# Building a CBR system for the Cocktail Challenge

Carlos Berg, Christian Reiser

# Content

1. Problem Definition
2. Case Base Structure
3. Architecture Overview
4. Similarity Measure
5. Domain Knowledge
6. Adaptation Function
7. Expert Evaluation
8. Live Demo

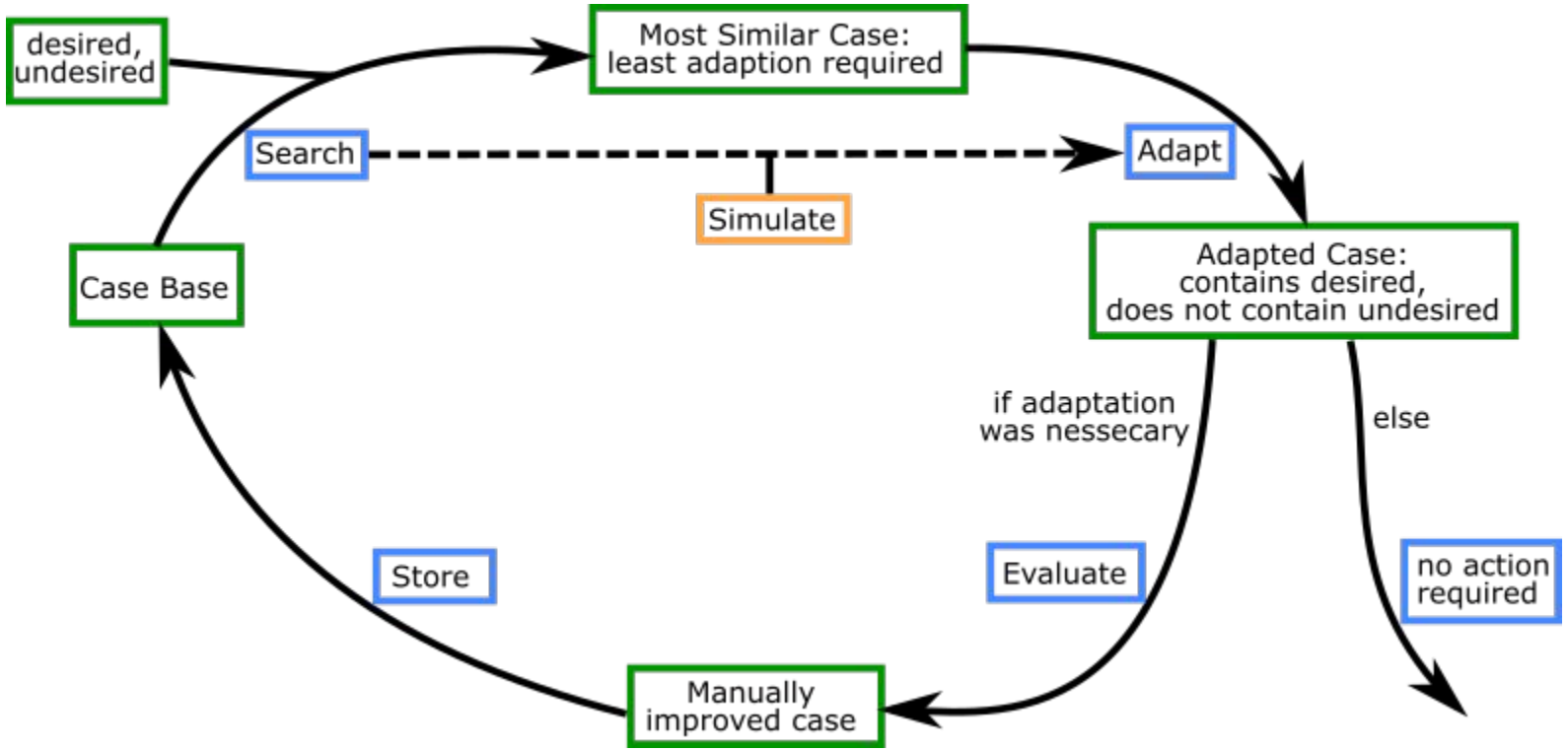
# Problem definition

- Case Base: Cocktail recipes
  - set of ingredients
  - ingredient is a tuple: (title, quantity, unit)
- Goal: return delicious cocktails, that contain
  - desired ingredients but do not contain any
  - undesired ingredients
- User provides sets desired ingredients and undesired ingredients
  - but no quantities of the desired ingredients are specified
- Case Base can be used as a starting point
  - since given recipes are supposed to be delicious

# Case Base Structure

- What are the advantages and disadvantages of the flat and hierarchical structures?
  - flat: more accurate, but takes longer
  - hierarchical: faster, but at most as accurate as a flat structure
- When should a hierarchical structure preferred over a flat one then?
  - when query times are too high
- We experienced instantaneous query times with a flat structure
  - kept the flat structure

# Architecture Overview



# Similarity Measure

- We do not have any
  - at least no explicit one
- Instead: simulate adaptation function for every case in the case base
  - Goal: all desired ingredients and no undesired ones
- Keep track of how many adaptations are needed
- The case (cocktail) that requires the fewest adaptation is chosen
- Inspired by Levenshtein Edit Distance
  - given two strings: how many insert, update, delete operations are needed to transform one string into the other

# Domain Knowledge

- Extracted the set of ingredients from the provided case base
- Manually assigned all ingredients a category
- Four categories: alcoholic, nonalcoholic, fruit and special
- Given a drink vodka with orange juice and a desired ingredient gin, what would you rather replace? Vodka by gin? Orange juice by gin?

```
<ingredient category="fruit" food="lime zest"/>
<ingredient category="alcoholic" food="porto"/>
<ingredient category="nonalcoholic" food="strawberry juice"/>
<ingredient category="fruit" food="berry"/>
<ingredient category="alcoholic" food="champagne"/>
<ingredient category="special" food="vanilla sugar"/>
<ingredient category="alcoholic" food="malibu rum"/>
<ingredient category="fruit" food="strawberry"/>
```

# Adaptation Function

- Returned result must contain all desired ingredients no undesired ones
- Existing recipes in case base are supposed to taste well (rated by human experts)
- Modify an existing cocktail as little as possible
- Four strategies for adaptation
    - ordered by their favorableness

# Types of ingredients

- missing desired: requested but not contained in candidate cocktail
- contained undesired: contained in candidate cocktail, but need to be removed
- optional: contained in candidate cocktail, neither desired nor undesired
- Example, candidate cocktail: gin, tonic, cucumber, water
  - desired: vodka, tonic, lemon, sugar
  - undesired: cucumber, berry, water
  - missing desired: vodka, lemon, sugar
  - contained undesired: cucumber, water
  - optional: gin
- Goal: the sets missing desired and contained undesired should both be empty

# First strategy

- Replace contained undesired by missing desired
- Both must be of the same category
  - otherwise composition would be changed
- Most favorable strategy, since two birds are killed with one stone
  - from both sets one element is taken away
- Example: gin, tonic, cucumber, water
  - missing desired: vodka, lemon, sugar → vodka, sugar
  - contained undesired: cucumber, water → water
  - optional: gin

# Second strategy

- Replace optional by desired
- Again: both of the same category
- Only from missing desired set one element is taken away
- Example: gin, tonic, lemon, water
  - missing desired: vodka, sugar → sugar
  - contained undesired: water
  - optional: gin → ∅

# Third strategy

- Replace undesired by random ingredient from the same category
- Equally as favorable as second strategy
  - they commute with each other
- There is some risk due to the randomness
  - is the random choice to the user's liking?
- Example: vodka, tonic, lemon, water
  - missing desired: sugar
  - contained undesired: water → ∅
  - optional: ∅
  - random: orange juice

# Fourth strategy

- Add small amount of missing desired
- Fall back strategy
  - Least favorable since composition of cocktail might be ruined
- Only strategy without a same category replacement
- Counts as two modification steps towards our edit distance
  - Cocktails that require that step are less likely to be selected
- Example: vodka, tonic, lemon, orange juice
  - missing desired: sugar → ∅
  - contained undesired: ∅
  - optional: ∅

# Quantities

- Annotation of alcohol content of all alcoholic ingredients
- Upon replacement of an alcoholic ingredient quantity is adjusted
  - Alcohol content should stay approximately th the same
- Example:
  - Old ingredient: 35cl Blue Curaçao with 25%
  - New ingredient: Malibu Rum with 40%
  - Since Malibu Rum is stronger than Blue Curaçao the alcohol level is decreased:
  - 35cl → 22cl = 35cl × 25% ÷ 40%

# Expert evaluation

- Whether a cocktail tastes good or not can be assessed by a human expert
- Only if there was some adaptation necessary
- Manually improve adapted cocktail
- Command line interface:
  - add ingredient
  - replace ingredient
  - remove ingredient
- System checks that still all desired an no undesired ingredients are in modified cocktail

# Live Demo