CS-602 - Design of Problem Solvers

Chapter 2 - Datasets for Logic Problem Solvers

Dr. Mahdi Khemakhem

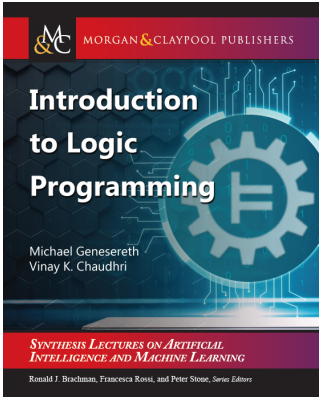
Department of Computer Science
College of Computer Engineering and Science
Prince Sattam bin Abdulaziz University

AY - 2025/2026

Outline

Materials

- **Textbook:** "Introduction to logic programming", 2022, by *Genesereth, Michael, and Vinay K. Chaudhri*
- $\Rightarrow$ **Read Chapter 2 for this chapter's material**



Outline

Introduction to Datasets

What is a Dataset?

Datasets are **collections of facts** representing **knowledge about the world**. They can be used **standalone** or **combined with Logic Programs** to build complex information systems.

Chapter Objectives:

1. **Conceptualization:**
   - Understanding how to model the world as **objects** and **relationships**
2. **Formal Language:**
   - Learning the syntax for *encoding information* as datasets
3. **Epilog Examples:**
   - Practical dataset examples using Epilog notation

Outline

Conceptualization: Modeling the World

Core Concept

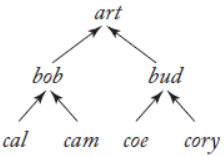
We model the world as *objects* and *relationships* between them.

Two Fundamental Components:

1. **Objects:**
   - People, places, buildings, vehicles
   - Example: `art, bob, cal`
2. **Relationships:**
   - Parenthood, friendship, location
   - Example: `parent(art, bob)`

Representation Methods:

- **Graphs:** Nodes (objects) + Arcs (relationships)
- **Tables:** Rows represent relationship instances
- **Sentences:** **Formal logic notation (our choice)**



parent

art	bob
art	bud
bob	cal
bob	cam
bud	coe
bud	cory

Sentential Representation

Formal Language Approach

We encode *individual relationships* as **sentences** in a formal language, where each fact combines a **predicate** with its **arguments**.

Syntax Structure:

General Form:

- `predicate(object1, object2)`
- **Predicate:** relationship name
- **Objects:** entities involved

Example - Kinship:

```
1 parent(art,bob)
2 parent(art,bud)
3 parent(bob,cal)
4 parent(bob,cam)
5 parent(bud,coe)
6 parent(bud,cory)
```

Each line is a **fact**: `parent` is the predicate, arguments are the objects in the relationship.

Why Sentential Representation?

Design Choice

While **graphs** and **tables** are intuitive, *sentential representation* is **optimal for logic programming** and computational reasoning.

Our Approach:

1. **Facts as Sentences:**
   - Each fact = one logical sentence
   - Example: `parent(art, bob)`
2. **World States as Sets:**
   - Different states = different sets of sentences
   - Easy to query, update, and reason about

Terminology Note

We use **relation** and **relationship** interchangeably throughout this course. While mathematically distinct, the difference is unimportant for our purposes.

Outline

Constants and Vocabulary

What are Constants?

Constants are the basic building blocks: **lowercase strings**, **digits**, **underscores**, **periods**, or **quoted ASCII strings**.

Syntax Rules:

Valid Constants:

- `a, b, comp225`
- `123, 3.14159`
- `barack_obama`
- `"Mind your p's and q's!"`

Invalid Constants:

- `Art` (uppercase)
- `p&q` (ampersand)
- `the-house` (hyphens)

Key Rule: Uppercase letters prohibited except within double quotes.

Vocabulary

A *vocabulary* is a **collection of constants** used in a dataset.

Types of Constants

Three Categories

Constants are classified into three types based on their *semantic role* in datasets.

Constant Categories:

1. **Symbols:**
   - Represent **objects** in the world
   - Example: `art, bob, cody`
2. **Constructors:**
   - Create **compound names** for objects
   - Example: `pair(a, b)`
3. **Predicates:**
   - Represent **relationships** between objects
   - Example: `parent(art, bob)`

Arity (Argument Count):

- **Unary:** 1 argument (e.g., `adult(art)`)
- **Binary:** 2 arguments (e.g., `parent(art, bob)`)
- **Ternary:** 3 arguments (e.g., `prefers(x, y, z)`)
- **n-ary:** n arguments

Note: Zero-argument predicates represent simple true/false conditions.

Ground Terms

Definition

A *ground term* is either a **symbol** or a **compound name**. The term "ground" means it contains **no variables**.

Two Types:

1. **Simple Symbols:**
   - Basic constants: `a, b, art, bob`
2. **Compound Names:**
   - Formed from **n-ary constructor** + **n ground terms**
   - Syntax: `constructor(term1, term2, ...)`

Example: Binary Constructor

Given symbols `a, b` and binary constructor `pair`:

- Valid compound names: `pair(a,a), pair(a,b), pair(b,a), pair(b,b)`

Herbrand Universe

Definition

The *Herbrand Universe* for a vocabulary is the **set of all ground terms** that can be formed from its **symbols** and **constructors**.

Size Characteristics:

1. **Without Constructors:**
   - Herbrand Universe is **finite**
   - Contains only the symbols themselves
   - Example: `{a, b, c}`
2. **With Constructors:**
   - Herbrand Universe is **infinite**
   - Contains symbols + all possible compound names
   - Example:`{a, b, pair(a,b), pair(a,pair(b,c)), ...}`

Infinite Growth

With constructor `pair`, we can create infinitely nested terms:
`pair(a,b)`,
`pair(a,pair(b,c))`,
`pair(a,pair(b,pair(c,d)))`,
...

Facts and Herbrand Base

What is a Fact?

A *fact* (datum/factoid) is formed from an **n-ary predicate** and **n ground terms**.
Syntax: `predicate(term1, term2, ...)`

Example:

- Predicate: `r` (binary)
- Symbols: `a, b`
- Fact: `r(a,b)`

More Examples:

- `parent(art, bob)`
- `likes(abby, cody)`
- `adult(art)`

Herbrand Base

The *Herbrand Base* is the **set of all possible facts** that can be formed from a vocabulary.

Example: Binary Predicate

Vocabulary: symbols `a, b` + binary predicate `r`
Herbrand Base: {`r(a,a), r(a,b), r(b,a), r(b,b)`}

Datasets: Formal Definition

What is a Dataset?

A *dataset* is a **collection of facts** that characterizes the **state** of an application. Formally, it's **any subset** of the Herbrand Base.

Closed-World Assumption:

1. **Included Facts:**
   - Facts **present** in dataset → *assumed TRUE*
2. **Excluded Facts:**
   - Facts **absent** from dataset → *assumed FALSE*
3. **Different States:**
   - Each dataset represents a **unique world state**

Key Insight

A dataset explicitly represents **what we believe is true**; everything else is implicitly **false**. This is called the **closed-world assumption**.

Outline

Example 1: Sorority World (1/3)

Consider the interpersonal relations of a small sorority. There are just four members—Abby, Bess, Cody, and Dana. Some of the girls like each other, but some do not.

The following table shows one set of possibilities. The checkmark in the first row here means that Abby likes Cody, while the absence of a checkmark means that Abby does not like the other girls (including herself). Bess likes Cody too. Cody likes everyone but herself. And Dana also likes the popular Cody.

	Abby	Bess	Cody	Dana
Abby			✓	
Bess			✓	
Cody	✓	✓		✓
Dana			✓	

Example 1: Sorority World (2/3)

In order to encode this information as a dataset, **we adopt a vocabulary with four symbols (`abby`, `bess`, `cody`, `dana`) and one binary predicate (`likes`)**. Using this vocabulary, we can encode the information by writing the dataset shown below.

```
1 likes(abby,cody)
2 likes(bess,cody)
3 likes(cody,abby)
4 likes(cody,bess)
5 likes(cody,dana)
6 likes(dana,cody)
```

→**Note that the likes relation has no inherent restrictions.**

- *It is possible for one person to like a second without the second person liking the first.*
- *It is possible for a person to like just one other person or many people or nobody.*
- *It is possible that everyone likes everyone or no one likes anyone.*

Example 1: Sorority World (3/3)

Even for a small world like this one, there are quite a few possible ways the world could be.

- Given four girls, there are **sixteen possible instances** of the likes relation—`likes(abby,abby)`, `likes(abby,bess)`, `likes(abby,cody)`, `likes(abby,dana)`, `likes(bess,abby)`, and so forth.
- Each of these sixteen can be either **true or false**.
- There are 2^{16} **(i.e., 65,536) possible combinations** of these true-false possibilities; and so there are 2^{16} **possible states** of this world and, therefore, 2^{16} **possible datasets**.

Example 2: Kinship (1/4)

As another example, consider a small dataset about kinship. The terms in this case once again represent people.The predicates name properties of these people and their relationships with each other. In our example, we use:

- The binary predicate `parent` to specify that one person is a parent of another. The sentences below constitute a dataset describing **six instances** of the parent relation.
- **The person named `art` is a parent of the person named `bob` and the person named `bea`; `bob` is the parent of `cal` and `cam`; and `bea` is the parent of `coe` and `cory`.**

```
1 parent(art,bob)
2 parent(art,bea)
3 parent(bob,cal)
4 parent(bob,cam)
5 parent(bea,coe)
6 parent(bea,cory)
```

Example 2: Kinship (2/4)

The `adult` relation is **a unary relation**, i.e., a simple property of a person, not a relationship with other people. In the dataset below, everyone is an `adult` except for Art's grandchildren.

```
1 adult(art)
2 adult(bob)
3 adult(bea)
```

Example 2: Kinship (3/4)

We can express **gender** with **two unary predicates** `male` and `female`. The following data expresses the genders of all of the people in our dataset. →Note that, in

principle, **we need only one relation** here, since **one gender is the complement of the other**. However, representing both allows us to enumerate instances of both gender equally efficiently, which can be useful in certain applications.

```
1 male(art) female(bea)
2 male(bob) female(coe)
3 male(cal) female(cory)
4 male(cam)
```

Example 2: Kinship (4/4)

As an example of a **ternary relation**, consider the data shown below.

- Here, we use `prefers` to represent the fact that **the first person likes the second person more than the third person**.
- **For example, the first sentence says that `art` prefers `bea` to `bob`; the second sentence says that `bob` prefers `cal` to `cam`.**

```
1 prefers(art,bea,bob)
2 prefers(bob,cal,cam)
```

→**Note that the order of arguments in such sentences is arbitrary.**

- Given the meaning of the `prefers` relation in our example, the first argument denotes the subject, the second one is the person who is preferred, and the third one denotes the person who is less preferred.
- We could equally well have interpreted the arguments in other orders.
- **The important thing is consistency—once we choose to interpret the arguments in one way, we must stick to that interpretation everywhere.**

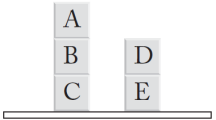
Kinship vs. Sorority World

One **noteworthy difference** between **Sorority World** and **Kinship** is that there is **just one relation** in the former (i.e., the `likes` relation), whereas there are **multiple relations** in the latter (**three unary predicates**, **one binary predicate**, and **one ternary predicate**).
A more **subtle** and **interesting difference** is that the relations in **Kinship** are **constrained in various ways** while the `likes` relation in **Sorority World is not**.

- It is possible for any person in **Sorority World** to like any other person; all combinations of likes and dislikes are possible.
- By contrast, in **Kinship there are constraints that limit the number of possible states**.
  - *For example, it is not possible for a person to be his own parent, and it is not possible for a person to be both male and female.*

Example 3: Block World (1/4)

The **Blocks World** is a popular application area for illustrating ideas in the field of Artificial Intelligence. A typical **Blocks World scene** is shown in the bellow figure.

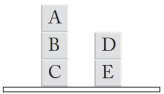


In order to describe this scene, we adopt a vocabulary with five symbols `(a, b, c, d, e)`, with **one symbol for each of the five blocks** in the scene.

Example 3: Block World (2/4)

In a **spatial conceptualization** of the **Blocks World**, there are numerous meaningful relations.

- For example, it makes sense to talk about the relation that **holds between two blocks if and only if one is resting on the other**.
  - *In what follows, we use the predicate* `on` *to refer to this relation.*
- We might also talk about the relation that **holds between two blocks if and only if one is anywhere above the other** (i.e., the first is resting on the second or is resting on a block that is resting on the second, and so forth).
  - *In what follows, we use the predicate* `above` *to talk about this relation.*

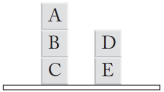



```
1 on(a,b)
2 on(b,c)
3 on(d,e)
```

```
1 above(a,b)
2 above(a,c)
3 above(b,c)
4 above(d,e)
```

Example 3: Block World (3/4)

There is the relation that **holds of three blocks that are stacked one on top of the other**.



- We use the predicate `stack` as a name for this relation.
- We use the predicate `clear` to denote the relation that **holds of a block if and only if there is no block on top of it**.
- We use the predicate `table` to denote the relation that **holds of a block if and only if that block is resting on the table**.

```
1 stack(a,b,c)
```

```
1 clear(a)
2 clear(d)
```

```
1 table(c)
2 table(e)
```

Example 3: Block World (4/4)

As with **Kinship**, the relations in **Blocks World** are constrained in various ways.

- **For example, it is not possible for a block to be on itself.**

Moreover, **some of these relations are entirely determined by others**.

- **For example, given the** `on` **relation, the facts about all of the other relations are entirely determined.**

$\rightarrow$In a later chapter, we see **how to write out definitions for such concepts** and thereby **avoid having to write out individual facts** for such defined concepts.

Example 4: Food World (1/3)

As another example of these concepts, consider a small dataset about food and menus. The goal here is to create a dataset **that lists meals that are available at a restaurant on different days of the week**.

- The symbols in this case come in two types:
  - days of the week (`monday, ... , friday`),
  - and, different types of food (`calamari, vichyssoise, beef, and so forth`).
- There are **three** constructors:
  - a **3-ary constructor** for three course meals (`three`),
  - a **4-ary constructor** for four course meals (`four`),
  - and, a **5-ary constructor** for five course meals (`five`).
- There is a **single binary predicate** `menu` that relates days of the week and available meals.

Example 4: Food World (2/3)

The following is an example of a dataset using this vocabulary:

- On Monday, the restaurant offers a three course meal with calamari and beef and shortcake, and it offers a different three course meal with puree and beef and ice cream for dessert.
- On Tuesday, the restaurant offers one of the same three-course meals and a four-course meal as well.
- On Wednesday, the restaurant offers just one meal—the four-course meal from the day before.
- On Thursday, the restaurant offers a five-course meal.
- On Friday, it offers a different five-course meal.

```
1 menu(monday,three(calamari,beef,shortcake))
2 menu(monday,three(puree,beef,icecream))
3 menu(tuesday,three(puree,beef,icecream))
4 menu(tuesday,four(consomme,greek,lamb,baklava))
5 menu(wednesday,four(consomme,greek,lamb,baklava))
6 menu(thursday,five(vichyssoise,caesar,trout,chicken,tiramisu))
7 menu(friday,five(vichyssoise,green,trout,beef,souffle))
```

Example 4: Food World (3/3)

→Note that, although there are constructors here, the dataset is finite in size. In fact, **there are strong restrictions on what sentences make sense**. For example:

- **Only symbols representing days of the week appear as the first argument of the menu relation.**
- **Only symbols representing foods appear as arguments in compound names.**
- and, **only whole meals appear as the second argument of the** `menu` **relation.**

→Note also that compound **names are not nested** here. **These kinds of restrictions are common in datasets**. Later in the course, we show how we can formalize these constraints.

Outline

Reformulation (1/3)

No matter how we choose to **conceptualize the world**, **it is important to realize that there are other conceptualizations as well**.

- **There need not be any correspondence** between the objects, functions, and relations in one conceptualization and the objects, functions, and relations in another.

In some cases, **changing one's conceptualization of the world can make it impossible to express certain kinds of knowledge**. **For example:**

- **The view of light as a wave phenomenon**
- **The view of light in terms of particles.**

In other cases, **changing one's conceptualization can make it more difficult to express knowledge, without necessarily making it impossible**.

Reformulation (2/3)

This **raises the question** of **what makes one conceptualization more appropriate than another**.
→Currently, **there is no comprehensive answer to this question**. **However, there are a few issues that are especially noteworthy**:

- The grain size **of the objects associated with a conceptualization**.
  →Choosing **too small** a grain can make knowledge formalization prohibitively boring.
  →Choosing **too large** a grain can make it impossible.

Reformulation (3/3)

Indistinguishability abstraction is **a form of object reformulation** that **deals with grain size**.

- If **several objects** mentioned in a dataset **satisfy all of the same conditions**, under appropriate circumstances, **it is possible to abstract the objects to a single object that does not distinguish the identities of the individuals**.
  $\rightarrow$**This can decrease the cost of processing queries**.

Outline

Key Takeaways

Key Takeaways

- A **vocabulary** consists of **symbols**, **constructors**, and **predicates**.
- A **ground term** is either a **symbol** or a **compound name** formed from a constructor and ground terms.
- The **Herbrand Universe** is the **set of all ground terms** that can be formed from a vocabulary.
- A **fact** is formed from a predicate and ground terms.
- The **Herbrand Base** is the **set of all possible facts** that can be formed from a vocabulary.
- A **dataset** is a **collection of facts** that characterizes the state of an application; formally, it is any subset of the Herbrand Base.
- The **closed-world assumption** states that **facts not included in a dataset are assumed false**.
- **Different datasets represent different states of the world**.
- **Changing one's conceptualization of the world can affect the ease or even the possibility of expressing knowledge about that world**.

Outline

Exercise 2.1

Consider the **Sorority World** introduced in subsection 1.
Write out a dataset describing a state in which every girl likes herself and no one else.

Solution 2.1

```
likes(abby,abby)
likes(bess,bess)
likes(cody,cody)
likes(dana,dana)
```

Exercise 2.2

Consider a variation of the **Sorority World** example in which we have a **single binary relation**, called `friend`.
`friend` differs from `likes` in two ways:

- It is **non-reflexive**, i.e., a girl cannot be friends with herself.
- It is **symmetric**, i.e., if one girl is a friend of a second girl, then the second girl is friends with the first.

Write out a dataset describing a state that satisfies the non-reflexivity and symmetry of the `friend` relation and so that exactly six friend facts are true.
→Note that there are multiple ways in which this can be done.

Solution 2.2

```
1 friend(abby,bess)
2 friend(bess,abby)
3 friend(abby,cody)
4 friend(cody,abby)
5 friend(bess,dana)
6 friend(dana,bess)
```

Exercise 2.3

A person x is a `sibling` of a person y if and only if x is a brother or a sister of y.
Write out the `sibling` facts corresponding to the `parent` facts shown below:

```
parent(art,bob)
parent(art,bea)
parent(bob,cal)
parent(bob,cam)
parent(bea,coe)
parent(bea,cory)
```

Solution 2.3

```
1 sibling(bob,bea)
2 sibling(bea,bob)
3 sibling(cal,cam)
4 sibling(cam,cal)
5 sibling(coe,cory)
6 sibling(cory,coe)
```

Exercise 2.4

Consider a world with n symbols and a **single binary predicate**.
How many distinct facts can be written in this language?

$$n, 2n, n^2, 2^n, 2^{n^2}, 2^{2^n}$$

Solution 2.4

There are n^2 distinct facts that can be written in this language.

- Each fact is formed from the binary predicate and two ground terms.
- Each ground term can be any of the n symbols.
- Thus, there are n choices for the first term and n choices for the second term, yielding a total of $n \times n = n^2$ distinct facts.

Outline

Warning

Homework 2.1, 2.2 and 2.3 should be submitted on Blackboard before the next course session!

Homework 2.1

Consider a variation of the **Sorority World** example in which we have a **single binary relation**, called `younger`.
`younger` differs from likes in three ways:

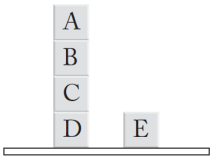
- It is **non-reflexive**, i.e., a girl cannot be younger than herself.
- It is **antisymmetric**, i.e., if one girl is younger than a second, then the second is not younger than the first.
- It is **transitive**, i.e., if one girl is younger than a second and the second is younger than a third, then the first is younger than the third.

Write out a dataset describing a state that satisfies the reflexivity, antisymmetry, and transitivity of the `younger` relation and so that the maximum number of younger facts are true. (Write and save your dataset on Sierra IDE using Epilog language)

→Note that there are multiple ways in which this can be done.

Homework 2.2

Consider the state of the **Blocks World** pictured below.
Write out all of the above facts that are true in this state. (Write and save your dataset on Sierra IDE using Epilog language)



Homework 2.3

Consider a world with n symbols and a **single binary predicate**.
How many distinct datasets are possible for this language?

$$n, 2n, n^2, 2^n, 2^{n^2}, 2^{2^n}$$

End of Chapter 2