Intelligent Drug Reasoning through Ontologies and Logic-Based Dialogue

Vito Nicola Losavio

In the context of modern medicine, both healthcare professionals and patients critically need access to reliable information on drugs, diseases, and their interactions. However, pharmaceutical data is often distributed across multiple sources in a variety of formats that lack an explicit semantic structure. This hinders the ability to perform automatic reasoning and logical deductions, as well as providing personalised assistance in real time.

Adopting an ontology in the pharmacological domain allows information on drugs, diseases, therapeutic indications, contraindications, and interactions between active ingredients to be represented in a structured, explicit, and formally interpretable way. Integrating this ontology with a rule engine (*Prolog*) and a conversational interface (*chatbot*), for example, enables the development of an intelligent system that can reason, answer user questions and support informed decision-making.

The developed ontology, supported by a reasoning engine based on declarative logic and a conversational interface, has a variety of meaningful yet non-clinical applications. Firstly, it can provide the average user with valuable information to help them understand how to use drugs correctly and identify possible risks associated with taking several products together. Secondly, it could be useful in professional contexts, for example as a tool for verifying drug compatibility or as a prototype in educational or analytical environments.

It is important to emphasise that this system is not intended to prescribe treatments or replace clinical judgement. The ultimate responsibility for any therapeutic decision always rests with a qualified medical professional. This tool is intended to provide intelligent support, helping users — whether patients, students or researchers — to explore and understand the field of pharmacology through a structured, explainable and interactive approach.

1 Definition of ontology

The ontology developed for this project is designed to represent knowledge of the pharmacological domain in a structured and queryable manner. It is organised around three main entities — **Person**, **Drug** and **Disease** — and a set of semantic relationships that facilitate the description of interactions between drugs, diseases and patients. The structure is formalised according to the GraphBRAIN [1] formalism and comprises a system of attributes and references that facilitate integration with reasoning engines.

1.1 Entity: Person

The **Person** entity represents an individual who may be affected by a disease and who may be taking one or more drugs. It is characterised by a unique identifier (ID), which is mandatory, while the other attributes, such as name, gender, date of birth and blood group, are optional in line with privacy requirements and use cases where the user's data are not known or needed.

1.2 Entity: Drugs

The **Drugs** entity represents a drug and is identified by its trade name and the active ingredient it contains. The ontology allows further details, such as the ATC code, the manufacturer, the route of administration, and the equivalence group code, to be specified. This information enables drugs to be categorised by therapeutic class and linked to equivalent products, providing a useful basis for reasoning about compatibility and treatment.

1.3 Entity: Disease

The **Disease** entity represents a disease or clinical condition. It includes a mandatory name and optional attributes that describe its symptoms, causes, and clinical category. The available categories cover a broad spectrum, including infectious, genetic, degenerative, psychiatric, cardiovascular and paediatric diseases, among others. This provides a classification system that is useful for both logical reasoning and user interaction.

1.4 Relationships

The ontology comprises a coherent set of semantic relationships, each of which is formalised with a subject, an object, and optional attributes. The main ones are:

- **prescribedTo/hasPrescription**: links a drug to the person taking it, including attributes such as start date, frequency and dosage.
- **hasDisease/hasPatient** links a person to a diagnosed disease and can indicate the disease's clinical state (active, in remission, chronic, etc.).
- **treatedWith/treats** links a disease to the drugs used in its treatment and can specify whether it is the reference therapy.
- **contraindicatedFor/contraindicates:** represents a clinical incompatibility between a drug and a disease, indicating that the drug should not be administered if the condition is present. This relationship can be assigned a severity level (from mild to critical) and may include notes that justify the contraindication.

These relationships enable the system to draw conclusions, such as *"Can this patient take this drug?"* and *"Which drugs should not be administered together?"*.

Of course, this system is not intended to replace medical judgement; the final decision always rests with the physician. However, it can serve as a valuable support tool, providing instant, structured insights that may help users to understand potential drug interactions or treatment incompatibilities quickly.

2 Formalisation of the ontology in first-order logic

Once the pharmacological domain ontology has been defined, the next step is to formalise the knowledge in logical terms to enable interpretation and manipulation by a deductive engine. First-Order Logic (FOL) is used to represent entities, attributes, and semantic relationships between domain objects, such as people, drugs, and diseases, according to a structured, formally consistent model.

To ensure a compact, extensible representation, a convention is adopted that identifies predicates associated with multi-argument data predicates. Each entity is identified via a unary predicate (e.g. $person(1)$), while the associated information is grouped into an n-ary predicate (e.g. $person\_data(1, marco, \ldots, \ldots)$) encapsulating its main attributes. This scheme maintains a clear, modular, queryable logical structure, facilitating the separation of logical identity and descriptive content.

The objective of this formalisation is to encode knowledge in a way that can be read by a logical system and to prepare for subsequent stages of automatic reasoning, consistency verification and conversational interaction without ever losing adherence to the semantic constraints defined by the original ontology.

As specified in the ontology, the **Person** entity represents an individual who is identified by the mandatory *id* attribute and described by optional attributes such as *name*, *surname*, *gender*, *birthDate* and *bloodGroup*. In first-order logic (FOL), each person is modelled as an object *x*, where $person(x)$ denotes their membership of the class of persons.

An n-ary predicate is defined for each individual identified by x, which aggregates the main attributes provided in the ontology.

The general structure of the formalisation is as follows:

$$\exists x(person(x) \wedge person\_data(x, name, surname, gender, birthDate, bloodGroup)) \tag{1}$$

This representation distinguishes between the logical identity of the individual and its descriptive properties while maintaining semantic coherence with the ontological definition. This allows simpler relations to be realised by using IDs instead of objects.

To ensure semantic consistency with the reference ontology, certain attributes have explicit constraints defined on their domains in the form of closed enumerative sets. In particular:

- The *gender* attribute is constrained to a discrete domain of values only.

$$gender \in \{M, F\} \tag{2}$$

- The *bloodGroup* attribute is restricted to the following values:

$$bloodGroup \in \{A^+, A^-, B^+, B^-, AB^+, AB^-, 0^+, 0^-\} \tag{3}$$

For example, an individual identified as Marco Longo (male; born 20 May 1994; blood group O positive) will be described as follows:

$$\exists x(person(x) \wedge person\_data(x,' Marco',' Longo',' male',' 1994 - 05 - 20',' O+')) \tag{4}$$

If one or more attributes are unavailable, the logic model will adopt a *empty string* to represent the absence of information. This is permitted by the ontological definition of optional values and may take the form of a symbolic value, such as null or missing. This approach maintains the syntactic completeness of the n-ary predicate without compromising the system's formal consistency.

As regard the **Disease** entity, each instance is identified by a mandatory attribute name and can be further described by optional attributes, such as a description, symptoms, causes and category. The latter takes predefined values that can be selected from a set of predefined clinical categories (e.g. metabolic, autoimmune, reproductive, etc.).

In the context of first-order logic, each disease is modelled as an object z for which $disease(z)$ is true, indicating that z belongs to the disease class. As with the Person entity, a structured predicate is introduced that aggregates the information attributes according to the schema provided by the ontology.

The general logical structure of the formalisation is as follows:

$$\begin{aligned}\exists z(disease(z) \wedge disease\_data(&\\ &z, name, description,\\ &symptoms, causes, category,\\ &chronic, rare, hereditary\\ ))&\end{aligned} \tag{5}$$

This representation collects all the relevant descriptive elements of the disease in a single logical expression while maintaining the distinction between the logical identifier (z) and the semantic content.

As mentioned previously with regard to the Person entity, an explicit *empty string* will be used here to indicate that disease data has not been entered, in addition, there is an attribute whose domain is restricted to a portion of possible values, and is defined as follows:

$$\begin{aligned} category \in \{&infectious, genetic,\\ &autoimmune, metabolic,\\ &neoplastic, degenerative,\\ &psychiatric, traumatic,\\ &cardiovascular, respiratory,\\ &gastrointestinal, neurological,\\ &dermatological, endocrine,\\ &hematologic, musculoskeletal,\\ &urological, ophthalmologic,\\ &otolaryngologic, reproductive,\\ &pediatric, geriatric\} \end{aligned} \tag{6}$$

Finally, the **Drugs** entity represents a drug and is identified by the mandatory *drug* attribute (trade name). It is also described by other attributes, such as *activeIngredient* (active ingredient), *ATC* (ATC code), *company* (manufacturer), *usage* (route of administration), *equivalenceGroup-Code* and *notes*. Some of these attributes, such as ATC, company or notes, are optional and may therefore be absent from the data.

In first-order logic, each *drug* is represented by an object d such that $drug(d)$ denotes its belonging to the drug class. The information associated with the drug is collected in a structured predicate in accordance with the ontology and according to the following general form:

$$\begin{aligned} \exists y(drug(y) \wedge drug\_data(y, name,&\\ &activeIngredient, ATC, company, usage,\\ &equivalenceGroupCode, notes\\ &))\end{aligned} \tag{7}$$

This formalisation binds a logical identifier (d) to the main drug information while maintaining the separation between the concept's identity and its descriptive attributes.

The $usage$ attribute has a limited domain defined as follows:

$$\begin{aligned} usage \in \{&oral, sublingual, intravenous, intramuscular,\\ &parenteral, parenteralAndOral, rectal\} \end{aligned} \tag{8}$$

The relationships defined in the ontology represent the semantic links between the domain's main entities: drugs, diseases and people. In the logical formalisation, each relationship is represented by an n-ary predicate, which includes references to the entities involved, as well as any attributes associated with the relationship. This structured approach enables knowledge to be expressed compactly and consistently with ontological semantics while retaining the ability to handle missing values with an empty string.

The relationships described in the ontology are defined as follows:

- **prescribedTo**: This relationship links a drug to the person to whom it has been prescribed. It can be enriched with optional attributes, such as the start and end dates of administration, dosage, and frequency.

  This relationship is formalised via an n-ary predicate of the form:

$$\begin{aligned}&\exists x \exists y(\\ &\quad person(x) \wedge drug(y) \wedge \\ &\quad prescribed\_to(y, x, startDate, endDate, dosage, frequency \\ &))\end{aligned} \tag{9}$$

- **hasPrescription**: this relationship represents the inverse of *prescribedTo*, linking a person to the drug they have been prescribed. It includes optional attributes such as the start and end dates of administration, the prescribed dosage, and the frequency of intake.

  This relationship is formalised via an n-ary predicate of the form:

$$\begin{aligned}&\exists x \exists y(\\ &\quad person(x) \wedge drug(y) \wedge \\ &\quad has\_prescription(x, y, startDate, endDate, dosage, frequency \\ &))\end{aligned} \tag{10}$$

- **hasDisease**: this relationship links a person to a diagnosed disease. Optional attributes can be added, such as the date of diagnosis and the clinical status of the disease (e.g. active, chronic or in remission). Boolean flags can also be added to indicate whether the disease is chronic, rare or hereditary.

  This relationship is formalised via an n-ary predicate of the form:

$$\begin{aligned}&\exists x \exists z(\\ &\quad person(x) \wedge disease(z) \wedge \\ &\quad has\_disease(x, z, diagnosisDate, status \\ &))\end{aligned} \tag{11}$$

  $status$ presents a limited domain as follows:

$$\begin{aligned}status \in \{&active, inactive, remission, cured, terminal, \\ &chronic, acute, relapsing, progressive, stable, advanced\}\end{aligned} \tag{12}$$

- **hasPatient**: This relationship is the inverse of 'hasDisease' and connects a diagnosed disease with the person affected by it. This allows the disease entity to retain information about its clinical context in specific individuals. Optional attributes may include the date of diagnosis, the current status of the disease (e.g. active, chronic or in remission) and Boolean indicators, such as whether the disease is chronic, rare or hereditary, in the context of that patient.

  It is defined as follows:

$$\begin{aligned}&\exists z \exists x(\\ &\quad disease(z) \wedge person(x) \wedge \\ &\quad has\_patient(z, x, diagnosisDate, status \\ &))\end{aligned} \tag{13}$$

- **treats**: this relationship links a drug to the disease it is used to treat. Optionally, it can specify whether the treatment represents the standard of care for that condition via a Boolean attribute.

  This relationship is formalised via an n-ary predicate of the form:

$$\exists y \exists z (drug(y) \land disease(z) \land treats(y, z, standardOfCare)) \tag{14}$$

- **treatedWith**: this relationship is the inverse of *treats*, linking a disease to the drug used to treat it. As with its counterpart, the optional boolean attribute *standardOfCare* can be included to indicate whether the drug is part of the standard therapeutic protocol for that disease.

  This relationship is formalised via an n-ary predicate of the form:

$$\exists z \exists y (disease(z) \land drug(y) \land treated\_with(z, y, standardOfCare)) \tag{15}$$

- **contraindicatedFor**: this relation indicates that a drug should not be administered in the presence of a specific disease, due to clinical contraindications.

  This relationship is formalised via an n-ary predicate of the form:

$$\exists y \exists z (drug(y) \land disease(z) \land contraindicated\_for(y, z, severity, notes)) \tag{16}$$

  $severity$ presents a limited domain as follows:

$$severity \in \{mild, moderate, severe, critical\} \tag{17}$$

- **contraindicates**: the inverse relation of *counterindicatedFor* has the same attributes as the *counterindicatedFor* relation and is defined as follows:

$$\exists z \exists y (disease(z) \land drug(y) \land contraindicates(z, y, severity, notes)) \tag{18}$$

3 From FOL to Prolog definition.

In this section, we convert the ontology formalisation from first-order logic (FOL) to Prolog. All entities and relationships are defined as facts, alongside a set of rules that enable logical reasoning.

3.1 Entity

The definitions of the three main entities in Prolog are given below.

- **Person** is defined as two predicates:
  (a) $person/1$, this identifies the individual at a logical level through their unique identifier.
  (b) $person\_data/6$, describes the individual's attributes.

- **Disease** is defined using two predicates:
  (a) $disease/1$: represents the logical identity of a disease.
  (b) $disease\_data/9$: captures the descriptive attributes of the disease.

- **Drug** is also defined by two predicates:
  (a) $drug/1$ identifies the drug as an entity.
  (b) $drug\_data/7$ provides its descriptive properties.

3.2 Relationships

This section outlines the relationships between the core entities of the ontology. Each relationship is represented as a Prolog predicate linking two or more entities and capturing how individuals interact or relate within the domain. Some relations are binary (e.g. a person has a disease), while others include additional parameters, such as severity, diagnosis date or clinical status.

The unique identifier identifies the individual at the logical level.

- **hasDisease** It is a predicate that links a person to a diagnosed disease and includes clinical information that enriches this relationship.

  The predicate for this relationship is $has\_disease/5$.

- **hasPatient** It is the inverse predicate of 'hasDisease', linking a diagnosed disease to the person affected by it. This relationship reflects the same clinical information and can be used to query the ontology from a disease-specific perspective.

  The predicate for this relationship is $has\_patient/5$.

- **prescribedTo** A predicate links a drug to the person to whom it has been prescribed. Additional information may be included, such as the date of the prescription, dosage and duration.

  The predicate for this relationship is $prescribed\_to/6$.

- **hasPrescription** It is the inverse predicate of *prescribedTo*, connecting a person to the drug they have been prescribed and providing relevant prescription details.

  The predicate for this relationship is $has\_prescription/6$.

- **treats** A predicate links a drug to the disease it is used to treat. This relationship expresses a therapeutic indication and may optionally indicate whether it is a standard treatment.

  The predicate for this relationship is $treats/3$.

- **treatedWith** It is the inverse predicate of *treats* and links a disease to the drug used to treat it. It provides the same therapeutic context from the perspective of the condition being treated.

  The predicate for this relationship is $treated\_with/3$.

- **contraindicatedFor** It is a predicate that links a drug to a disease for which it is contraindicated. This relation represents a clinical incompatibility and can include a severity level and an optional explanatory note.

  The predicate for this relationship is $contraindicated\_for/4$.

- **contraindicates** It is the inverse predicate of *contraindicatedFor*, and links a disease to the drug that should not be administered in its presence. It carries the same clinical implications and metadata.

  The predicate for this relationship is $contraindicates/4$.

3.3 Rules

In this section, we define the logical rules that enable new facts to be inferred from existing knowledge. These rules combine the earlier definitions of entities and relationships to capture domain-specific reasoning. Examples include identifying unsafe prescriptions, determining therapeutic indications and inferring contraindications based on patient conditions.

3.4 Reasoning based on disease

These rules allow you to check which drugs are used to treat a disease and which clinical categories it belongs to.

Standard treatment of a disease: A treatment is standard if marked as such:

$$standard\_treatment(Drug, Disease) :- treats(Drug, Disease, true). \tag{19}$$

Non standard treatment: Inverse of standard treatment:

$$non\_standard\_treatment(Drug, Disease) :- \backslash + standard\_treatment(Drug, Disease). \tag{20}$$

Has treatments: Check whether a disease has at least one standard treatment:

$$has\_treatment(Disease) :- standard\_treatment(\_, Disease). \tag{21}$$

3.5 Reasoning based on person

These rules help to determine whether a drug is safe for a patient based on their medical condition.

Contraindicated For Person: A drug is contraindicated for a patient if they have a disease for which the drug is contraindicated.

$$\begin{aligned} &contraindicated\_for\_person(Person, Drug) :- \\ &\qquad has\_disease(Person, Disease, \_, \_), \\ &\qquad contraindicated\_for(Drug, Disease, \_, \_). \end{aligned} \tag{22}$$

Safe For: A drug is considered safe if it is not contraindicated.

$$safe\_for(Person, Drug) :- \backslash + contraindicated\_for\_person(Person, Drug). \tag{23}$$

Get all non standard treatment: This rule enables the identification of drugs prescribed to a patient that are not standard treatments for their diagnosed disease. This allows for a critical evaluation of the administered therapies and is defined as follows:

$$\begin{aligned} &get\_all\_non\_standard\_treatment(Person, Drug, Disease) :- \\ &\qquad has\_disease(Person, Disease, \_, \_), \\ &\qquad prescribed\_to(Drug, Person, \_, \_, \_, \_), \\ &\qquad \backslash + standard\_treatment(Drug, Disease). \end{aligned} \tag{24}$$

Prescribed drug info : The $prescribed\_drug\_info/3$ predicate provides a combined view of prescription and drug metadata. It returns the drug ID, name and patient to whom the drug has been prescribed. The rule is defined as follows:

$$\begin{aligned} &prescribed\_drug\_info(Person, Drug, Name) :- \\ &\qquad prescribed\_to(Drug, Person, \_, \_, \_, \_), \\ &\qquad drug\_data(Drug, Name, \_, \_, \_, \_, \_). \end{aligned} \tag{25}$$

4 Chatbot

To interrogate the knowledge base and carry out reasoning based on it, a italian chatbot was developed to facilitate interaction between the knowledge base and the user, making it more fluent. To achieve this, a Prolog-based logical inference system was integrated with an LLM-based language module. The chatbot can understand natural language, categorise user requests, and provide personalised, informative, or prescriptive responses based on available data.

The chatbot uses an intent detection system based on language models (LLMs) that can understand and classify messages written in Italian. The $extract\_intent$ function analyses each user message and associates it with one of the system's provided actions (eg. "Trattamento applicabile", "Aggiungere patologie", "Conversazione generica"), thanks to a prompt engineering and local inference mechanism.

The LLaMA 3.1 8B[2] model is executed locally via Ollama[3] to guarantee the privacy and protection of clinical data and avoid sending it to external cloud services.

To take advantage of the customised functionalities, users must register, even if only partially, by providing at least a unique ID or name. The system automatically verifies the user's presence in the knowledge base. If the user is not present, the system proposes data entry. Only after this step can the user receive personalised answers based on their clinical profile. Furthermore, the user may choose to withhold clinical information, such as pathologies or medication. In this case, the chatbot will still be usable, but will only be able to provide general answers and facilitate a non-personalised conversation.

The chatbot offers a range of clinical and informative features that are automatically activated based on the intent identified in the user's message. The available actions are:

- **Medicina Incompatibile con la malattia**: Checks whether a mentioned drug is contraindicated with respect to a given disease.
- **Trattamento applicabile personalizzato** Verifies if a specific drug is appropriate for treating a disease, considering the clinical history of the registered user.
- **Trattamento applicabile**: It performs a general check to determine whether a drug is indicated for a disease, regardless of user-specific data.
- **Trattamento in conflitto personalizzato**: Identifies drugs or treatments that are not recommended for the user, based on their known conditions.
- **Lista Farmaci prescritti**: Retrieves the list of drugs that have been prescribed to the user currently in the knowledge base.
- **Elencare tutti i trattamenti non standard**: Lists all drugs prescribed to the user that are not considered standard treatments for the corresponding diseases.
- **Aggiungere farmaci prescritti**: Adds one prescribed drug at a time to the knowledge base, including attributes such as dosage, start and end dates, and frequency.
- **Aggiungere patologie**: Inserts a new disease associated with the patient, including optional clinical attributes such as diagnosis date and status.
- **Verificare esistenza trattamento per malattia**: Answers whether at least one standard treatment exists for a given disease, even in the absence of user-specific data.

- **Conversazione generica**: Handles open-ended or general questions, providing informative answers about drugs, diseases, or how the system works, especially when no clinical profile is available.

Regarding the functionality, *Medicina Incompatibile con la Malattia* enables the chatbot to determine whether a specific drug is incompatible with a given disease due to clinical contraindications. The system prompts the large language model (LLM) to extract two key entities from the user's message: the full name of the disease (even if it consists of multiple words) and the name of the drug. These are then used to query the knowledge base via the $contraindicated\_for/4$ relationship (3.2), which encodes drug–disease incompatibility relations.

If the drug is contraindicated for the extracted disease, the chatbot returns a warning message indicating that the treatment is incompatible with the patient's condition. Otherwise, it confirms that the treatment is compatible. If either entity cannot be found in the knowledge base, the system returns a message to this effect.

The *Trattamento applicabile personalizzato* functionality determines whether a specific drug is safe for a given user based on their diagnosed conditions. After extracting the drug name from the user's message and resolving its identifier, the system evaluates the Prolog $safe\_for/2$ rule (23), which checks that the drug is not contraindicated for any of the user's conditions.

If the rule succeeds, the chatbot confirms that the treatment is compatible. Otherwise, it warns the user about a possible conflict with their clinical profile. This check does not require the drug to have already been prescribed, enabling the system to be used for prevention or advice.

The *Trattamento applicabile* functionality enables the chatbot to verify whether a drug is recognised as a standard treatment for a specific disease, regardless of the user's clinical history. After extracting the names of the drug and disease from the message, the system performs a check using the $standard\_treatment/2$ rule (19), which returns true only when the drug is explicitly registered in the knowledge base as a standard therapy for the given condition.

Standard treatments are considered medically validated and safe in most clinical scenarios. Therefore, if the rule succeeds, the chatbot will respond to say that the drug is suitable for treating the disease. Otherwise, it informs the user that the treatment is not indicated according to the available medical knowledge.

This functionality is particularly useful for general enquiries and when the chatbot is being used without a personal clinical profile, or for an exploratory consultation.

Trattamento in conflitto personalizzato, this functionality verifies whether a drug is incompatible with the user's clinical profile due to medical contraindications. After extracting the drug name from the message and identifying the patient, the system checks whether the drug is explicitly contraindicated for any diseases that have been diagnosed in the user.

This is based on the $contraindicated\_for\_person/2$ rule (22 to determine if a conflict exists. If a match is found, the chatbot informs the user that the treatment is unsafe and should be avoided. Otherwise, the treatment is considered acceptable.

This personalised safety validation functionality is particularly useful in preventing potentially harmful prescriptions based on the user's medical history.

The *Lista Farmaci prescritti* functionality retrieves a list of all the drugs that have been prescribed to the user who is currently identified in the system. The system will use the rule $prescribed\_drug\_info/3$ (25) to retrieve all the information about a user's prescription and the readable name and ID for easier reading.

The resulting list is returned in a readable format, enabling users to review their current or previous prescriptions stored in the knowledge base. This functionality is only available to registered users for whom at least one drug has been added.

The *Elencare tutti i trattamenti non standard* functionality identifies drugs that have been prescribed to the user but are not considered standard treatments for their diagnosed conditions. It uses the previously defined $get\_all\_non\_standard\_treatment/3$ rule (24) to detect inconsistencies between prescriptions and established therapeutic guidelines.

The chatbot then returns a list of these treatments to support clinical review and decision-making, highlighting any potential discrepancies in the user's current therapy.

With, *Aggiungere farmaci* , we give to the users the opportunity to add new drug prescriptions to their clinical profile. This is essential for maintaining an up-to-date and personalised knowledge base, especially in a dynamic system where users interact with the chatbot over time.

As the knowledge base evolves as new users are added or existing users provide additional information, an automatic update mechanism has been implemented. This mechanism adds new prescription information while preserving existing data and attempting to avoid duplicates.

However, avoiding redundancy is not always guaranteed. Although the system applies strict formatting and identifier resolution rules, using a LLM for information extraction can result in different formulations of the same concept, particularly with free-form natural language input. Consequently, semantically equivalent facts may be recorded in different forms.

Similarly to drug prescriptions, the chatbot supports the *Aggiungere patologie* functionality, which enables users to add newly diagnosed diseases to their personal profile in the knowledge base. This process follows the same incremental update strategy described above: new facts are added without overwriting existing data and duplication is minimised as much as possible. During insertion, the system uses the LLM to extract the disease name, the date of diagnosis (if available) and the clinical status (e.g. chronic, active or in remission). These attributes are then validated against the previously defined ontology, ensuring that the values conform to the accepted set of disease statuses and concept definitions.

As with drug entries, using a generative model introduces variability in expression, which can result in semantic redundancies despite the presence of strict extraction and formatting rules.

Using the *Verificare esistenza trattamento per malattia* tool, we can determine whether a standard treatment is available for a disease mentioned by the user, regardless of any associated clinical data. This is particularly useful for answering general medical questions, such as whether a certain condition can be treated according to the knowledge base. To do this, the chatbot uses the previously defined rule $has\_treatment/1$ (21).

Ultimately, *Conversazione generica* (set when the request doesn't match any intent) enables the chatbot to handle open-ended or general-purpose interactions, such as questions about how the system works, the purpose of certain drugs or the meaning of medical terms. It is triggered when the user's message does not match any of the predefined actionable intents or when sufficient clinical data has not yet been provided for personalised reasoning.

In these cases, the chatbot relies solely on the language model to generate non-personalised, informative answers without accessing or querying the knowledge base. This ensures that the system remains accessible and functional, even for unregistered users or in the early stages of interaction.

5 Conclusion

Although the system performs reliably across a wide range of clinical and conversational queries, certain limitations must be acknowledged. Since natural language understanding relies on a generative language model, the user's intent may occasionally be misclassified. In such cases, the fallback mechanism triggers the generic conversation flow, enabling the interaction to continue uninterrupted.

Similarly, the extraction of key entities, such as disease and drug names, may sometimes be imprecise, particularly when the input is ambiguous, colloquial or poorly structured. Efforts have been made to mitigate this through prompt design and ontology-based validation, but such errors cannot be entirely eliminated.

Finally, it is important to note that the Prolog knowledge base used in this project contains only a subset of possible diseases, drugs and relationships.

References

[1] Stefano Ferilli and Domenico Redavid. The graphbrain system for knowledge graph management and advanced fruition. In *International Symposium on Methodologies for Intelligent Systems*, pages 308–317. Springer, 2020.

[2] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[3] Francisco S Marcondes, Adelino Gala, Renata Magalhães, Fernando Perez de Britto, Dalila Durães, and Paulo Novais. Using ollama. In *Natural Language Analytics with Generative Large-Language Models: A Practical Approach with Ollama and Open-Source LLMs*, pages 23–35. Springer, 2025.