関係データベースを用いたルールベースシステムの実装

目次

1. 外部仕様
2. 内部仕様
   - クラス設計
   - 関係データベース
   - 推論アルゴリズム
3. 実行デモ
4. 考察
5. 感想

外部仕様(1/2) – ZOOKEEPER

動物たちの特徴とルール集合を入力として取得し，
ルールに基づいてそれぞれの動物の同定をおこなう

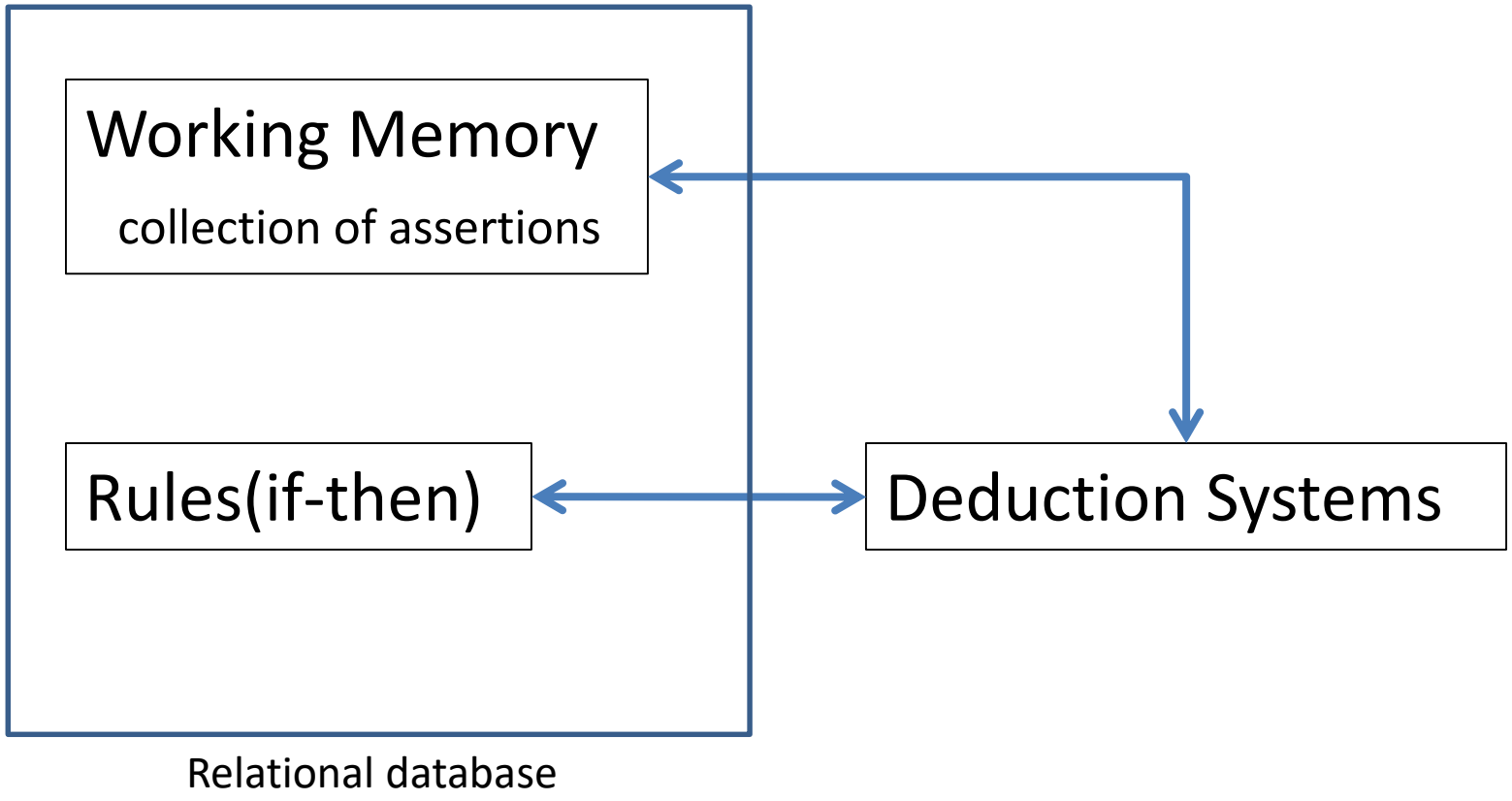
- 入力 – CSVファイルから読み込み
  - 動物の特徴集合
  - If-then ルール集合
- 出力 – 標準出力へ表示
  - 同定の結果
  - 同定されなかった動物は途中経過を出力

外部仕様(2/2) – 実行

- コマンドラインから実行
  - ディレクトリ/Zookeeper/bin/
  - java -classpath .;../csv.jar Zookeepe
- 読み込むファイル
  - features.csv
  - identifier.csv
  - antecedent.csv(if-rules)
  - consequent.csv(then-rules)

内部仕様(1/12) – ルールベースシステム

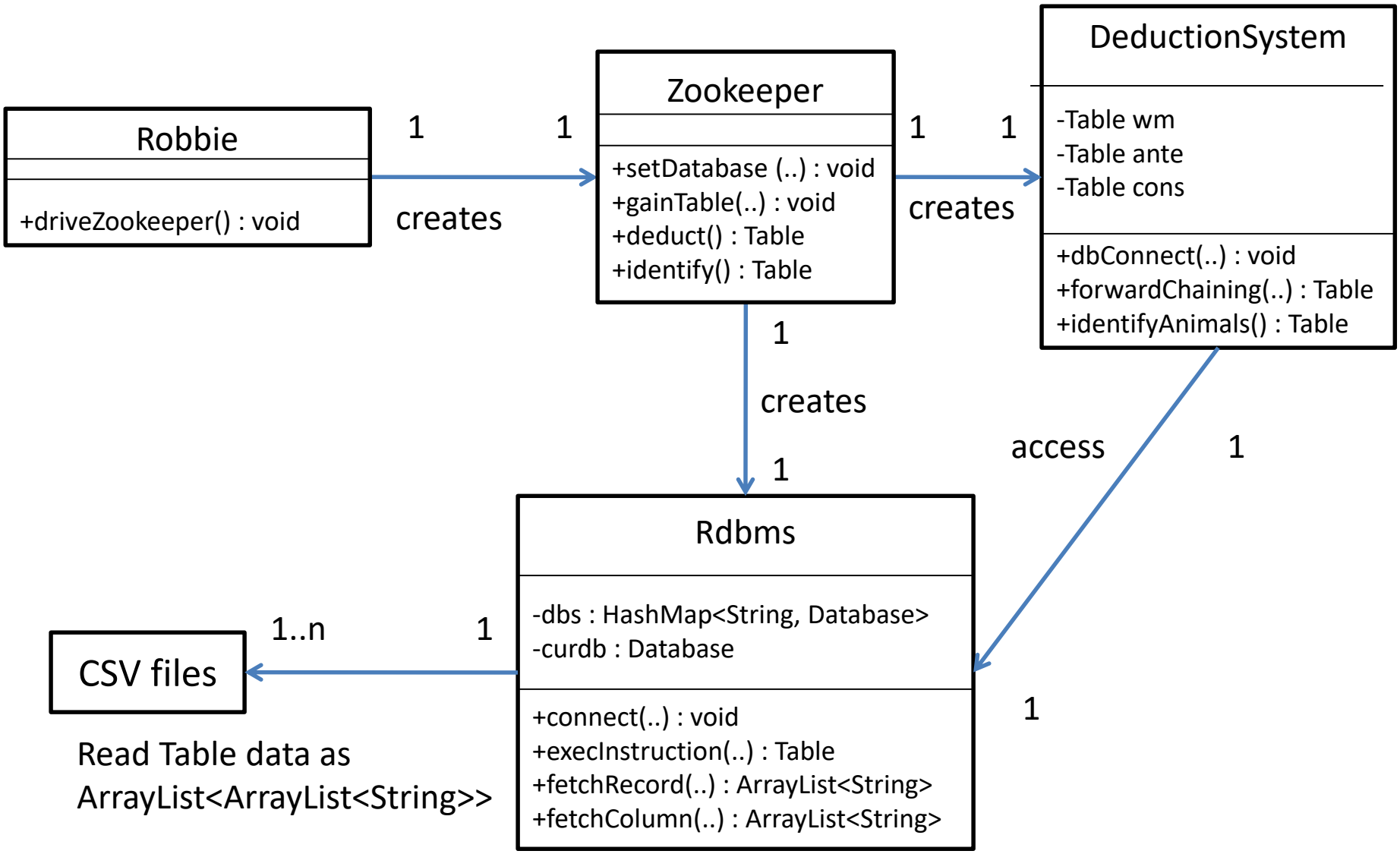
- ルールベースシステムの構成



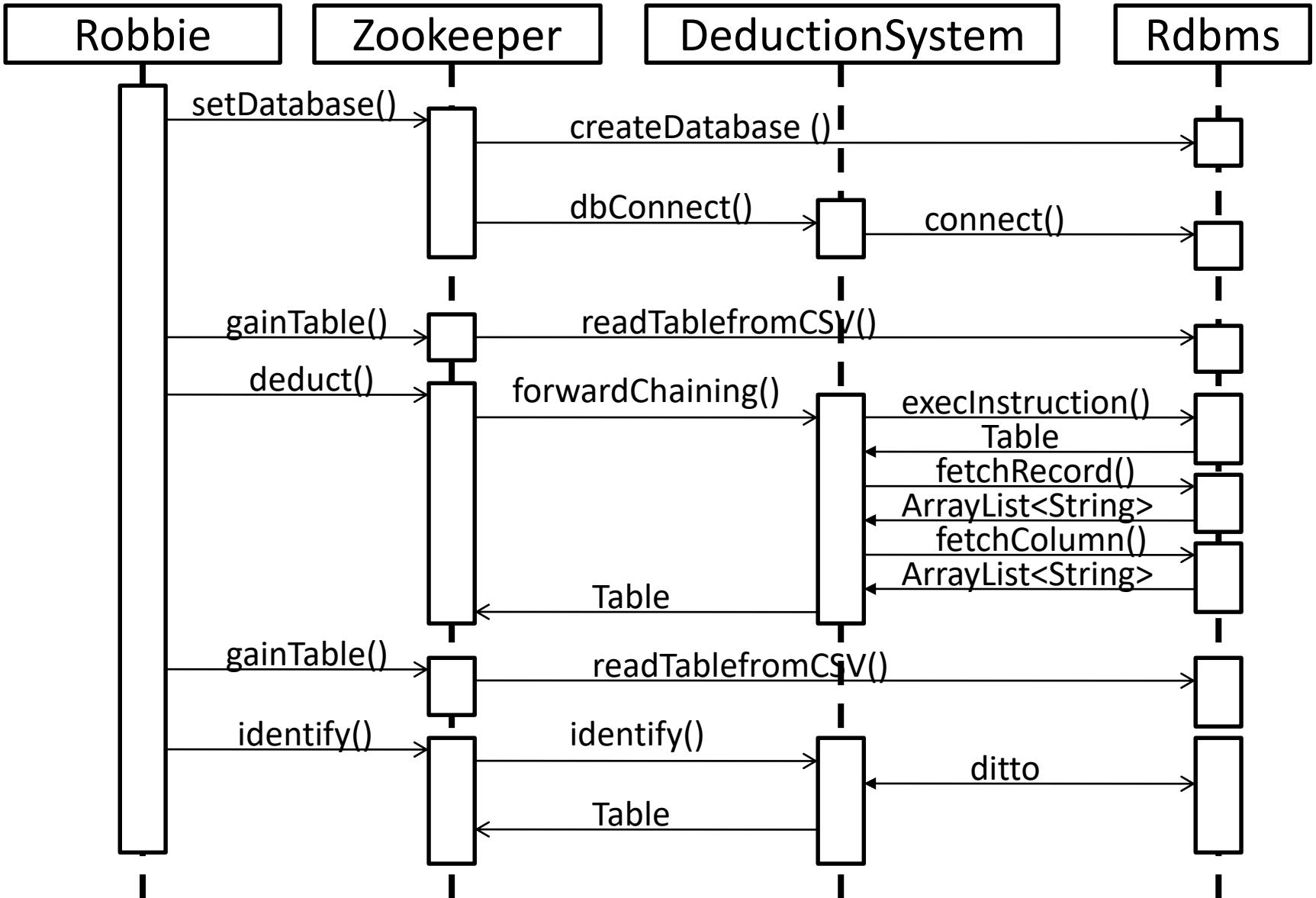
内部仕様(2/12) – ZOOKEEPER

- データ構造
  - 関係データベース
- 推論方式
  - Forward Chaining(前向き探索)
  - パターンマッチングを関係演算によって実装
- Assertion
  - 動物の特徴(Ex. Alpha has hair)
- Working Memory
  - collection of assertions

内部仕様(3/12) – クラス図



内部仕様(4/12) – シーケンス図



内部仕様(5/12) – クラス詳細(1/3)

- Robbie
  - Zookeeperを作成，駆動する(driveZookeeper())
  - 動物の特徴とルール集合をZookeeperに入力する
  - ファイル名を指定してCSVファイル読み込みのメッセージを送ることで入力の動作を表現する

内部仕様(6/12) – クラス詳細(2/3)

- Zookeeper
  - RdbmsとDeductionSystemをもつルールベースシステム
  - RdbmsとDeductionSystemとを関連づける(setDatabase())
  - 推論，同定をDeductionSystemに実行させ，(deduct(), identify())結果を受け取る
- DeductionSystem
  - Forward Chainingによる推論をおこなう(forwardChaining())
  - 動物の種類をidentifierに基づいて同定する(identify())

内部仕様(7/12) – クラス詳細(3/3)

- Rdbms
  - 関係データベースと問い合わせ処理を提供
  - 実行可能な問い合わせ(execInstruction()):
    - ADD:レコード追加
    - SELECT:選択
    - PROJECT:射影
    - JOIN:結合

    Ex. SELECT table with class = Wrench
  - Tableからデータを取り出すためのメソッド fetchRecord()とfetchColumn()

内部仕様(8/12) – 関係データベース(1/3)

- クラスTable
  - records: LinkedHashSet<HashMap<String, String>>
    単一のrecord : HashMap<String, String>の集合表現
  - fields : LinkedHashSet<String>
    field : Stringの集合表現
  - recordsやfieldsを操作するメソッドを提供する

fields →

records

name	verb	feature
Alpha	has	hair
Bravo	has	feathers
Charlie	eats	meat

内部仕様(9/12) – 関係データベース(2/3)

- ルール集合

antecedent

rule	if-verb	if-feature
Z1	has	hair
Z2	gives	milk
Z3	has	feathers
etc…		

consequent

rule	then-verb	then-feature
Z1	is	a-mammal
Z2	is	a-mammal
Z3	is	a-bird
etc…		

- Working Memory

workingmemory

name	verb	feature
Alpha	has	hair
Alpha	chews	cud
Bravo	has	feathers
Charlie	eats	meat
Charlie	gives	milk
Alpha	is	a-mammal
Charlie	is	a-mammal
Bravo	is	a-bird
etc..		

内部仕様(10/12) – 関係データベース(3/3)

- クラスDatabase
  - tables: HashMap<String, Table>

    テーブル集合をHashMapで表現
  - 関係演算とテーブル間操作，ファイル入出力を提供する
    - ADD,SELECT,PROJECT,JOIN演算
    - CSV形式でのファイル入出力
- クラスInstruction
  - データベースへの問い合わせを解釈・実行する
  - 字句解析のみ

内部仕様(11/12) – Forward Chaining

- アルゴリズム – 関係演算による前向き推論

ループ　新たなassertionが生成されなくなるまで
　ループ　各々のルールについて
　　workingmemoryとantecedentを結合(JOIN)する
　　ループ　結合されたテーブルの各々の動物について
　　if　ある動物がすべてのantecedentを満足している
　　then その動物についてconsequentを参照して新たな
　　assertionをworkingmemoryに追加する

(Working Memory, ルール集合は有限なので，必ず停止する)

内部仕様(12/12) – Z4のマッチング例

workingmemory

name	verb	feature
Alpha	has	hair
Alpha	chews	cud
Echo	lays	eggs
Echo	flies	

antecedent

rule	if-verb	if-feature
Z1	has	hair
Z4	flies	
Z4	lays	eggs

join workingmemory and [select antecedent with rule = Z4]
with verb = if-verb and feature = if-feature

name	verb	feature	rule	if-verb	if-feature
Echo	lays	eggs	Z4	lays	eggs
Echo	flies		Z4	flies	

record数が
antecedentの
rule = Z4と等しい

add Echo is a-bird to workingmemory

select consequent with rule = Z4から参照

実行デモ

考察

- 工夫点
  - データベースアクセス
    - 外部からはすべて問い合わせによってアクセスする
    - 外部から直接テーブル操作，関係演算をしない
    - テーブルのデータを外部から取得するには，fetchRecord()かfetchColumn()を用いる
  - 複数のassertionsに対して同時にマッチング可能
  - 関係演算によるマッチング
    - JOIN演算によって実現

考察

- 既知の問題点
  - 推論において意味的な解析をしていない

    Ex. A bird flies and a penguin is a bird, but a penguin does not fly.

    Ex. Alpha is a tiger and Alpha is a penguin!?
  - 適用済みルールを何度も適用する
  - Backward Chaining未実装
  - 関係データベース
    - 問い合わせはADD,SELECT,PROJECT,JOINしかできない
    - 一貫性制約が存在しない

参考文献

1. P. H. Winston, Artificial Intelligence, *Addison-Wesley Pub*, 2005
2. Tucker,『憂鬱なプログラマのためのオブジェクト指向開発講座』,翔泳社, 1998