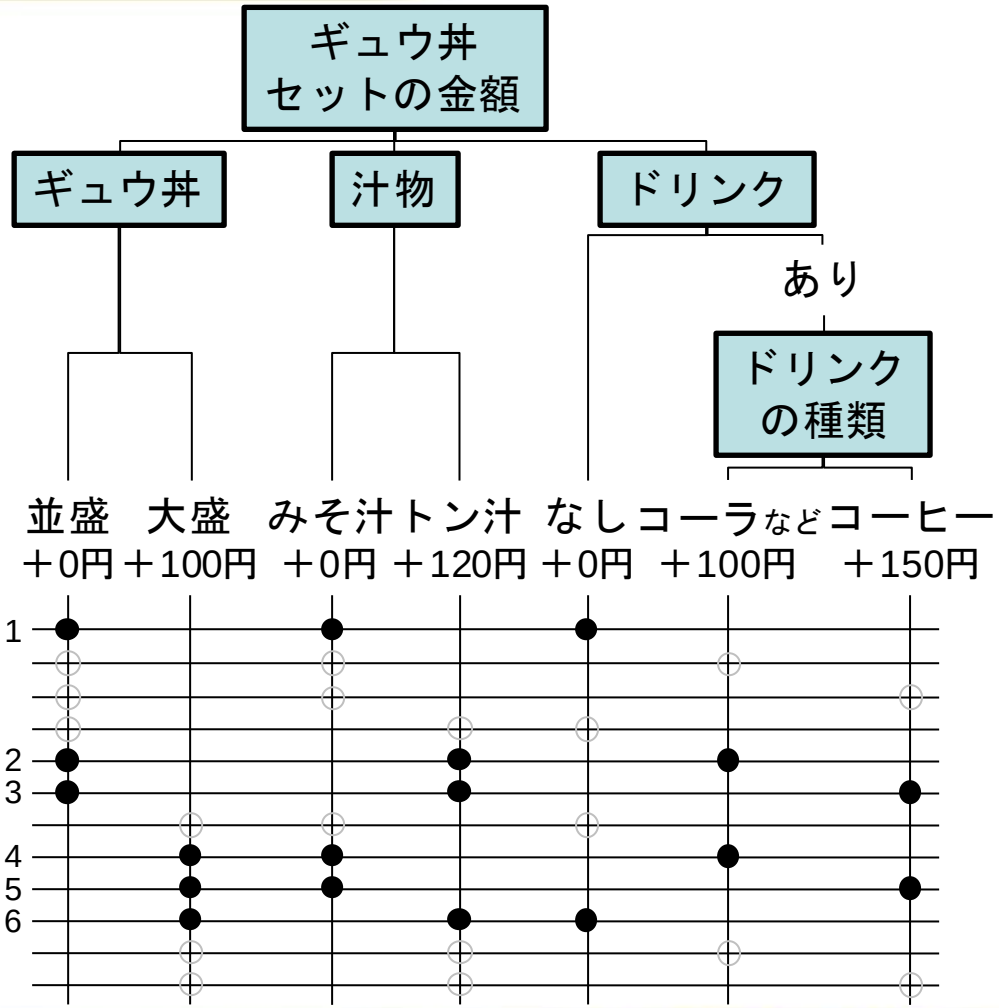
Pair-Wise(6ケース)

２CLNごとの組合せを網羅

右例は以下の2CLN間での全組合せが網羅されている

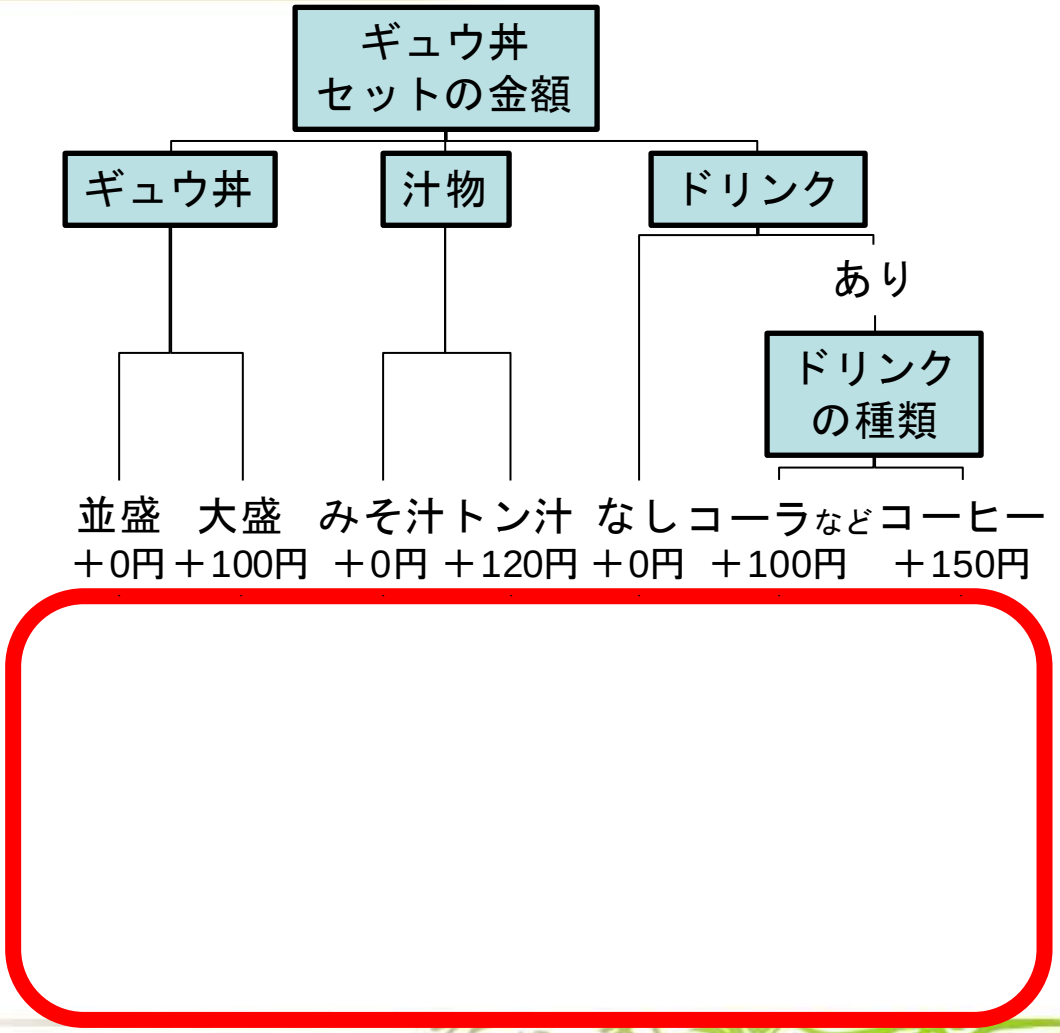
- 「ギュウ丼」×「汁物」
- 「ギュウ丼」×「ドリンク」
- 「汁物」×「ドリンク」

ツールを使って生成することが一般的(Pictなど)



【ワーク】3. 組合せテーブルを作る

- クラシフィケーションツリー末端の**クラス**の下に縦線を引こう
- 縦線を横断するように横線を引こう
  - 横線は１つのテストケースに相当する
- 横線ごとにテストケースを検討しよう
  - **「全組合せ網羅」**となるテストケースを考えよう

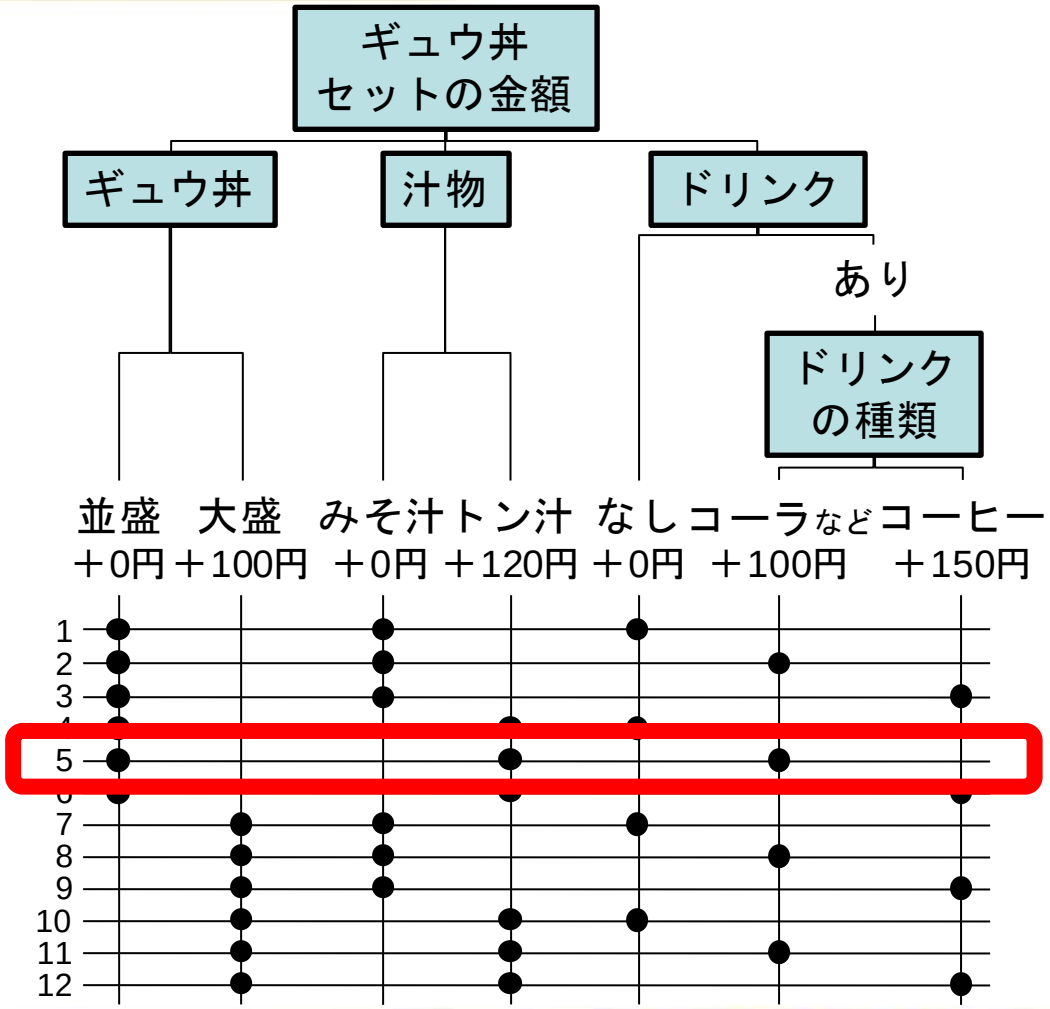


CT技法によるテストケースの作り方

1. テスト対象を選択する
2. クラシフィケーションツリーを作成する
3. 組合せテーブルを作成する
4. テストデータ、期待結果を定義する

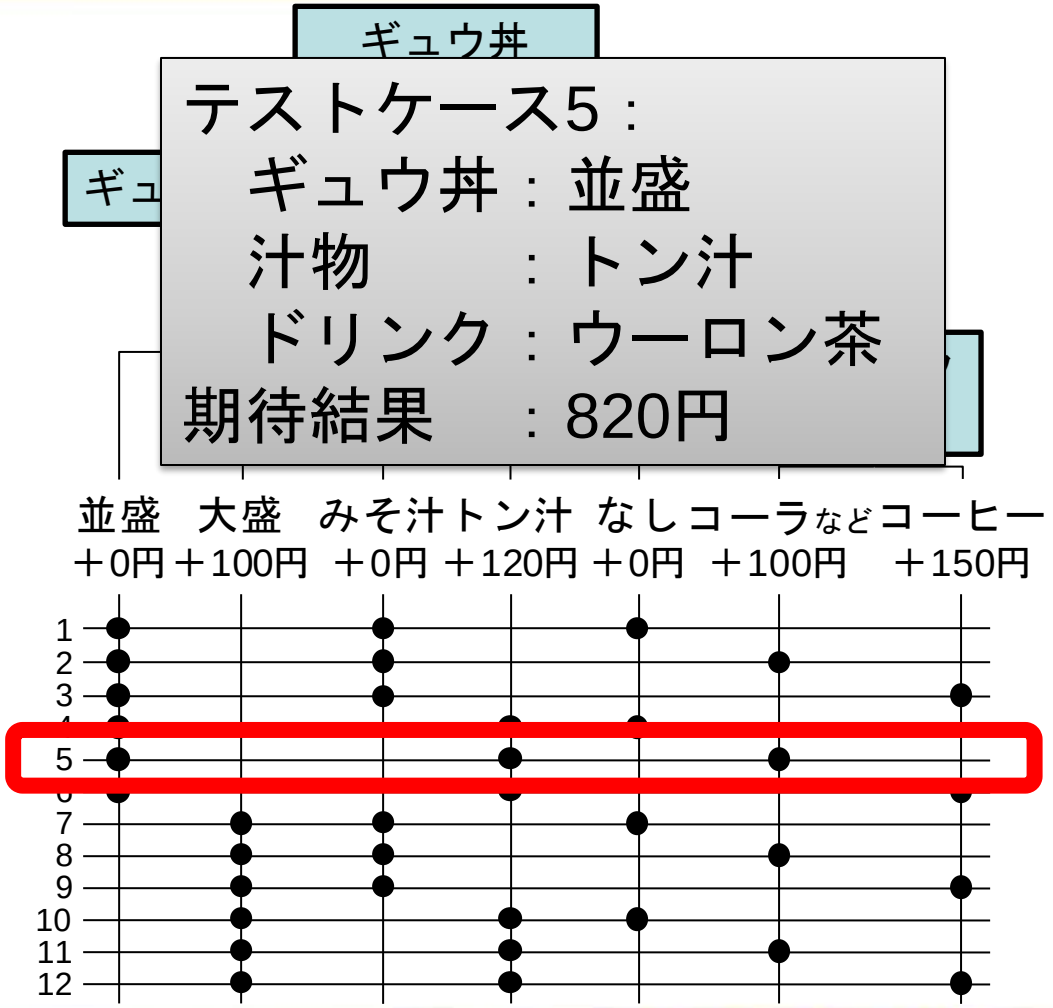
４．テストデータ、期待結果を定義する

- 組合せテーブルからテストケースごとにテストデータと期待結果を定める
- テストデータは**クラス**を具体値にする
  - 「コーラなど」→ウーロン茶
  - **クラス**の範囲であれば、テストケースごとにデータを変えると良い
- 期待結果は、決めたテストデータでどのような出力になるか、を表す



４．テストデータ、期待結果を定義する

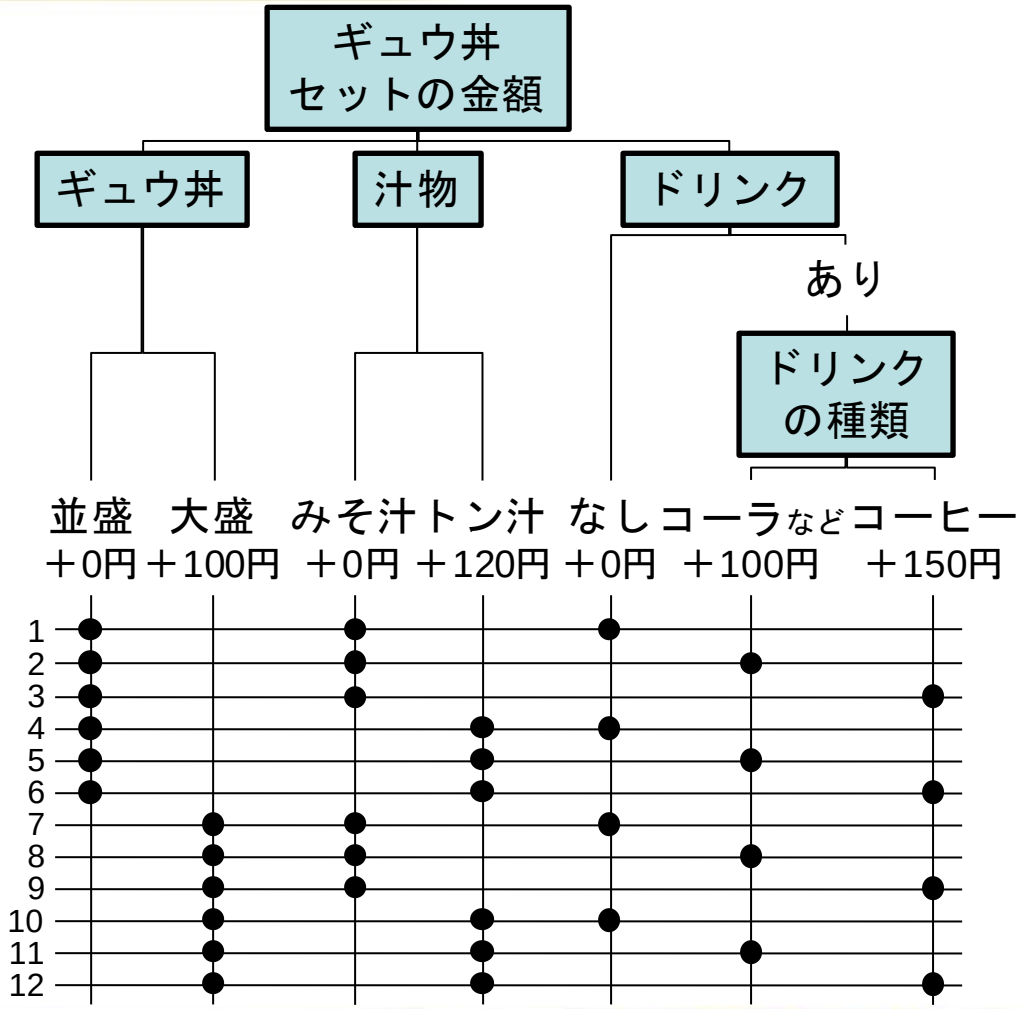
- 組合せテーブルからテストケースごとにテストデータと期待結果を定める
- テストデータは**クラス**を具体値にする
  - 「コーラなど」→ウーロン茶
  - **クラス**の範囲であれば、テストケースごとにデータを変えると良い
- 期待結果は、決めたテストデータでどのような出力になるか、を表す



【ワーク】4. テストデータ、期待結果を定義する

テストケースからひとつ選んで、
具体的なテストデータと期待結果を
書き出してみよう

テストケース？：
　ギュウ丼：？
　汁物　　：？
　ドリンク：？
期待結果　：？

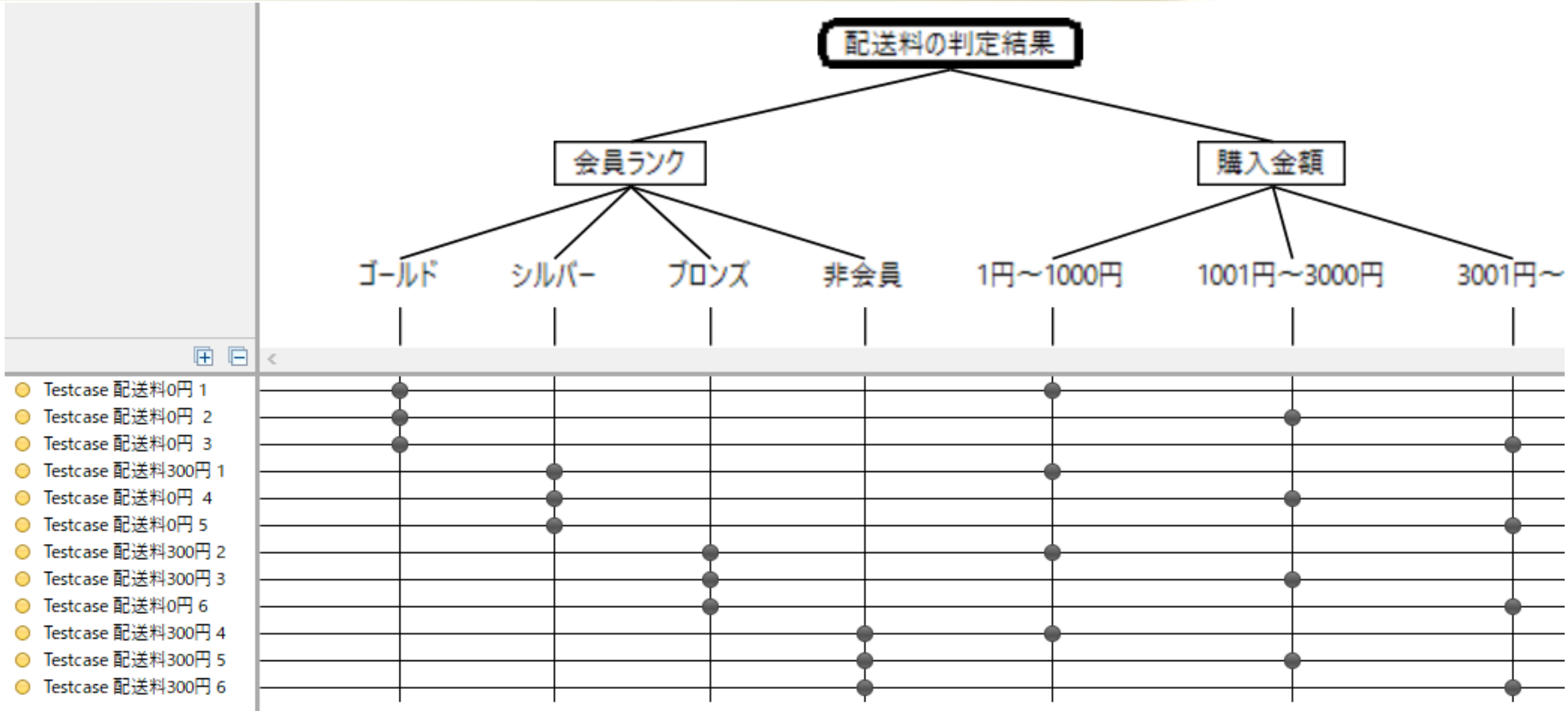


練習問題１「ECサイトの配送料の判定」

- 通販サイト政宗市場では会員登録をすると会員ランクによって
  配送料の特典サービスがある。
  会員ランクは、ゴールド、シルバー、ブロンズの3段階となっている。
- ゴールドランクは、購入金額によらず配送料は無料となる。
- シルバーランクは、購入金額が1000円を超える場合、配送料は無料となる。
- ブロンズランクは、購入金額が3000円を超える場合、配送料は無料となる。
- 上記をいずれも満たさない（非会員を含む）場合、配送料は300円となる。

- 配送料についてクラス分けを行い、クラシフィケーションツリーを使って
  全組合せの組合せテーブルを作り、期待結果を確認してください。
- なお条件が重複した場合、期待結果は一番安い配送料となります。

練習問題１の解答例


配送料の判定結果
会員ランク
購入金額
ゴールド
シルバー
ブロンズ
非会員
1円～1000円
1001円～3000円
3001円～
1 Testcase 配送料0円 1
2 Testcase 配送料0円 2
3 Testcase 配送料0円 3
4 Testcase 配送料300円 1
5 Testcase 配送料0円 4
6 Testcase 配送料0円 5
7 Testcase 配送料300円 2
8 Testcase 配送料300円 3
9 Testcase 配送料0円 6
10 Testcase 配送料300円 4
11 Testcase 配送料300円 5
12 Testcase 配送料300円 6

練習問題１の解説

- 会員ランクと購入金額によって配送料が決定される問題となります
- 問題文にある仕様から関連する項目を抜き出し、整理します
  - テストしたいこと（ルート）：配送料の判定結果
  - クラシフィケーション：「会員ランク」「購入金額」
  - 「会員ランク」クラス：「ゴールド」「シルバー」「ブロンズ」「非会員」
  - 「購入金額」クラス：「1～1000円」「1001円～3000円」「3001円～」
- 上記をクラシフィケーションツリーにし、組合せテーブルを作成していきます
  - 「条件が重複した場合、一番安い配送料になる」ため、期待結果は解答例のようになります

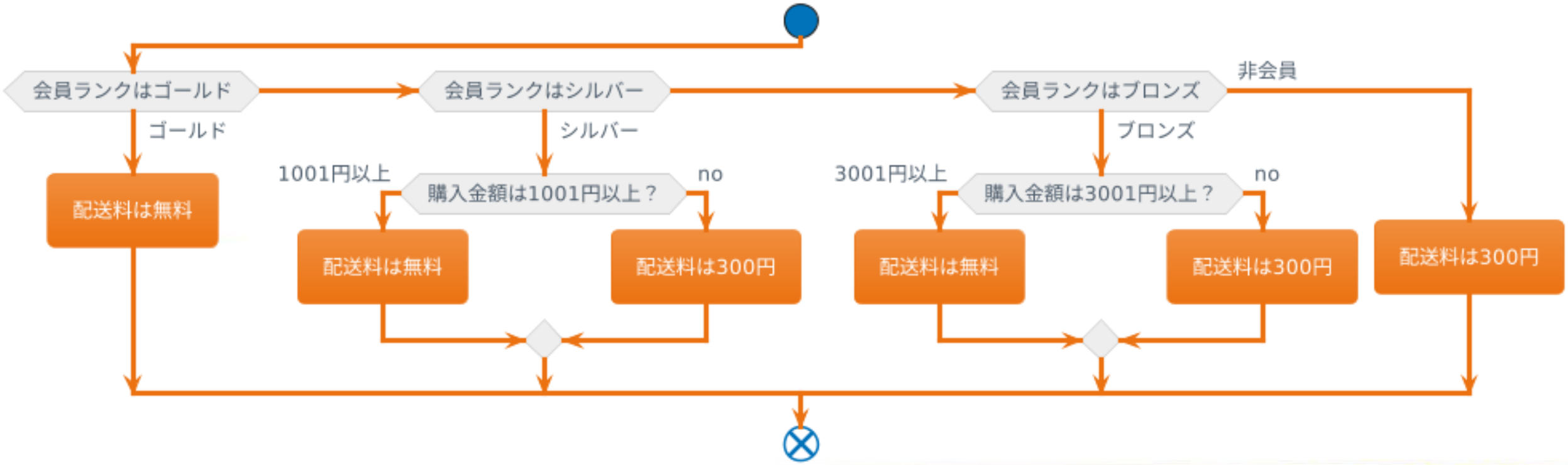
練習問題２「ECサイトの配送料の判定その２」

- 通販サイト政宗市場では配送料無料クーポンを発行することになった。
  クーポンを使用すると会員ランクを問わずだれでも、配送料が無料になる。
- 開発チームには配送料無料クーポンの有無チェックは配送料取得処理の
  一番最初に判定することを確認できている。
- 確認結果を踏まえ、さきほどの練習問題で作成したクラシフィケーション
  ツリーの成果物をもとに必要なテストを検討することとした。

- 配送料についてクラス分けを行い、クラシフィケーションツリーを使って
  テストケース数が最小（影響が少ない、または無いテストケースは間引きた
  い）となる組合せを作成してください。

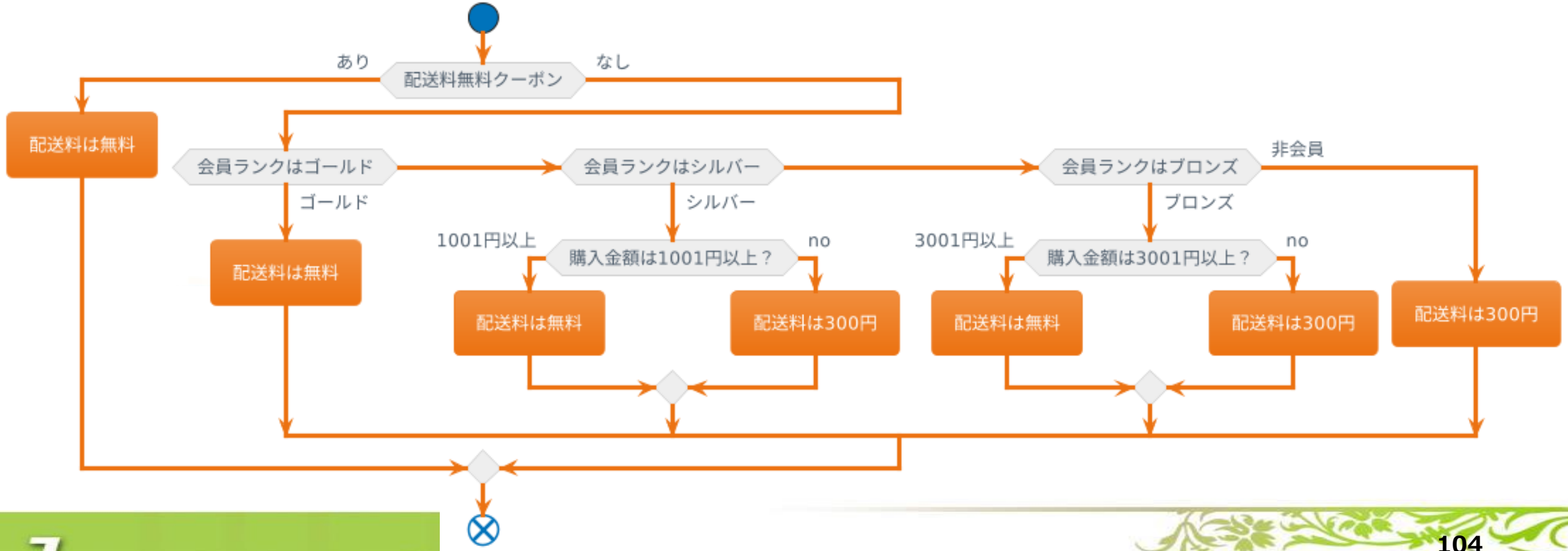
練習問題２「ECサイトの配送料の判定その２補足」

- テスト設計する前に、配送料無料クーポンに対応前の配送料の判定ロジックを開発チームにヒアリングした結果、会員ランクごとに購入金額を判定し、配送料を決めていることがわかった

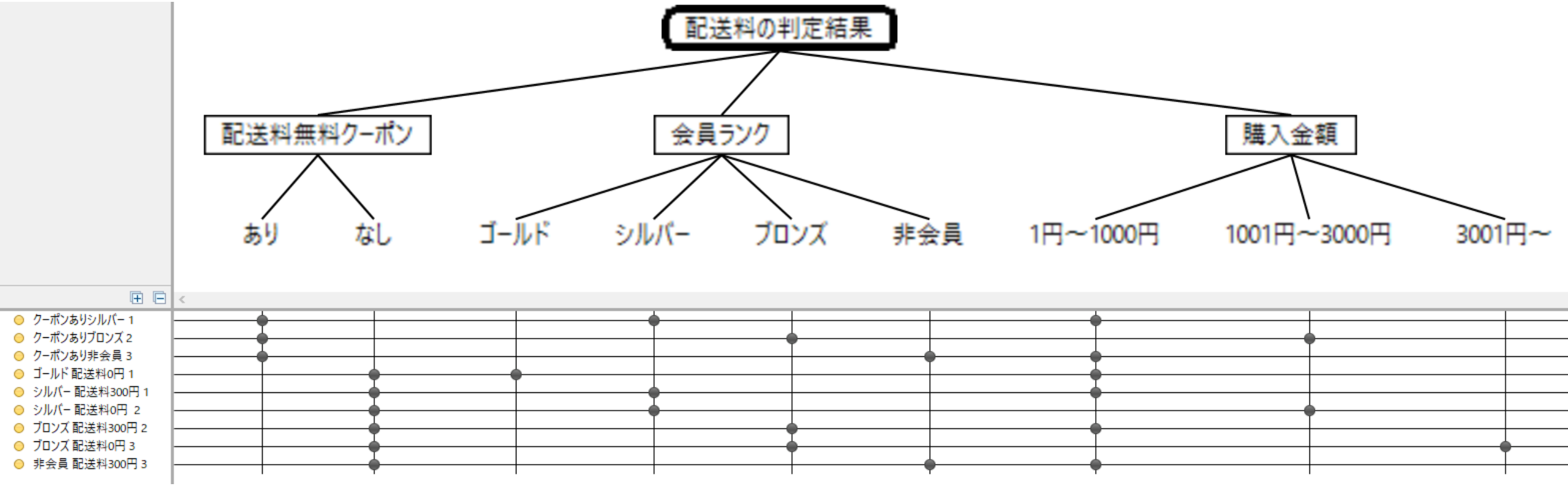


練習問題２「ECサイトの配送料の判定その２補足」

- 配送料無料クーポン対応後の配送料の判定ロジックを開発チームにヒアリングした結果、以下のロジックになることがわかった



練習問題２の解答例


配送料の判定結果
配送料無料クーポン
会員ランク
購入金額
あり
なし
ゴールド
シルバー
ブロンズ
非会員
1円～1000円
1001円～3000円
3001円～
クーポンありシルバー 1
クーポンありブロンズ 2
クーポンあり非会員 3
ゴールド 配送料0円 1
シルバー 配送料300円 1
シルバー 配送料0円 2
ブロンズ 配送料300円 2
ブロンズ 配送料0円 3
非会員 配送料300円 3

練習問題２の解説

- 今回のお題には「テストケース数が最小となる組合せを作成してください」とあったため、練習問題１のクラシフィケーションツリーに開発チームへのヒアリング結果を反映していきます
  - クーポン対応前の判定順は、会員ランク→購入金額
  - クーポン対応後の判定順は、クーポン有無→会員ランク→購入金額
- 会員ランクにより配送料が300円になる組合せにクーポンありの条件が組み合わさった場合に配送料が0円になる組合せを追加（3件）
  - ※ヒアリング結果をそのまま設計し、クーポンありの組合せを1件でも可
- クーポンなしの場合に、各会員ランクで配送料が変化する組合せを選出（12件→5件）

おわりに

- CT技法は組み合わせテストを作る技法で
  「クラシフィケーションツリー」「組合せテーブル」によって
  テストケースを生成します
- CT技法によるテスト設計は組合せで使う要素と組合せ方が
  同時に見えるために一覧性が良く、
  組合せの根拠がわかりやすく、説明しやすい
- 規模が大きい場合はツールを用いると良い
  - TESTONA

参考文献

- Grochtmann, Matthias and Klaus Grimm (1993). "Classification Trees for Partition Testing."
- ISO/IEC/IEEE 29119-4: 2021, Software and systems engineering — Software testing — Part4: Test techniques
- Rex Black(2015). Advanced Software Testing - Vol. 1, 2nd Edition: Guide to the ISTQB Advanced Certification as an Advanced Test Analyst. Rocky Nook
- Graham Bath, Judy McKay(2014). The Software Test Engineer's Handbook, 2nd Edition: A Study Guide for the ISTQB Test Analyst and Technical Test Analyst Advanced Level Certificates 2012. Rocky Nook
- Paul Ammann, Jeff Offutt(2016). Introduction to Software Testing

参考文献

- ISTQB® 用語集
  https://glossary.istqb.org/
- ASTERセミナー標準テキスト
  https://aster.or.jp/business/seminar_text.html
- ISTQB® テスト技術者資格制度 Advanced Levelシラバス日本語版 テストアナリスト
  https://jstqb.jp/syllabus.html#syllabus_advanced_altta
- 井芹洋輝(2014). クラシフィケーション・ツリー法入門.
  https://speakerdeck.com/goyoki/introduction-to-classification-tree-method
- 辰巳敬三(2019). 組み合わせテストの設計(PictMaster勉強会)
  https://www.slideshare.net/Bugler/pictmaster-2008717
- JaSST東北2019のデシジョンテーブルワークショップ資料
  https://www.jasst.jp/symposium/jasst19tohoku/pdf/S5-1.pdf