Interactive Clinical Validation with ASP Chef in the RADIOAMICA Project

Mario Alviano, Pietro Macrì*, Luis Angel Rodriguez Reiners and Pierpaolo Sestito

University of Calabria, Department of Mathematics and Computer Science

Abstract

The integration of heterogeneous clinical data poses significant challenges in domains such as neuro-oncology, where diagnostic processes require the joint analysis of multiple dimensions, including anatomical location, histological type, imaging parameters, and genetic mutations. Ensuring consistency across these dimensions is a complex task, often prone to errors when performed manually. This paper presents an approach based on Answer Set Programming (ASP) to support the formal representation and automated analysis of multi-dimensional clinical data. Clinical records are encoded as logical facts, while domain knowledge is expressed through declarative rules and integrity constraints capturing medically relevant relationships. This reasoning process enables the automatic detection of inconsistencies across different clinical dimensions, such as incompatible combinations of histological types, imaging features, and mutational profiles. To support interpretability, the reasoning outcomes are integrated into a secure, client-side interactive dashboard powered by ASP Chef, combining tabular exploration with graph-based visualizations of detected inconsistencies, and providing structured insights into the underlying logical derivations. The implemented approach highlights the effectiveness of logic-based methods in improving data consistency, supporting transparent reasoning, and enabling the analysis of complex clinical datasets.

Keywords

Answer Set Programming (ASP), Clinical Data Integration, Inconsistency Detection, Explainable Reasoning, Visual Analytics, Declarative Knowledge Representation

1. Introduction

Answer Set Programming (ASP) is a declarative programming paradigm based on nonmonotonic logic and stable model semantics [1, 2, 3, 4, 5], particularly suited for modeling complex constraints and dependencies across heterogeneous data dimensions. ASP is especially well-suited for tackling problems that involve combinatorial search, reasoning under incomplete or uncertain information, and constraint satisfaction. It has been effectively applied across many knowledge-intensive domains [6, 7, 8, 9], and has also found different applications in the medical field [10, 11, 12, 13, 14, 15, 10, 16, 17, 18], mostly related to scheduling problems, queries explainability and treatment management automation. Despite these applications, the integration and analysis of multi-dimensional clinical data remain a challenge, particularly in highly specialized fields such as neuro-oncology, mainly due to the heterogeneity of data sources and the lack of automated mechanisms for cross-dimensional validation. Evaluating neurological lesions requires synthesizing information across multiple, distinct clinical axes. In our context, this data is modeled across four key dimensions: Anatomical Location, which specifies the lesion's occurrence in the nervous system; Histological Type, defining the cellular or tissue origin; Magnetic Resonance Imaging (MRI) Parameters (e.g., T2, FLAIR, diffusion, perfusion), capturing the imaging phenotype; and Mutational Profile (e.g., IDH, MGMT), which records critical genetic biomarkers. As the volume and complexity of this data grow, manually ensuring consistency across these four dimensions becomes prone to error, potentially leading to clinically inconsistent interpretations or suboptimal therapeutic decisions. For example, a specific histological type may be logically or biologically inconsistent with a reported mutational profile or a particular set of MRI parameters.

CILC 2026: The 41st Italian Conference on Computational Logic, June 23–25, 2026, Ferrara, Italy

*Corresponding author.

mario.alviano@unical.it (M. Alviano); pietro.macri@unical.it (P. Macrì); luis.reiners@unical.it (L. A. Rodriguez Reiners); pierpaolo.sestito@unical.it (P. Sestito)

0000-0002-2052-2063 (M. Alviano); 0009-0001-3434-0404 (P. Macrì); 0009-0000-1808-9910 (L. A. Rodriguez Reiners); 0009-0004-7685-6152 (P. Sestito)

To address this challenge, we propose an integrated architecture designed to automate reasoning over clinical data to ensure data integrity and detect inconsistencies. The core of the system is a secure, client-side interactive dashboard developed in Svelte that consolidates patient records in a structured, tabular view. The underlying logic engine is provided by ASP Chef [19], a versatile platform for declarative reasoning and visualization, within which ASP rules encode medical constraints across the four clinical dimensions. By leveraging URL hash fragments for data transfer, the architecture ensures that anonymized clinical data are processed strictly client-side without traveling over the network, thus preserving privacy. ASP Chef processes these rules against the patient data to automatically detect inconsistencies. Crucially, rather than simply outputting logical models, ASP Chef generates graph-based visual representations of the reasoning process. This visualization component allows clinicians to instantly identify not just that an inconsistency exists, but why it occurred, thereby enhancing the explainability of the automated reasoning and supporting more reliable validation of clinical data through the combined use of formal reasoning and visual analytics.

2. Background

This section introduces the formal foundations and modeling perspective underlying the proposed approach, focusing on ASP and the representation of multi-dimensional clinical data.

2.1. Answer Set Programming

An ASP program is a finite set of rules. Each rule typically has a head, representing a conclusion (which may be atomic or a choice), and a body, representing a set of conditions that must hold (a conjunction of literals, aggregates and inequalities). Formally, an ASP program Π induces a collection (zero or more) of answer sets (also known as stable models), which are interpretations that satisfy all the rules in Π while also fulfilling the stability condition (i.e., the models must be supported and minimal in a specific formal sense [20]). The intended output of a program can be specified using `#show` directives of the form

`#show` $p(\bar{t})$ `:` *conjunctive_query*`.`

Here, p denotes an optional predicate symbol, $\bar{t}$ is a (possibly empty) sequence of terms, and *conjunctive_query* is a conjunction of literals serving as a condition for displaying instances of $p(\bar{t})$. Answer sets are then projected accordingly. For details on syntax and semantics, including `#show` and other directives, we refer to the ASP-Core-2 standard format [21].

Example 1. *Consider a set of patient records to be checked for clinical consistency. Each patient is described by a histological type, an optional mutational profile, an age, and a diffusion MRI value. The goal is to automatically flag patients whose attribute combinations violate known medical constraints. The following ASP program combines* derivation rules *with* integrity constraints*:*

```
r1 :  violation(P, astro_idh) :- histology(P, astro_idh), age(P, A), A > 60.
r2 :  violation(P, astro_idh) :- histology(P, astro_idh), diffusion(P, D), D > 10.
r3 :  :- histology(P, gbm), mutation(P, idh1).
r4 :  :- histology(P, gbm), mutation(P, idh2).
r5 :  #show P : violation(P, _).
```

Rules r_1 and r_2 are derivation rules encoding clinical anomalies for IDH-mutant astrocytoma patients: r_1 flags patients older than 60, since this tumour class is atypically rare in that age group; r_2 flags patients with high diffusion restriction (value $>$ 10), which is imaging evidence of higher-grade behaviour inconsistent with the expected profile of this histological type. Rules r_3 and r_4 are integrity constraints encoding a hard classification rule from the WHO 2021 CNS classification: glioblastoma is defined as IDH-wildtype [22], so any co-occurrence with an IDH1 or IDH2 mutation is structurally invalid and yields no answer set. The `#show` directive in r_5 projects only the identifiers of patients with at least one detected violation. Suppose the following input facts describe the patient cohort:

```
histology(p1, astro_idh).  mutation(p1, idh1).  age(p1, 65).  diffusion(p1,  8).
histology(p2, astro_idh).  mutation(p2, idh1).  age(p2, 35).  diffusion(p2,  3).
histology(p3, gbm).                             age(p3, 70).  diffusion(p3, 12).
histology(p4, astro_idh).  mutation(p4, idh2).  age(p4, 42).  diffusion(p4,  5).
```

Patient p3 carries no IDH mutation, so the integrity constraints r_3 and r_4 are satisfied. Rule r_1 fires for p1 (IDH-mutant astrocytoma, age 65 > 60); no diffusion value in the cohort exceeds 10, so r_2 does not fire. Patients p2, p3, and p4 satisfy all encoded rules. The program yields the projected answer set p1. ■

2.2. ASP Chef

An *operation O* is a function receiving in input a sequence of interpretations and producing in output a sequence of interpretations. Operations may produce side outputs (e.g., a graph visualization) and accept parameters to influence their behavior. An *ingredient* is an instantiation of a parameterized operation with side output. A *recipe* is a tuple of the form (*encode*, *Ingredients*, *decode*), where *Ingredients* is a (finite) sequence $O_1\langle P_1\rangle, \ldots, O_n\langle P_n\rangle$ of ingredients, and *encode* and *decode* are Boolean values. If *encode* is true, the input of the recipe is mapped to `[[__base64__("s")]]`, where $s = Base64(s_{in})$ (i.e., the Base64–encoding of the input string s_{in}). After that, the ingredients are applied one after another. Finally, if *decode* is true, every occurrence of `__base64__(s)` is replaced with (the ASCII string associated with) $Base64^{-1}(s)$. Among the operations supported by ASP Chef there are $Encode\langle p, s\rangle$ to extend every interpretation in input with the atom `p("t")`, where $t = Base64(s)$; $Search\ Models\langle \Pi, n\rangle$ to replace every interpretation I in input with up to n answer sets of $\Pi \cup \{p(\bar{t}).\mid p(\bar{t}) \in I\}$; $Show\langle \Pi\rangle$ to replace every interpretation I in input with the projected answer set $\Pi \cup \{p(\bar{t}).\mid p(\bar{t}) \in I\}$ (where Π comprises only `#show` directives). $Optimize\langle \Pi, n\rangle$ to replace every interpretation I in input with up to n optimal answer sets of $\Pi \cup \{p(\bar{t}).\mid p(\bar{t}) \in I\}$.$JSON\ Path\ Plus\langle Q, P\rangle$ to replace every interpretation I in input with the interpretation obtained by applying each query in Q to Base64–encoded JSON documents in I, and mapping each match to a fact $p(\bar{t})$ according to the JSON-to-ASP transformation specified by P.

Example 2. *The inconsistency detection task from Example 1 can be addressed in ASP Chef by a recipe comprising a single* Search Models$\langle\{r_1, r_2, r_3, r_4, r_5\}, 1\rangle$. *Alternatively, a recipe separating detection from presentation would use two ingredients:* Search Models$\langle\{r_1, r_2, r_3, r_4\}, 1\rangle$ *to derive all violations, followed by* Show$\langle\{r_5\}\rangle$ *to project only the identifiers of patients with at least one flagged constraint.* ■

Several operations in ASP Chef support expansion of *Mustache templates* [23]; among them, there are *Expand Mustache Queries*, *@vis.js/Network* (to visualize graphs), *Tabulator* (to arrange data in interactive tables), and *Chart.js* (to produce different kinds of charts). A Mustache template comprises queries of the form `{{ Π }}`, where Π is an ASP program with `#show` directives—alternatively, `{{= p(t) : conjunctive_query }}` for `{{ #show p(t) : conjunctive_query. }}`. Intuitively, queries are expanded using one projected answer set of $\Pi \cup \{p(\bar{t}).\mid p(\bar{t}) \in I\}$, where I is the interpretation on which the template is applied on. Separators can be specified using the predicates `separator/1` (for tuples of terms), and `term_separator/1` (for terms within a tuple). The variadic predicate `show/*` extends a shown tuple of terms (its first argument) with additional arguments that enable repeating tuples in output and can be used as sorting keys (using predicate `sort/1`). Moreover, Mustache queries can use `@string_format(format, ...)` to format a string using the given format string and arguments, and floating-point numbers are supported with the format `real("NUMBER")`. Format strings can also be written as (multiline) *f-strings* of the form `{{f"..."}}`, using data interpolation `${`*expression*`:`*format*`}` to render *expression* according to the given *format*.

Example 3. *Recipes from Example 2 can be further extended to produce visualizations of the computed results. To this aim, the following Mustache template can be combined with* Chart.js *to obtain the bubble chart shown in Figure 1.*

```
{
  type: 'bubble',
  data: {
```

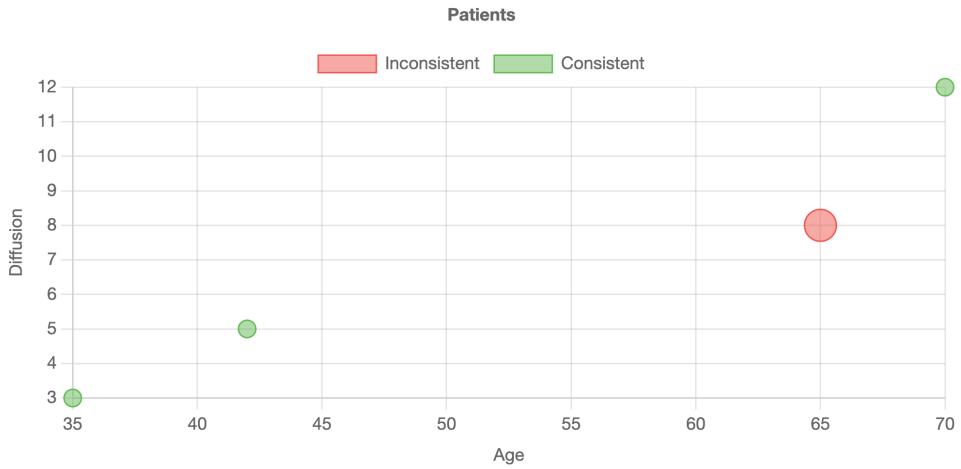


Figure 1: Bubble chart of the patient cohort from Example 1. Each bubble represents one patient; position encodes patient age (x-axis) and diffusion MRI value (y-axis), and colour distinguishes consistent patients (green) from inconsistent ones (red). For inconsistent records, the bubble radius is scaled proportionally to the severity of the violation, indicating how far the value exceeds the tolerated limit.

```
    datasets: [
      { label: "Inconsistent",
        data: [{{= {{f"{x: ${A}, y: ${D}, }"}}
                    : age(P,A), diffusion(P,D), violation(P,_) }}],
        borderColor: "#ff4444",
        backgroundColor: "rgba(255, 68, 68, 0.5)",
        radius: [{{= @expr({{f"6 + math.max(${A} - 60, ${D} - 10)"}})
                     : age(P,A), diffusion(P,D), violation(P,_) }}]
      },
      { label: "Consistent",
        data: [{{= {{f"{x: ${A}, y: ${D}, }"}}
                   : age(P,A), diffusion(P,D), not violation(P,_) }}],
        borderColor: "#44bb44",
        backgroundColor: "rgba(68, 187, 68, 0.5)",
        radius: 6
      }
    ],
  },
  options: {...},
}
```

Each Mustache query emits one coordinate pair [*age*, *diffusion*] *per patient, populating the respective series directly from the ground facts. A recipe implementing the detection and producing the chart shown in Figure 1 is available at https://asp-chef.alviano.net/s/CILC2026/bubble-chart#pit0500/asp-chef-short-links.* ■

3. Knowledge Representation of Neurological Lesions

To enable automated reasoning over heterogeneous clinical data, the raw multi-dimensional information must be transformed into a formal, declarative logic representation. The initial challenge in developing a robust clinical decision support system is handling the nested nature and the dimensionality of the raw input. In neuro-oncology, a single patient record encapsulates several interlinked attributes that span different medical domains, from basic demographics to advanced genetic profiling. This section covers two complementary aspects of that challenge: Section 3.1 describes the anatomy of the clinical data as it arrives from the backend, showing how each field maps to one of the four reasoning dimensions;

```
{
  "id": "5d7351c4-c1b6-4560-86a9-2a183efe2403",
  "data_esame": "2026-03-31T00:00:00.000Z",
  "eta": 90,
  "sede": "Lobulo cerebrale temporale",
  "istologia": "Sangue",
  "parametri_rm": {
    "t2": false,
    "flair": true,
    "mismatch": false,
    "diffusione": { "valore": 5 },
    "perfusione": { "valore": -83 }
  },
  "mutazioni": {
    "MGMT": false,
    "IDH1": true,
    "IDH2": false,
    "ATRX": false,
    "p53": true
  }
}
```


Figure 2: Example of a patient's clinical record formatted as a JSON payload

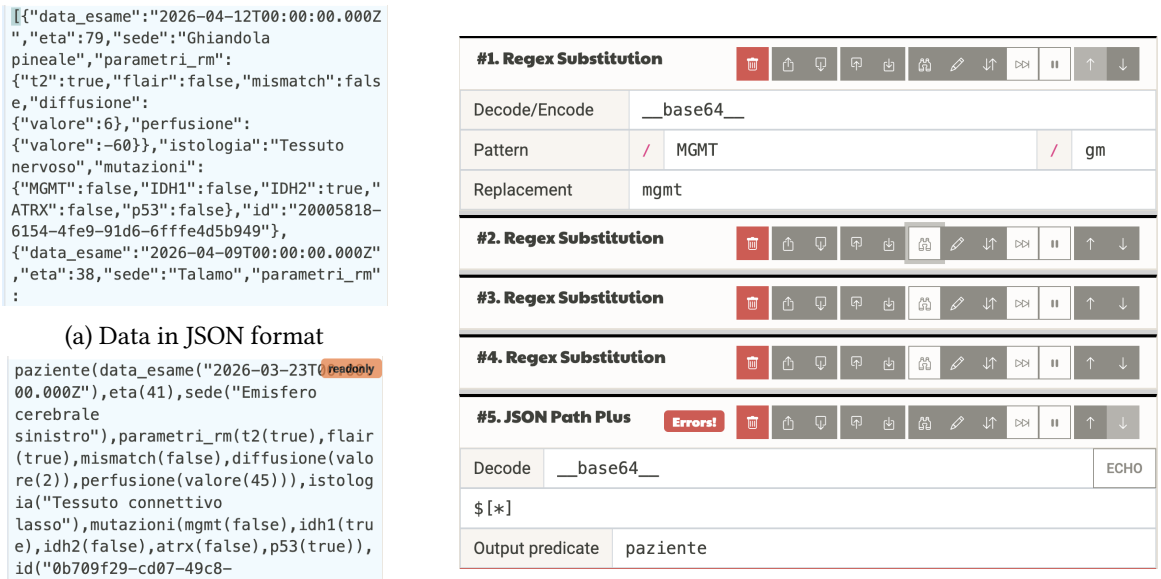
Section 3.2 then presents the transformation pipeline that converts those JSON records into flat ASP predicates ready for further reasoning. Our approach leverages ASP to carry out this transformation as a semantic restructuring: nested payloads are decomposed into first-order logic predicates, enabling the subsequent visualizations and application of integrity constraints across all clinical dimensions simultaneously.

3.1. Clinical Data Structure

In modern health information systems, data is predominantly exchanged via RESTful APIs using the JavaScript Object Notation (JSON) format. In our architecture, the system receives an array of JSON objects, where each object represents a comprehensive snapshot of a patient's neurological lesion. Before detailing the parsing methodology, it is crucial to understand the anatomy of the data. A typical patient record in our system is structured as shown in Figure 2. Each top-level field encodes one of the four clinical dimensions, or provides administrative metadata:

- **Anatomical Location:** the `"sede"` field records where in the nervous system the sample was taken (e.g., temporal lobe, brainstem).
- **Histological Type:** the `"istologia"` field identifies the cellular or tissue origin of the lesion (e.g., glioblastoma, astrocytoma).
- **MRI Parameters:** the nested object `"parametri_rm"` collects the imaging phenotype across five modalities: T2 signal (`"t2"`), FLAIR signal (`"flair"`), T2/FLAIR mismatch (`"mismatch"`), diffusion restriction (`"diffusione.valore"`), and perfusion index (`"perfusione.valore"`).
- **Mutational Profile:** the nested object `"mutazioni"` records the presence or absence of five critical genetic biomarkers: MGMT, IDH1, IDH2, ATRX, and p53 mutations.
- **Administrative/Demographic metadata:** `"id"` (unique patient identifier), `"data_esame"` (examination date), and `"eta"` (patient age) provide indexing and demographic context used in age-stratified constraints.

This structure is hierarchical and easily readable by web applications, but cannot be directly processed



(a) Data in JSON format

(b) Predicates representation

(c) Regex Configuration and JSON Path

Figure 3: The JSON-to-facts transformation pipeline: raw patient record in JSON (top left), resulting ASP predicates (bottom left), and the five-ingredient recipe configuration in ASP Chef (right).

by a flat logic solver. The nested objects for MRI parameters and mutational profiles must be decomposed into atomic predicates before any constraint evaluation can take place.

3.2. Data Ingestion and Flattening via ASP Chef

To bridge the gap between the structured JSON payloads described above and the ASP solver, we integrated a dynamic parsing mechanism within the *ASP Chef* pipeline composed of five sequential ingredients, as illustrated in Figure 3. The pipeline proceeds as follows.

Ingredients 1–4 (Regex Substitution). Before the JSON Path extraction can reliably map values to predicate arguments, the raw text output requires normalization. A sequence of four *Regex Substitution* ingredients is applied to perform a capital case to lower case transformation for the mutation values. This normalization step is convenient to ensure compatibility with ASP syntax, where constants and function names must begin with a lowercase letter to avoid being misinterpreted as variables by the solver.

Ingredient 5 (JSON Path Plus). After normalization, the *JSON Path Plus* ingredient navigates the JSON array using a path expression rooted at `$[*]` so that every element of the top-level array is treated as a separate patient record.

Example 4. *Consider the patient record in Figure 2. After the four normalization steps and the JSON Path extraction, the record is translated into the following ground fact:*

```
paziente( id("5d7351c4-c1b6-4560-86a9-2a183efe2403"),
  data_esame("2026-03-31T00:00:00.000Z"),
  eta(90), sede("Lobulo cerebrale temporale"), istologia("Sangue"),
  parametri_rm( t2(false), flair(true), mismatch(false),
    diffusione(valore(5)),perfusione(valore(-83)) ),
  mutazioni( mgmt(false),idh1(true),idh2(false),atrx(false),p53(true) ) ).
```

The integrity constraints and visualizations that will be introduced in Section 4 operate directly on these facts. ■

The resulting recipe is entirely self-contained: it accepts a JSON array as input and produces a normalized set of ASP facts as output with no external dependencies. Any other ASP Chef recipe that

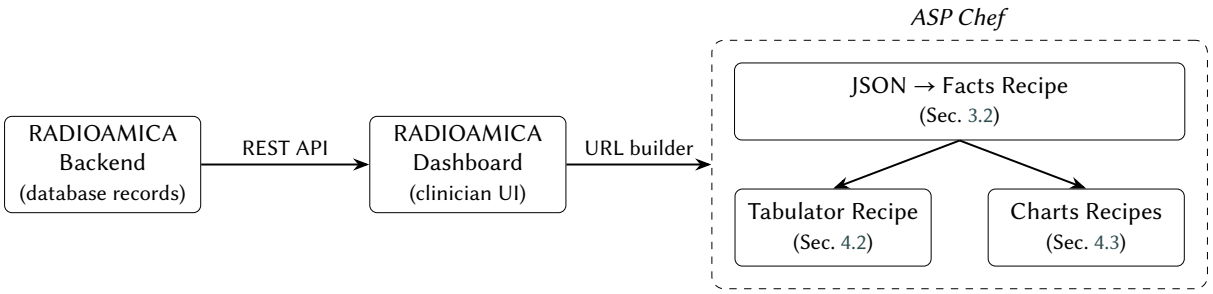


Figure 4: High-level architecture of the RADIOAMICA reasoning and visualization pipeline. The dashboard queries the backend and forwards the JSON payload to ASP Chef via a JavaScript-constructed URL. Inside ASP Chef every recipe first reuses the JSON-to-facts transformation (Section 3.2) and then applies its specific reasoning or visualization logic.

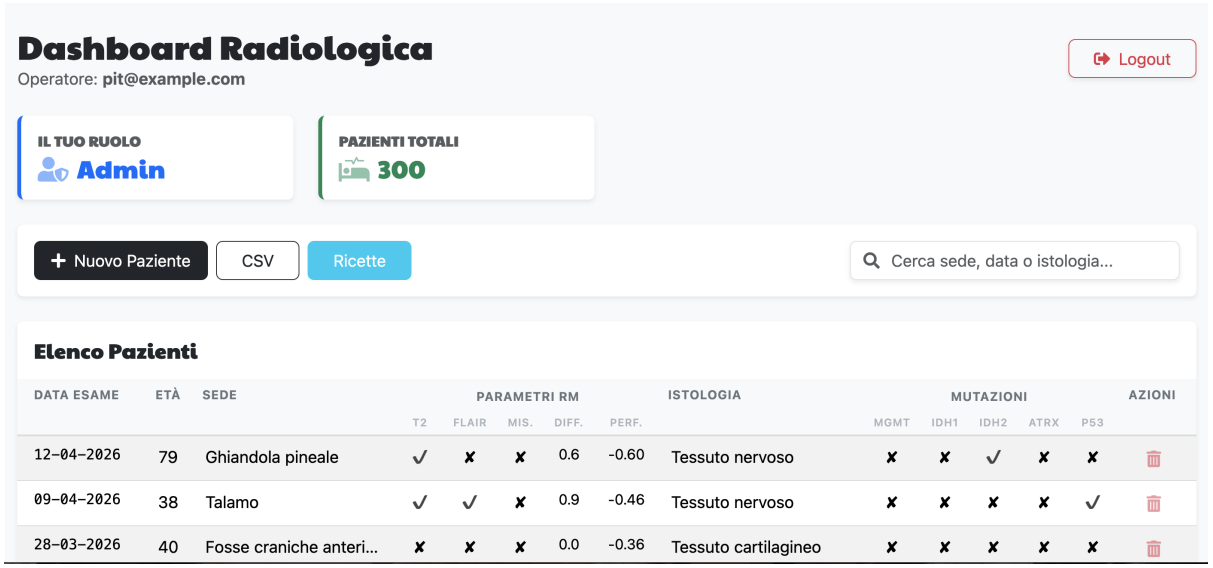


Figure 5: A screenshot of the web application's dashboard in which users are able to manage patients' records: add new ones, update existing ones, and delete them. Moreover, the button "Ricette" will redirect an user to ASP Chef recipes for the visualization tools.

processes the same clinical dataset can therefore embed this transformation by starting with a *Recipe* ingredient pointing to this recipe's short link (https://asp-chef.alviano.net/s/CILC2026/json-data-parse#pit0500/asp-chef-short-links). Downstream recipes receive the fact representation directly, without duplicating a single line of parsing logic.

4. System Design and Visual Integrations

The knowledge representation framework presented in the previous section is actualized through the architecture of the RADIOAMICA [24] platform (https://projects.dimes.unical.it/radioamica/). Clinical data, automated reasoning, and visual analytics are connected by a pipeline that links a RESTful backend to the *ASP Chef* reasoning engine. This section describes the data flow (Section 4.1), the tabular validation interface (Section 4.2), and the chart-based analytical dashboard (Section 4.3).

4.1. Architecture and Data Flow

The architecture presented in Figure 4 strictly decouples the data persistence layer from the reasoning and visualization layers. The end-to-end workflow proceeds in five steps.

1. **Data acquisition.** The clinician opens the RADIOAMICA Dashboard (Figure 5), which queries

```
export function compress(object: object): string {
        const json = JSON.stringify(object);
        const encoded = Base64.encode(json);
        const binData = pako.deflate(encoded);
        return Base64.fromUint8Array(binData);
}

export function packShortUrl(recipeUrl: string, theInput: string): string {
        const obj = compress({ input: theInput });
        if (recipeUrl.includes('#')) {
                return recipeUrl.replace('#', `#${obj};`);
        }
        return `${recipeUrl}#${obj};`;
}
```

Figure 6: JavaScript utility functions for URL construction. `compress` serializes the patient JSON payload, Base64-encodes it, and applies DEFLATE compression. `packShortUrl` injects the compressed payload into the target recipe URL, producing the full ASP Chef URL passed to the clinician's browser.

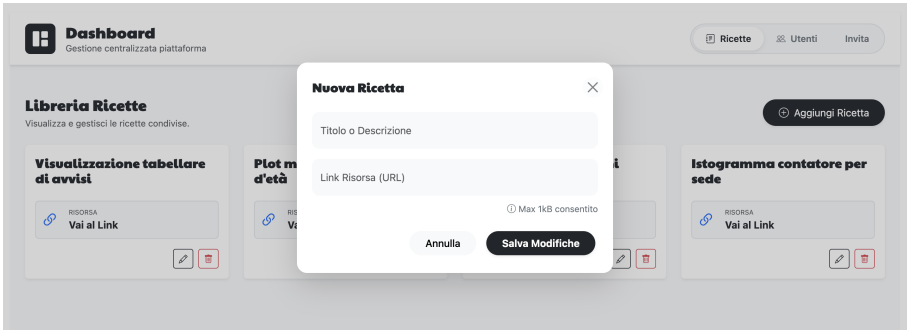


Figure 7: Administrators are the only ones to add new recipes to expand the set of visualization tools together with domain experts to improve clinicians' experiences. Since we are working with short links, the maximum size for the URL is 1 kilobyte, which is more than generous.

the backend service via a REST API call and receives a JSON array of patient records containing demographics, anatomical locations, MRI parameters, and mutational profiles.

2. **URL construction.** Rather than performing validation or visualization on the backend, the frontend hands the payload directly to ASP Chef. A pair of JavaScript functions constructs the full ASP Chef URL: a short-link base identifying the target recipe (e.g., https://asp-chef.alviano.net/s/CILC2026/tabella-avvisi#pit0500/asp-chef-short-links), followed by a # separator and the Base64-compressed JSON payload, and finally a ; separator followed by the GitHub username and repository credentials used by the short-link resolver (see Figure 6).
3. **Recipe invocation.** After an administrator adds a new recipe (Figure 7), opening the constructed URL loads the target recipe inside ASP Chef with the patient data already attached as input.
4. **Modular JSON-to-facts transformation.** Every recipe begins with a *Recipe* ingredient that embeds the self-contained transformation presented in Section 3.2. This ingredient receives the raw JSON array and outputs a normalized set of ASP ground facts, ensuring that all downstream reasoning and visualization logic operates on a consistent logical representation without duplicating any parsing code.
5. **Domain-specific reasoning and rendering.** The remaining ingredients of the recipe apply integrity constraints, aggregation rules, and Mustache templates to the normalized facts, producing the interactive UI components (Tabulator table, charts) that are displayed to the clinician in the browser.

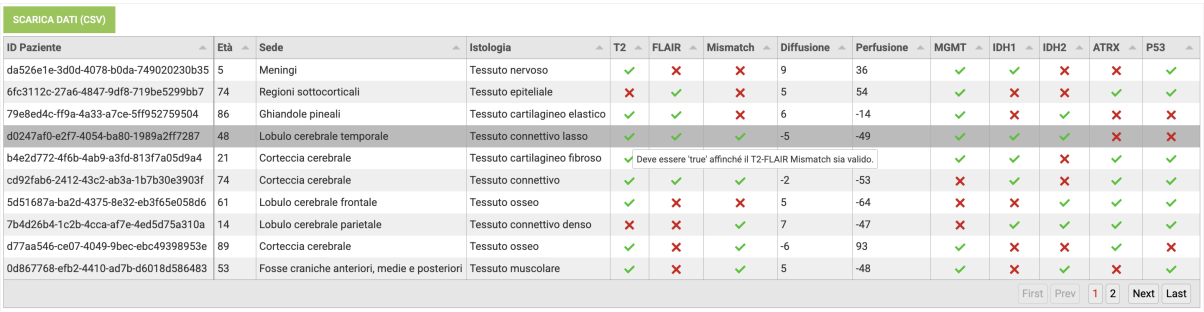


Figure 8: The interactive Tabulator interface generated via ASP Chef. The table lists patient records and employs visual indicators (green ticks and red crosses) to flag potential inconsistencies based on medical constraints. Hovering over a flagged row reveals a tooltip containing the constraint-grounded explanation produced by the ASP rules. https://asp-chef.alviano.net/s/CILC2026/tabella-avvisi#pit0500/asp-chef-short-links

4.2. Clinical Validation and Tabular Exploration

A critical requirement for clinical decision support systems is to avoid opaque, ”black-box” decision making. When dealing with complex neuro-oncological data, an automated system should not override a clinician’s judgment by rigidly discarding records deemed ”incorrect.” Instead, it must act as an advanced filtering system that highlights potential anomalies.

To address this, we leveraged ASP Chef’s integration with the `Tabulator` JavaScript library. Through Mustache queries, the ASP logic program dynamically generates the JSON configuration required to render an interactive data table. To achieve such a detailed table view, the following Mustache query is used:

```
{
    data: [
    {{= @string_format("{id: '%s', tipo: 'avviso', eta: %s, sede: '%s', istologia: '%s',
    t2: '%s', flair: '%s', mismatch: '%s', diffusione: %s, perfusione: %s, mgmt: '%s',
    idh1: '%s', idh2: '%s', atrx: '%s', p53: '%s'}",
         ID, Eta, Sede, Istologia, T2, Flair, Mismatch, Diffusione, Perfusione, MGMT,
    IDH1, IDH2, ATRX, P53)
       : dettaglio_avviso(ID, Eta, Sede, Istologia, T2, Flair, Mismatch, Diffusione,
    Perfusione, MGMT, IDH1, IDH2, ATRX, P53) }}
  ],
  columns: [...]
}
```

The `dettaglio_avviso/14` predicate triggers whenever ASP integrity constraints detect a violation of clinical logic. In the tabular view (Figure 8), these logical derivations are rendered as distinct visual cues—such as green ticks for expected values and red crosses for flagged attributes. Rather than enforcing rigid data rejection, these visual flags function as a decision support mechanism. They act as theoretical warnings to notify medical personnel that an unusual combination of attributes (e.g., a rare mutation in an atypical anatomical location) warrants expert review. To make this reasoning transparent, the interface maps these violations to cell-specific tooltips. By hovering over a flagged cell, the clinician is immediately presented with the explicit biological or technical constraint that was violated, embedding explainability directly into the workflow and providing real-time justification for every detected inconsistency.

4.3. Visual Analytics via Mustache Queries

Beyond tabular exploration, the volume and dimensionality of the clinical data motivate additional analytical views that allow domain experts to study cohort distributions and surface lateral correlations. Each visualization recipe in the RADIOAMICA dashboard begins with a *Recipe* ingredient that embeds the JSON-to-facts transformation from Section 3.2, ensuring that all charts operate on the same normal-

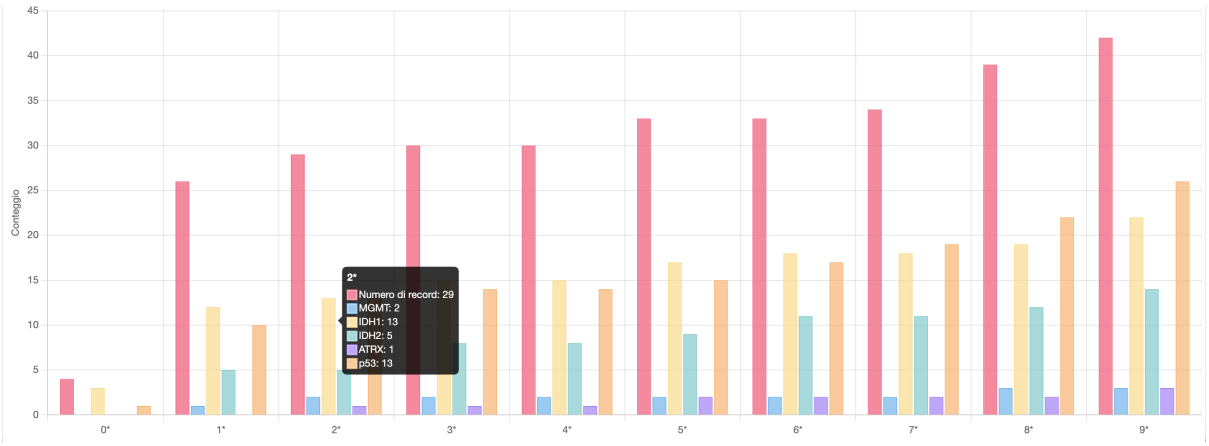


Figure 9: Bar chart visualizing the distribution of patient records and specific genetic mutations (MGMT, IDH1, IDH2, ATRX, p53) across different age decades, generated using Chart.js via Mustache templates. https://asp-chef.alviano.net/s/CILC2026/eta-plot#pit0500/asp-chef-short-links

ized logical representation as the validation table. The subsequent ingredients add aggregation rules and Mustache templates targeting specific JavaScript plotting libraries (e.g., `Chart.js` and `ApexCharts`). Here we showcase three specialized recipes:

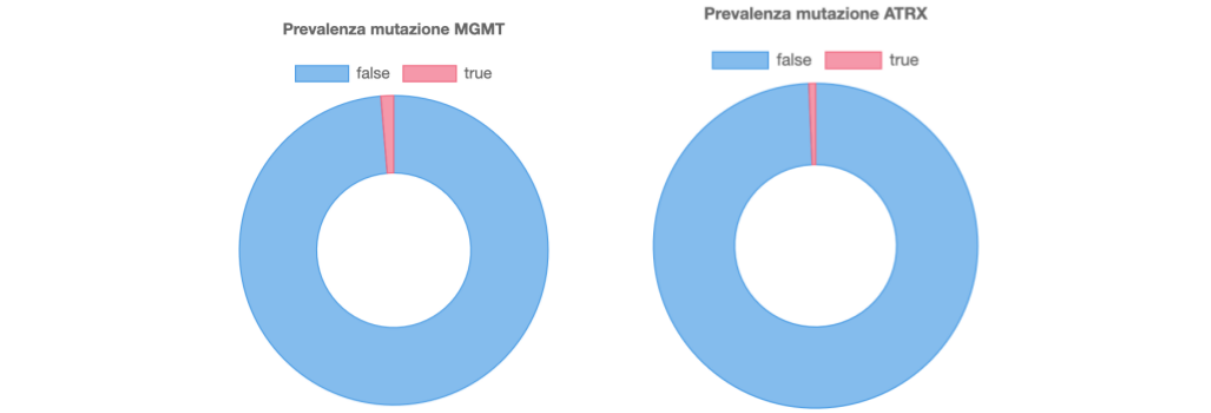
1. **Age and Mutation Distribution:** A bar chart (Figure 9) stratifies the frequency of each mutational profile across patient age decades. The ASP solver computes the aggregates from the ground facts, and the Mustache template populates the `Chart.js` dataset configurations. This view enables clinicians to identify age-dependent genetic trends, for instance spotting whether IDH mutations cluster disproportionately in younger cohorts as expected by current neuropathological knowledge.
2. **Mutation Prevalence:** Donut charts (Figure 10a) display the true/false ratio for each of the five critical biomarkers. The immediate visual summary supports rapid assessment of how widespread a particular mutation is in the current dataset, which in turn informs which integrity constraints are most active in the validation step.
3. **Anatomical Location Frequency:** A histogram (Figure 10b) ranks brain regions by the number of observed lesions, sorted by descending frequency. This distribution highlights which anatomical sites dominate the dataset, providing context for interpreting the constraint violations surfaced by the Tabulator.

By combining a shared parsing layer with purpose-specific reasoning and visualization ingredients, the RADIOAMICA dashboard provides an expressive and modular logic-driven analytical environment. Domain experts can navigate from broad cohort-level distributions to individual flagged records and their constraint-grounded explanations within a single integrated platform.

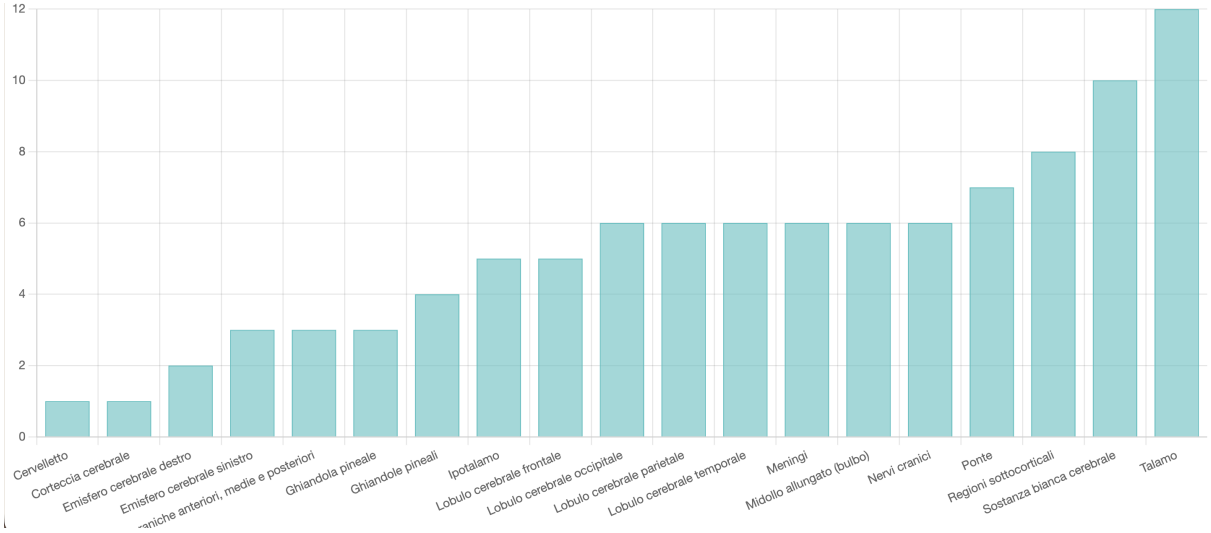
5. Related Work

Answer Set Programming (ASP) has seen growing adoption across a range of real-world domains, being the medical and clinical field one of its active application areas. Its declarative nature and strong formal semantics make it particularly well-suited for domains requiring expressive constraint modeling, consistency checking, scheduling and explainability [11, 12, 25, 26].

In the medical domain, ASP has been extensively applied to the analysis and execution of clinical guidelines. Several works focus on temporal conformance analysis, enabling the verification of whether clinical processes comply with formalized guidelines [27, 28, 29]. These approaches leverage Answer Set Programming to perform a posteriori analysis, capturing temporal constraints, interactions among multiple guidelines, and the interplay between clinical guidelines and basic medical knowledge, while



(a) Prevalence of mutations.



(b) Frequency of lesions.

Figure 10: Comparison of mutation prevalence and anatomical distribution. Pie chart: https://asp-chef.alviano.net/s/CILC2026/prevalenza-mutazioni#pit0500/asp-chef-short-links. Bar plot: https://asp-chef.alviano.net/s/CILC2026/sede-count#pit0500/asp-chef-short-links

providing explanations for observed deviations and inconsistencies. A related line of research addresses the problem of managing multiple concurrent clinical guidelines and resolving potential conflicts. Constraint-based and logic-based approaches have been proposed to detect and mitigate adverse interactions between treatments [30, 31, 32, 33]. Planning-based solutions, such as MitPlan, extend these ideas by generating conflict-free treatment strategies in the presence of multiple guidelines [34, 35, 36]. Similarly, ASP has been used to generate safe and consistent treatment plans for patients with comorbidities [37].

Beyond guideline management, ASP has also been applied to various healthcare optimization problems, particularly in scheduling and resource allocation. Recent works demonstrate its effectiveness in solving complex scheduling tasks in clinical settings, such as surgical planning and outpatient pathway management [38, 39, 40, 41, 42, 43, 44, 26]. These applications highlight the scalability and flexibility of ASP in real-world healthcare scenarios.

In parallel, logic-based approaches have been explored for reasoning over biomedical and clinical data, including cancer data analysis and biological systems modeling [45, 46, 47, 48]. More recently, rule-based systems such as Vadalog have been proposed to support reasoning over patient pathways and health records [49]. Inconsistency-tolerant reasoning has also been studied in the context of knowledge bases, providing formal tools to handle conflicting information even when complete consistency cannot be guaranteed [50]. Building on this capacity for inconsistency handling, ASP has been applied to

detect inconsistencies in formal requirements specifications for software systems [51], with validation carried out on artifacts from a healthcare application [51]. Ontology-based approaches have similarly been proposed to detect semantic inconsistencies across heterogeneous clinical data representations in electronic health records [52], underlining the broader recognition that consistency checking across multi-source clinical data is a significant open challenge.

Despite these advances, most existing approaches focus either on guideline execution, treatment planning, or optimization problems, and do not explicitly address the integration and consistency checking of heterogeneous, multi-dimensional clinical data. Moreover, while some works consider explainability, they often lack interactive and visual tools to support domain experts in exploring and understanding inconsistencies. In contrast, our work targets the integration of heterogeneous clinical data by combining ASP-based reasoning with an interactive dashboard. The proposed approach enables the detection of inconsistencies across multiple dimensions (e.g., anatomical, histological, imaging, and genetic) and provides graph-based explanations, thus bridging the gap between logical reasoning, data integration, and user-centered visualization.

6. Conclusion

This paper presented an ASP-based approach for the analysis of clinical data in neuro-oncology, combining declarative reasoning with interactive visual analytics. The proposed system addresses the validation of heterogeneous clinical records by transforming raw JSON inputs into a logical representation, enabling the application of domain-specific constraints through Answer Set Programming. Within this framework, inconsistencies are identified as violations of integrity constraints and are made accessible through a pipeline that connects data preprocessing, logical inference, and visual exploration. Starting from JSON inputs, the system produces a normalized set of logical facts, which are then evaluated against clinical constraints to identify inconsistent configurations. The approach is integrated into the RADIOAMICA platform through the ASP Chef pipeline, which combines JSON Path-driven data ingestion with declarative reasoning and Mustache-based visualization. The resulting system provides both tabular interfaces and chart-based visualizations, allowing domain experts to inspect flagged cases and understand the conditions that led to each inconsistency in a transparent and interactive way. Rather than enforcing automated decisions, the system supports clinical validation by highlighting potentially inconsistent or unusual combinations of attributes, while preserving the role of expert judgment. In this perspective, it provides an explainable decision-support layer that links formal reasoning techniques with practical data analysis workflows. Future work will focus on extending the set of medical constraints, evaluating the approach on larger datasets, and investigating methods to handle incomplete or uncertain information.

Acknowledgments

This work was supported by the Italian Ministry of Health (MSAL) under POS projects CAL.HUB.RIA (CUP H53C22000800006) and RADIOAMICA (CUP H53C22000650006); by the Italian Ministry of Enterprises and Made in Italy under project STROKE 5.0 (CUP B29J23000430005); under PN RIC project ASVIN “Assistente Virtuale Intelligente di Negozio” (CUP B29J24000200005); by Regione Calabria under project NextGenGuides “Innovazione Tecnologica per le Guide Turistiche del Futuro tramite l’ausilio di tecnologie innovative AI e sistemi di geolocalizzazione avanzata” (CUP J39I24002040005); under project MOZART “Modelli e tecniche di mOdernizzazione di applicaZioni e data silos a supporto di clienti esterni, field force e impiegati di backoffice basati su AI GeneRativa e apprendimento auTomatico” (CUP J49I24001740005); by the LAIA lab (part of the SILA labs). Mario Alviano is a member of Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT for grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Commun. ACM 54 (2011) 92–103. doi:10.1145/2043174.2043195.

[2] E. Erdem, M. Gelfond, N. Leone, Applications of answer set programming, AI Mag. 37 (2016) 53–68.

[3] V. Lifschitz, Answer Set Programming, Springer, 2019.

[4] R. Kaminski, J. Romero, T. Schaub, P. Wanko, How to build your own asp-based system?!, Theory Pract. Log. Program. 23 (2023) 299–361.

[5] M. Alviano, C. Dodaro, S. Fiorentino, A. Previti, F. Ricca, ASP and subset minimality: Enumeration, cautious reasoning and muses, Artif. Intell. 320 (2023) 103931.

[6] F. Wotawa, On the use of answer set programming for model-based diagnosis, in: H. Fujita, P. Fournier-Viger, M. Ali, J. Sasaki (Eds.), IEA/AIE 2020, Kitakyushu, Japan, September 22-25, 2020, Proceedings, volume 12144 of *LNCS*, Springer, 2020, pp. 518–529. doi:10.1007/978-3-030-55789-8_45.

[7] R. Taupe, G. Friedrich, K. Schekotihin, A. Weinzierl, Solving configuration problems with ASP and declarative domain specific heuristics, in: M. Aldanondo, A. A. Falkner, A. Felfernig, M. Stettinger (Eds.), Proceedings of the 23rd International Configuration Workshop (CWS/ConfWS 2021), Vienna, Austria, 16-17 September, 2021, volume 2945 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 13–20. URL: https://ceur-ws.org/Vol-2945/21-RT-ConfWS21_paper_4.pdf.

[8] F. Wotawa, D. Kaufmann, Model-based reasoning using answer set programming, Appl. Intell. 52 (2022) 16993–17011. URL: https://doi.org/10.1007/s10489-022-03272-2. doi:10.1007/S10489-022-03272-2.

[9] L. M. Prikler, F. Wotawa, Faster diagnosis with answer set programming (short paper), in: I. Pill, A. Natan, F. Wotawa (Eds.), 35th International Conference on Principles of Diagnosis and Resilient Systems, DX 2024, Vienna, Austria, November 4-7, 2024, OASIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, pp. 24:1–24:13. URL: https://doi.org/10.4230/OASIcs.DX.2024.24. doi:10.4230/OASICS.DX.2024.24.

[10] P. Cappanera, M. Gavanelli, M. Nonato, M. Roma, Logic-based benders decomposition in answer set programming for chronic outpatients scheduling, TPLP 23 (2023) 848–864. doi:10.1017/S147106842300025X.

[11] M. Cardellini, P. D. Nardi, C. Dodaro, G. Galatà, A. Giardini, M. Maratea, I. Porro, Solving rehabilitation scheduling problems via a two-phase ASP approach, TPLP 24 (2024) 344–367. doi:10.1017/S1471068423000030.

[12] E. Erdem, U. Öztok, Generating explanations for biomedical queries, Theory Pract. Log. Program. 15 (2015) 35–78. URL: https://doi.org/10.1017/S1471068413000598. doi:10.1017/S1471068413000598.

[13] Z. Chen, Automating disease management using answer set programming, in: M. Carro, A. King, N. Saeedloei, M. D. Vos (Eds.), Technical Communications of the 32nd International Conference on Logic Programming, ICLP 2016 TCs, New York City, USA, October 16-21, 2016, OASIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, pp. 22:1–22:10. URL: https://doi.org/10.4230/OASIcs.ICLP.2016.22. doi:10.4230/OASICS.ICLP.2016.22.

[14] S. Caruso, G. Galatà, M. Maratea, M. Mochi, I. Porro, Scheduling pre-operative assessment clinic with answer set programming, J. Log. Comput. 34 (2024) 465–493. URL: https://doi.org/10.1093/logcom/exad017. doi:10.1093/LOGCOM/EXAD017.

[15] P. Cappanera, S. Caruso, C. Dodaro, G. Galatà, M. Gavanelli, M. Maratea, C. Marte, M. Mochi,

M. Nonato, M. Roma, Recent answer set programming applications to scheduling problems in digital health, in: D. Aineto, R. D. Benedictis, M. Maratea, M. Mittelmann, G. Monaco, E. Scala, L. Serafini, I. Serina, F. Spegni, E. Tosello, A. Umbrico, M. Vallati (Eds.), Proceedings of the International Workshop on Artificial Intelligence for Climate Change, Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (AI4CC-IPS-RCRA-SPIRIT 2024) co-located with 23rd International Conference of the Italian Association for Artificial Intelligence AIxIA 2024, November 25-28th, 2024, Bolzano, Italy, CEUR Workshop Proceedings, CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3883/paper3_RCRA8.pdf.

[16] A. Vozna, A. Monaldini, S. Costantini, D. Pado, V. Pitoni, Scalable and ethical medical scheduling: An asp-based framework with patient-centered experimental validation, in: D. Azzolini, S. Batsakis, M. A. B. Santana, P. Bruno, L. A. Cecchi, J. Fandinno, M. Hecher, M. Maratea, J. F. Morales, B. Muñiz, E. Papadakis, L. Robaldo, L. Serafini, F. L. Scudo, I. Tachmazidis, A. Tarzariol, M. Vallati, A. Wyner (Eds.), Joint Proceedings of the Workshops and Doctoral Consortium of the 41st International Conference on Logic Programming (ICLP-WS-DC 2025) co-located with 41st International Conference on Logic Programming (ICLP 2025), Rende, Italy, September 9-13, 2025, CEUR Workshop Proceedings, CEUR-WS.org, 2025, p. 8. URL: https://ceur-ws.org/Vol-4117/short_rcra_5.pdf.

[17] V. Pitoni, S. Costantini, A. Monaldini, A. Vozna, From blueprint personas to epistemic agents: A comparative study of asp-based and l-dinf-based approaches to medical appointment scheduling, in: D. Guidotti, L. Pandolfo, L. Pulina (Eds.), Proceedings of the 40th Italian Conference on Computational Logic, Alghero, Italy, June 25-27, 2025, CEUR Workshop Proceedings, CEUR-WS.org, 2025. URL: https://ceur-ws.org/Vol-4003/paper21.pdf.

[18] A. Vozna, A. Monaldini, S. Costantini, A trust-aware architecture for personalized digital health: Integrating blueprint personas and ontology-based reasoning, J. Medical Syst. 49 (2025) 125. URL: https://doi.org/10.1007/s10916-025-02255-3. doi:10.1007/S10916-025-02255-3.

[19] M. Alviano, D. Cirimele, L. A. R. Reiners, Introducing ASP recipes and ASP chef, in: J. Arias, S. Batsakis, W. Faber, G. Gupta, F. Pacenza, E. Papadakis, L. Robaldo, K. Rückschloß, E. Salazar, Z. G. Saribatur, I. Tachmazidis, F. Weitkämper, A. Z. Wyner (Eds.), Proceedings of the International Conference on Logic Programming 2023 Workshops co-located with the 39th International Conference on Logic Programming (ICLP 2023), London, United Kingdom, July 9th and 10th, 2023, CEUR Workshop Proceedings, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3437/paper4ASPOCP.pdf.

[20] M. Gelfond, V. Lifschitz, Logic programs with classical negation, in: D. Warren, P. Szeredi (Eds.), Logic Programming: Proc. of the Seventh International Conference, 1990, pp. 579–597.

[21] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, M. Maratea, F. Ricca, T. Schaub, ASP-Core-2 input language format, Theory Pract. Log. Program. 20 (2020) 294–309. URL: https://doi.org/10.1017/S1471068419000450. doi:10.1017/S1471068419000450.

[22] D. N. Louis, A. Perry, P. Wesseling, D. J. Brat, I. A. Cree, D. Figarella-Branger, C. Hawkins, H. K. Ng, S. M. Pfister, G. Reifenberger, R. Soffietti, A. von Deimling, D. W. Ellison, The 2021 who classification of tumors of the central nervous system: a summary, Neuro-Oncology 23 (2021) 1231–1251. URL: https://doi.org/10.1093/neuonc/noab106. doi:10.1093/neuonc/noab106. arXiv:https://academic.oup.com/neuro-oncology/article-pdf/23/8/1231/39535372/noab106.pdf.

[23] M. Alviano, L. A. R. Reiners, W. Faber, ASP chef grows mustache to look better, Theory Pract. Log. Program. 25 (2025) 417–436. URL: https://doi.org/10.1017/s1471068425100094. doi:10.1017/S1471068425100094.

[24] A. Guzzo, G. Fortino, G. Greco, M. Maggiolini, Data and model aggregation for radiomics applications: Emerging trend and open challenges, Inf. Fusion 100 (2023) 101923. URL: https://doi.org/10.1016/j.inffus.2023.101923. doi:10.1016/J.INFFUS.2023.101923.

[25] Z. Chen, Automating disease management using answer set programming (2018).

[26] M. Alviano, R. Bertolucci, M. Cardellini, C. Dodaro, G. Galatà, M. K. Khan, M. Maratea, M. Mochi, V. Morozan, I. Porro, M. Schouten, Answer set programming in healthcare: Extended overview, in: R. D. Benedictis, M. Maratea, A. Micheli, E. Scala, I. Serina, A. Umbrico, M. Vallati (Eds.), Joint

Proceedings of the 8th Italian Workshop on Planning and Scheduling and the 27th International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion co-located with AIxIA 2020, Online Event, November 25-27, 2020, CEUR Workshop Proceedings, CEUR-WS.org, 2020. URL: https://ceur-ws.org/Vol-2745/paper7.pdf.

[27] M. Spiotta, P. Terenziani, D. Theseider Dupré, Answer set programming for temporal conformance analysis of clinical guidelines execution, in: Conference on Artificial Intelligence in Medicine in Europe, Springer, 2015, pp. 65–79.

[28] M. Spiotta, P. Terenziani, D. T. Dupré, Temporal conformance analysis and explanation of clinical guidelines execution: An answer set programming approach, IEEE Transactions on Knowledge and Data Engineering 29 (2017) 2567–2580.

[29] L. Piovesan, P. Terenziani, D. T. Dupré, Conformance analysis for comorbid patients in answer set programming, Journal of Biomedical Informatics 103 (2020) 103377.

[30] L. Piovesan, G. Molino, P. Terenziani, Supporting physicians in the detection of the interactions between treatments of co-morbid patients, in: Healthcare informatics and analytics: emerging issues and trends, IGI Global Scientific Publishing, 2015, pp. 165–193.

[31] L. Piovesan, P. Terenziani, A constraint-based approach for the conciliation of clinical guidelines, in: Ibero-American Conference on Artificial Intelligence, Springer, 2016, pp. 77–88.

[32] S. Wilk, W. Michalowski, M. Michalowski, K. Farion, M. M. Hing, S. Mohapatra, Mitigation of adverse interactions in pairs of clinical practice guidelines using constraint logic programming, Journal of biomedical informatics 46 (2013) 341–353.

[33] S. Wilk, M. Michalowski, W. Michalowski, D. Rosu, M. Carrier, M. Kezadri-Hamiaz, Comprehensive mitigation framework for concurrent application of multiple clinical practice guidelines, Journal of Biomedical Informatics 66 (2017) 52–71.

[34] M. Michalowski, S. Wilk, W. Michalowski, D. Lin, K. Farion, S. Mohapatra, Using constraint logic programming to implement iterative actions and numerical measures during mitigation of concurrently applied clinical practice guidelines, in: Conference on Artificial Intelligence in Medicine in Europe, Springer, 2013, pp. 17–22.

[35] M. Michalowski, S. Wilk, W. Michalowski, M. Carrier, Mitplan: A planning approach to mitigating concurrently applied clinical practice guidelines, in: Conference on artificial intelligence in medicine in Europe, Springer, 2019, pp. 93–103.

[36] M. Michalowski, S. Wilk, W. Michalowski, M. Carrier, Mitplan: A planning approach to mitigating concurrently applied clinical practice guidelines, Artificial Intelligence in Medicine 112 (2021) 102002.

[37] E. Merhej, S. Schockaert, T. G. McKelvey, M. De Cock, Generating conflict-free treatments for patients with comorbidity using answer set programming, in: International Workshop on Process-oriented Information Systems in Healthcare, Springer, 2016, pp. 111–119.

[38] P. Cappanera, M. Gavanelli, M. Nonato, M. Roma, Decomposition approaches for scheduling chronic outpatients' clinical pathways in answer set programming, Journal of Logic and Computation 33 (2023) 1851–1871.

[39] M. Mochi, G. Galatà, M. Maratea, Master surgical scheduling via answer set programming, Journal of Logic and Computation 33 (2023) 1777–1803.

[40] G. Galatà, M. Maratea, M. Mochi, Master surgical scheduling via answer set programming tested on real data., in: HC@ AIxIA, 2023, pp. 130–144.

[41] G. Galatà, M. Maratea, C. Marte, M. Mochi, Rescheduling master surgical schedules via answer set programming, Progress in Artificial Intelligence (2024) 1–16.

[42] C. Dodaro, G. Galata, A. Grioni, M. Maratea, M. Mochi, I. Porro, An asp-based solution to the chemotherapy treatment scheduling problem, Theory and Practice of Logic Programming 21 (2021) 835–851.

[43] C. Dodaro, G. Galatà, M. Maratea, C. Marte, M. Mochi, Asp-based approaches for solving the nuclear medicine scheduling problem, Journal of Logic and Computation 36 (2026) exaf071.

[44] P. Cappanera, S. Caruso, C. Dodaro, G. Galatà, M. Gavanelli, M. Maratea, C. Marte, M. Mochi, M. Nonato, M. Roma, et al., Recent answer set programming applications to scheduling problems in

digital health, in: CEUR WORKSHOP PROCEEDINGS, volume 3883, CEUR-WS, 2024, pp. 129–141.
[45] A. Tarzariol, E. Zanazzo, A. Dovier, A. Policriti, Towards a logic programming tool for cancer data analysis, Fundamenta Informaticae 176 (2020) 299–319.
[46] T. Khaled, B. Benhamou, Dealing with biology systems in the framework of answer set programming, Procedia Computer Science 176 (2020) 450–459.
[47] K. Becker, H. Klarner, M. Nowicka, H. Siebert, Designing mirna-based synthetic cell classifier circuits using answer set programming, Frontiers in bioengineering and biotechnology 6 (2018) 70.
[48] L. Chebouba, B. Miannay, D. Boughaci, C. Guziolowski, Discriminate the response of acute myeloid leukemia patients to treatment by using proteomics data and answer set programming, BMC bioinformatics 19 (2018) 59.
[49] O. P. Dwyer, T. Baldazzi, J. Davies, E. Sallinger, A. Vlad, Reasoning over health records with vadalog: a rule-based approach to patient pathways, CEUR Workshop Proceedings, 2023.
[50] M. Bienvenu, C. Bourgaux, Inconsistency-tolerant querying of description logic knowledge bases, in: Reasoning Web International Summer School, Springer, 2016, pp. 156–202.
[51] W. Li, D. Brown, J. H. Hayes, M. Truszczynski, Answer-set programming in requirements engineering, in: C. Salinesi, I. van de Weerd (Eds.), Requirements Engineering: Foundation for Software Quality - 20th International Working Conference, REFSQ 2014, Essen, Germany, April 7-10, 2014. Proceedings, Lecture Notes in Computer Science, Springer, 2014, pp. 168–183. URL: https://doi.org/10.1007/978-3-319-05843-6_13. doi:10.1007/978-3-319-05843-6_13.
[52] Y. K. Awuklu, F. Mougin, R. Griffier, M. Bienvenu, V. Jouhet, Ontology-driven identification of inconsistencies in clinical data: A case study in lung cancer phenotyping, J. Biomed. Informatics 165 (2025) 104808. URL: https://doi.org/10.1016/j.jbi.2025.104808. doi:10.1016/J.JBI.2025.104808.