Universidad Rey Juan Carlos
Escuela Técnica Superior Ingeniería Informática

Razonamiento jurídico informatizado con s(LAW)

y otras líneas de investigación de CETINIA

Sascha Ossowski and Joaquín Arias

Grupo de Inteligencia Artificial de la URJC
Center for Intelligent Information Technologies (CETINIA)
Móstoles, Madrid

1 Diciembre 2023 (JornadasIA'23)

Who I am

Joaquín Arias
Professor at Universidad Rey Juan Carlos (URJC).

- 2020 - today: Researcher at Group of IA at CETINIA, URJC.
- 2013 - 2020: Researcher at IMDEA Sotware Institute.
- Academic background:
  - Ph.D. in Computer Science (2020).
  - M.Sc. in Software and Systems (2015).
  - B.Sc. in Mathematics and informatic (2014).
  - M.Arch. in Architecture (2002).
- PhD Thesis: "Advanced Evaluation Techniques for (Non)-Monotonic Reasoning using Rules with Constraints".

Commonsense Reasoning: Introduction

- Humans employ commonsense reasoning to explain things:
  - E.g., we convert sensory input to knowledge, over which we reason.
- To automate explainability/interpretability in AI: automate the system 2 reflective thinking, i.e., automate commonsense reasoning. "Thinking Fast and Slow" (2011)

Commonsense Reasoning: Introduction

- Humans employ commonsense reasoning to explain things:
  - E.g., we convert sensory input to knowledge, over which we reason.
- To automate explainability/interpretability in AI: automate the system 2 reflective thinking, i.e., automate commonsense reasoning. "Thinking Fast and Slow" (2011)

Commonsense reasoning can be approximated with answer set programming

Default rules, integrity constraints, and assumption-based reasoning

Commonsense Reasoning: Introduction

- Humans employ commonsense reasoning to explain things:
  - E.g., we convert sensory input to knowledge, over which we reason.
- To automate explainability/interpretability in AI: automate the system 2 reflective thinking, i.e., automate commonsense reasoning. "Thinking Fast and Slow" (2011)

Commonsense reasoning can be approximated with answer set programming

Default rules, integrity constraints, and assumption-based reasoning

- **Default Rules**: express what is true in a majority of cases but not always. E.g., "by default birds fly, but there are exceptional ones that do not".

```
flies(X) :- bird(X), not ab0(X).        ab0(X) :- penguin(X).
```

Commonsense Reasoning: Introduction

- Humans employ commonsense reasoning to explain things:
  - E.g., we convert sensory input to knowledge, over which we reason.
- To automate explainability/interpretability in AI: automate the system 2 reflective thinking, i.e., automate commonsense reasoning. "Thinking Fast and Slow" (2011)

Commonsense reasoning can be approximated with answer set programming

Default rules, integrity constraints, and assumption-based reasoning

- **Default Rules**: express what is true in a majority of cases but not always. E.g., "by default birds fly, but there are exceptional ones that do not".

```
flies(X) :- bird(X), not ab0(X).          ab0(X) :- penguin(X).
```

- **Integrity Constraints**: express impossibility conditions. E.g., A person cannot sit and stand at the same time.

```
false :- sit(X), stand(X).
```

Commonsense Reasoning: Introduction

- Humans employ commonsense reasoning to explain things:
  - E.g., we convert sensory input to knowledge, over which we reason.
- To automate explainability/interpretability in AI: automate the system 2 reflective thinking, i.e., automate commonsense reasoning. "Thinking Fast and Slow" (2011)

Commonsense reasoning can be approximated with answer set programming

Default rules, integrity constraints, and assumption-based reasoning

- **Default Rules**: express what is true in a majority of cases but not always. E.g., "by default birds fly, but there are exceptional ones that do not".
  `flies(X) :- bird(X), not ab0(X).` `ab0(X) :- penguin(X).`
- **Integrity Constraints**: express impossibility conditions. E.g., A person cannot sit and stand at the same time. `false :- sit(X), stand(X).`
- **Assumption based reasoning**: if we don't know something, we will assume it holds (or does not hold) and continue (abductive reasoning). E.g., Game of clue.

Commonsense Reasoning: Event Calculus

Commonsense Reasoning (**CR**)

- Requires modelling:
  - Non-monotonicity.
  - Continuous characteristics of the world.
- Event Calculus (**EC**): formalism that represents continuous change and captures *law of inertia*.
- EC components:

  Narrative A description of the world we want to model. Assumes circumscription.

  Axioms A generic description of how the world behaves given a narrative.
- Implementing EC: logic + continuous domains.

Example:

A tap fills a vessel [37].

Commonsense Reasoning: Event Calculus (cont.)

Reasoning on EC:

- Deduction / proving in first order logic (+ circumscription).

Related Work

- Non-interactive theorem prover: likely won't always answer.
- **Prolog:** incomplete implementations [12; 30; 38].
- Answer Set Programming (**ASP**): logic programming paradigm.
  - Has been used to model (discrete) EC [23; 24].
- Classical (C)ASP systems require *grounding*:
  - Limited to variables ranging over discrete, finite domains.
- CASP proposals not applied (yet) to modeling EC:
  - ASPMT [22]: Action languages [17] + continuous time.
  - PDDL+ [6; 14]: Planning & Diagnosis.

Commonsense Reasoning: Event Calculus under s(CASP)

- Goal directed execution **without** grounding.
  - The execution starts with a query. `?- T #> 5, T #< 8, cross(T).`
  - Returns partial stable models [16] Only literals supporting the query.
  - For each successful top-down derivation, on backtracking returns:
    - A justification tree. Explanation for observations.
    - Bindings for the free variables. `T=6.`
    - Constraint representing refined domain of the variables. `T #> 5, T #< 7.`

Commonsense Reasoning: Event Calculus under s(CASP)

- Goal directed execution **without** grounding.
  - The execution starts with a query. `?- T #> 5, T #< 8, cross(T).`
  - Returns partial stable models [16] Only literals supporting the query.
  - For each successful top-down derivation, on backtracking returns:
    - A justification tree. Explanation for observations.
    - Bindings for the free variables. `T=6.`
    - Constraint representing refined domain of the variables. `T #> 5, T #< 7.`
- Provides a constructive and sound negation:
  - Default negation: In the absence of information. `cross(T):- not train(T).`
  - Classical negation: Explicit knowledge. `cross(T):- -train(T).`

Commonsense Reasoning: Event Calculus under s(CASP)

- Goal directed execution **without** grounding.
  - The execution starts with a query. `?- T #> 5, T #< 8, cross(T).`
  - Returns partial stable models [16] Only literals supporting the query.
  - For each successful top-down derivation, on backtracking returns:
    - A justification tree. Explanation for observations.
    - Bindings for the free variables. `T=6.`
    - Constraint representing refined domain of the variables. `T #> 5, T #< 7.`
- Provides a constructive and sound negation:
  - Default negation: In the absence of information. `cross(T):- not train(T).`
  - Classical negation: Explicit knowledge. `cross(T):- -train(T).`
- Allows rules with negated heads. `-train(T):- not barrier(up,T).`
  - Global constraints ensure consistency. `:- train(T), -train(T).`

Commonsense Reasoning: Event Calculus under s(CASP) (cont.)

- EC uses a universal theory (axioms) to reason about scenarios (narrative).
  - Event are actions that may happens at a time point. `tapOn` Open the tap.
  - A fluent is a time-varying property of the world. `filling` The vessel is been filled.
  - Time and/or fluent may have continuous quantities. `level(X)` Level of water.

Commonsense Reasoning: Event Calculus under s(CASP) (cont.)

- EC uses a universal theory (axioms) to reason about scenarios (narrative).
  - Event are actions that may happens at a time point. `tapOn` Open the tap.
  - A fluent is a time-varying property of the world. `filling` The vessel is been filled.
  - Time and/or fluent may have continuous quantities. `level(X)` Level of water.
- State constraints: represent restrictions on the model:
  - To ensure consistency of the narrative w.r.t. the axioms.

```
:- holds(F,T), -holds(F,T).
```

Commonsense Reasoning: Event Calculus under s(CASP) (cont.)

- EC uses a universal theory (axioms) to reason about scenarios (narrative).
  - Event are actions that may happens at a time point. `tapOn` Open the tap.
  - A fluent is a time-varying property of the world. `filling` The vessel is been filled.
  - Time and/or fluent may have continuous quantities. `level(X)` Level of water.
- State constraints: represent restrictions on the model:
  - To ensure consistency of the narrative w.r.t. the axioms.
    `:- holds(F,T), -holds(F,T).`
- Trajectory: a fluent depends on the time elapsed since an event:
  - If at `T1` the level of water is `L1`. Then, at `T2` the level is `L1+T2-T1`.

Commonsense Reasoning: Event Calculus under s(CASP) (cont.)

- EC uses a universal theory (axioms) to reason about scenarios (narrative).
  - Event are actions that may happens at a time point. `tapOn` Open the tap.
  - A fluent is a time-varying property of the world. `filling` The vessel is been filled.
  - Time and/or fluent may have continuous quantities. `level(X)` Level of water.
- State constraints: represent restrictions on the model:
  - To ensure consistency of the narrative w.r.t. the axioms.
    `:- holds(F,T), -holds(F,T).`
- Trajectory: a fluent depends on the time elapsed since an event:
  - If at `T1` the level of water is `L1`. Then, at `T2` the level is `L1+T2-T1`.
- Non-monotonic reasoning
  - Different scenarios/worlds (uncertainty). Vessel size `10` or `16`.
  - Abductive reasoning: events may happens or not. Sequence of events.

Foundations of Constraint ASP without Grounding

ASP + Constraints - Grounding = s(CASP)

- s(CASP) evaluates ASP programs with constraints:
  - Follows a top-down execution strategy based on s(ASP) [Marple et al. 2017].
  - Constructive negation, `not p(X)`, is resolved against the dual of `p(X)`.

```
p(0) :- s.
p(X) :- q(X,Y).
```

```
not p(X) :- not p1(X), not p2(X).
not p1(X) :- X\=0.
not p1(X) :- X=0, not s.
not p2(X) :- c_forall(Y, not q(X,Y)).
```

  - A new clp($\neq$) solver handles (partially) the negation of the unification.
  - A new `c_forall/2` predicate computes the universal quantifier.
- To ensure consistency the *extended* compiler synthesizes NMR-check.

```
p(X) :- q(X,Y), not p(X).
```

$$\forall x(\ chk_i(x) \longleftrightarrow \forall y(\ \neg q(x,y) \vee p(x)\ )\ )$$

- Facilitates the integration of different constraint domains, e.g., clp(Q) and clp(R).

Foundations of Constraint ASP without Grounding (cont.)

Internals: Universal quantifier

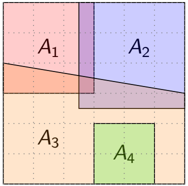
We execute `c_forall(X,goal(X))` to determine if `goal(X)` is true
...for all possible values of `X` in its constraint domain.

Foundations of Constraint ASP without Grounding (cont.)

Internals: Universal quantifier

We execute `c_forall(X,goal(X))` to determine if `goal(X)` is true

...for all possible values of `X` in its constraint domain.



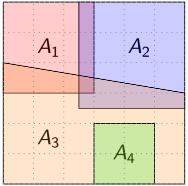
answers

`goal(X)`$\{\top\}$

Foundations of Constraint ASP without Grounding (cont.)

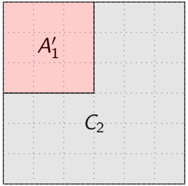
Internals: Universal quantifier

We execute `c_forall(X,goal(X))` to determine if `goal(X)` is true
...for all possible values of `X` in its constraint domain.



answers

`goal(X)`$\{\top\}$



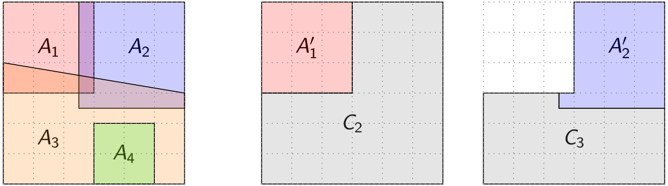
(a)

$A'_1 = A_1 \wedge \top$
$C_2 = \top \wedge \neg A'_1$
`goal(X)`$\{C_2\}$

Foundations of Constraint ASP without Grounding (cont.)

Internals: Universal quantifier

We execute `c_forall(X,goal(X))` to determine if `goal(X)` is true
...for all possible values of `X` in its constraint domain.



answers

`goal(X)`$\{\top\}$

(a)

$A'_1 = A_1 \wedge \top$
$C_2 = \top \wedge \neg A'_1$
`goal(X)`$\{C_2\}$

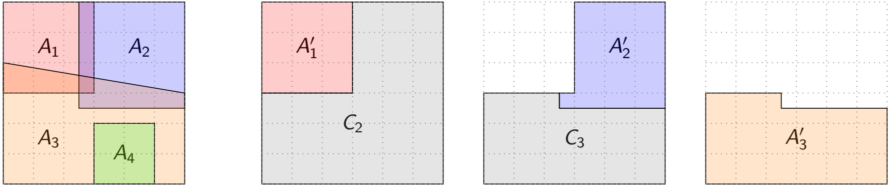
(b)

$A'_2 = A_2 \wedge C_2$
$C_3 = C_2 \wedge \neg A'_2$
`goal(X)`$\{C_3\}$

Foundations of Constraint ASP without Grounding (cont.)

Internals: Universal quantifier

We execute `c_forall(X,goal(X))` to determine if `goal(X)` is true
...for all possible values of `X` in its constraint domain.



answers | (a) | (b) | (c)

`goal(X)`$\{\top\}$

(a)
$A'_1 = A_1 \wedge \top$
$C_2 = \top \wedge \neg A'_1$
`goal(X)`$\{C_2\}$

(b)
$A'_2 = A_2 \wedge C_2$
$C_3 = C_2 \wedge \neg A'_2$
`goal(X)`$\{C_3\}$

(c)
$A'_3 = A_3 \wedge C_3$
$C_4 = C_3 \wedge \neg A'_3 = \emptyset$
End

The *C-forall* evaluation succeeds.

Foundations of Constraint ASP without Grounding (cont.)

Internals: Universal quantifier (cont.)

If the set of answers of `goal(X)` is constraint-compact the algorithm finishes.

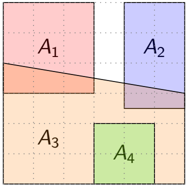
...this happens also when the answers do not cover the domain of `X`.

Foundations of Constraint ASP without Grounding (cont.)

Internals: Universal quantifier (cont.)

If the set of answers of `goal(X)` is constraint-compact the algorithm finishes.

...this happens also when the answers do not cover the domain of `X`.



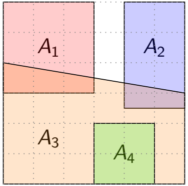
answers

`goal(X)`$\{\top\}$

Foundations of Constraint ASP without Grounding (cont.)

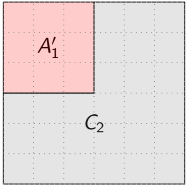
Internals: Universal quantifier (cont.)

If the set of answers of `goal(X)` is constraint-compact the algorithm finishes.
...this happens also when the answers do not cover the domain of `X`.



answers

`goal(X)`$\{\top\}$



(a)

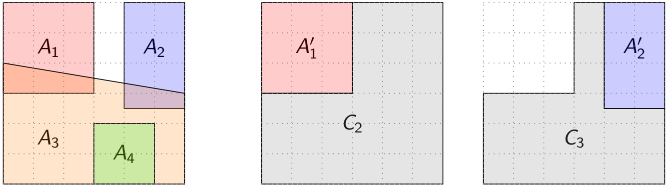
$A'_1 = A_1 \land \top$
$C_2 = \top \land \neg A'_1$
`goal(X)`$\{C_2\}$

Foundations of Constraint ASP without Grounding (cont.)

Internals: Universal quantifier (cont.)

If the set of answers of `goal(X)` is constraint-compact the algorithm finishes.

...this happens also when the answers do not cover the domain of `X`.



answers — (a) — (b)

`goal(X)`$\{\top\}$

(a)

$A'_1 = A_1 \wedge \top$
$C_2 = \top \wedge \neg A'_1$
`goal(X)`$\{C_2\}$

(b)

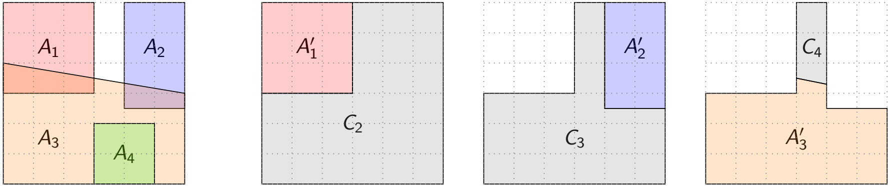
$A'_2 = A_2 \wedge C_2$
$C_3 = \top \wedge \neg A'_2$
`goal(X)`$\{C_3\}$

Foundations of Constraint ASP without Grounding (cont.)

Internals: Universal quantifier (cont.)

If the set of answers of `goal(X)` is constraint-compact the algorithm finishes.

...this happens also when the answers do not cover the domain of `X`.



answers | (a) | (b) | (c)

`goal(X)`$\{\top\}$

(a)
$A'_1 = A_1 \land \top$
$C_2 = \top \land \neg A'_1$
`goal(X)`$\{C_2\}$

(b)
$A'_2 = A_2 \land C_2$
$C_3 = \top \land \neg A'_2$
`goal(X)`$\{C_3\}$

(c)
$A'_3 = A_3 \land C_3$
$C_4 = \top \land \neg A'_3$
`goal(X)`$\{C_4\}$ fails

The *C-forall* evaluation fails.

Foundations of Constraint ASP without Grounding (cont.)

Implementation

- The resulting program is evaluated by a *new* interpreter.
  - Delegates unification, disequality, and constraint propagation in Ciao Prolog.
  - Detects and handles different types of recursion to avoid infinite loops.
  - Returns partial stable models and their corresponding proof trees.

Foundations of Constraint ASP without Grounding (cont.)

Implementation

- The resulting program is evaluated by a *new* interpreter. ∼10X faster
  - Delegates unification, disequality, and constraint propagation in Ciao Prolog.
  - Detects and handles different types of recursion to avoid infinite loops.
  - Returns partial stable models and their corresponding proof trees.

	s(CASP)	s(ASP)
hanoi(8,T)	**1,528**	13,297
queens(4,Q)	**1,930**	20,141
One hamicycle	**493**	3,499
Two hamicycle	**3,605**	18,026

Run time (ms) comparison for different ASP programs.

Foundations of Constraint ASP without Grounding (cont.)

Implementation

- The resulting program is evaluated by a *new* interpreter. ~10X faster
  - Delegates unification, disequality, and constraint propagation in Ciao Prolog.
  - Detects and handles different types of recursion to avoid infinite loops.
  - Returns partial stable models and their corresponding proof trees.

	s(CASP)	s(ASP)
hanoi(8,T)	**1,528**	13,297
queens(4,Q)	**1,930**	20,141
One hamicycle	**493**	3,499
Two hamicycle	**3,605**	18,026

Run time (ms) comparison for different ASP programs.

- s(CASP) is available as a bundle of Ciao Prolog [https://gitlab.software.imdea.org/ciao-lang/sCASP]

Foundations of Constraint ASP without Grounding (cont.)

Implementation

- The resulting program is evaluated by a *new* interpreter. ∼10X faster
  - Delegates unification, disequality, and constraint propagation in Ciao Prolog.
  - Detects and handles different types of recursion to avoid infinite loops.
  - Returns partial stable models and their corresponding proof trees.

	s(CASP)	s(ASP)
hanoi(8,T)	**1,528**	13,297
queens(4,Q)	**1,930**	20,141
One hamicycle	**493**	3,499
Two hamicycle	**3,605**	18,026

Run time (ms) comparison for different ASP programs.

- s(CASP) is available as a bundle of Ciao Prolog ... and SWI Prolog
  [https://gitlab.software.imdea.org/ciao-lang/sCASP]

Applications of s(CASP)

- Event Calculus Reasoner. [Arias et al. 2019]
  - Model real-word avionics systems. [Hall et al. 2021]
  - Used for cyber-defense (network of device). [Moyle et al. 2023]
- Explainable Artificial Intelligence (XAI). [Arias et al. 2020]
  - Medical advice system. [Chen et al. 2016]
  - Inductive Logic Programming. [Shakerin and Gupta 2019]
- Natural language understanding systems. [Basu et al. 2021a]
  - CASPR: chatbot for the "Alexa Grand Challenge 4". [Basu et al. 2021b]
- Coding rule 34 of the Singapore Bar. [Morris 2021]
- Administrative and judicial discretion reasoner. [Arias et al. 2021]
- A spatial reasoner for Building Information Modelling. [Arias et al. 2022]

... and others (visit GDE'21, GDE'22, GDE'23 Workshops).

Applications of s(CASP)

- Event Calculus Reasoner. [Arias et al. 2019]
  - Model real-word avionics systems. [Hall et al. 2021]
  - Used for cyber-defense (network of device). [Moyle et al. 2023]
- Explainable Artificial Intelligence (XAI). [Arias et al. 2020]
  - Medical advice system. [Chen et al. 2016]
  - Inductive Logic Programming. [Shakerin and Gupta 2019]
- Natural language understanding systems. [Basu et al. 2021a]
  - CASPR: chatbot for the "Alexa Grand Challenge 4". [Basu et al. 2021b]
- Coding rule 34 of the Singapore Bar. [Morris 2021]
- **Administrative and judicial discretion reasoner.** [Arias et al. 2021]
- A spatial reasoner for Building Information Modelling. [Arias et al. 2022]

... and others (visit GDE'21, GDE'22, GDE'23 Workshops).

Applications of s(CASP)

- Event Calculus Reasoner. [Arias et al. 2019]
  - Model real-word avionics systems. [Hall et al. 2021]
  - Used for cyber-defense (network of device). [Moyle et al. 2023]
- Explainable Artificial Intelligence (XAI). [Arias et al. 2020]
  - Medical advice system. [Chen et al. 2016]
  - Inductive Logic Programming. [Shakerin and Gupta 2019]
- Natural language understanding systems. [Basu et al. 2021a]
  - CASPR: chatbot for the "Alexa Grand Challenge 4". [Basu et al. 2021b]
- Coding rule 34 of the Singapore Bar. [Morris 2021]
- **Administrative and judicial discretion reasoner.** [Arias et al. 2021]
- **A spatial reasoner for Building Information Modelling.** [Arias et al. 2022]

... and others (visit GDE'21, GDE'22, GDE'23 Workshops).

Applications of s(CASP)

- Event Calculus Reasoner. [Arias et al. 2019]
  - Model real-word avionics systems. [Hall et al. 2021]
  - Used for cyber-defense (network of device). [Moyle et al. 2023]
- Explainable Artificial Intelligence (XAI). [Arias et al. 2020]
  - Medical advice system. [Chen et al. 2016]
  - **Inductive Logic Programming.** [Shakerin and Gupta 2019]
- Natural language understanding systems. [Basu et al. 2021a]
  - CASPR: chatbot for the "Alexa Grand Challenge 4". [Basu et al. 2021b]
- Coding rule 34 of the Singapore Bar. [Morris 2021]
- **Administrative and judicial discretion reasoner.** [Arias et al. 2021]
- **A spatial reasoner for Building Information Modelling.** [Arias et al. 2022]

... and others (visit GDE'21, GDE'22, GDE'23 Workshops).

Modeling Administrative Discretion Using Goal-Directed Answer Set Programming.

Joaquín Arias, Mar Moreno-Rebato, José A. Rodríguez-García, and Sascha Ossowski (2021).

s(LAW): Introduction [Arias et al. 2021]

- Formal representation & automated reasoning of legal texts:
  - Interest in smart contracts, and public administrations [13; 27; 39].
  - For deterministic rules: proposals on logic programming [33; 35].
- However, they do not easily represent ambiguity or discretion.

s(LAW): Introduction [Arias et al. 2021]

- Formal representation & automated reasoning of legal texts:
  - Interest in smart contracts, and public administrations [13; 27; 39].
  - For deterministic rules: proposals on logic programming [33; 35].
- However, they do not easily represent ambiguity or discretion.

Our Proposal: s(LAW)

- Based on s(CASP) [Arias et al. 2018].
- Allows modeling legal rules involving ambiguity:
  - E.g. awarding school places in "Comunidad de Madrid".
- Supports reasoning and infers conclusion based on these rules.
- Provides justifications of the inferences (in NL) [Arias et al. 2020].

s(LAW): Administrative and judicial discretion reasoner

Two main contributions:

- Set of patterns to translate legal rules into ASP.
  - Including patterns to generate readable justifications.
- Framework to model, reason, and justify conclusions based on:
  - Evidences provided by the user.
  - The applicable legislation.
  - Representing ambiguity, discretion or incomplete information .

s(LAW): Administrative and judicial discretion reasoner

Two main contributions:

- Set of patterns to translate legal rules into ASP.
  - Including patterns to generate readable justifications.
- Framework to model, reason, and justify conclusions based on:
  - Evidences provided by the user.
  - The applicable legislation.
  - Representing ambiguity, discretion or incomplete information .

Related Work:

- *Human-Understandable* explanation for AI advice:
  - Not possible for ML-based systems.
- *Current ASP explanation frameworks* [10; 32; 34]:
  - Only support grounded programs,
  - ... or do not justify negated literals,
  - ... and, do not support constraints and/or dense domains.

s(LAW): latest BREAKING *use case*

It's News: 4th Stimulus Check / Stimulus Check Live / Latest Sports News / Novak Djokovic / Stimulus Checks 2022 / Stimulus Check / Transfer News / Fitness / Child Benefit Rates / UK Tax Brackets / Universal Credit / Child Ta

MARCA EN NFL NBA Soccer MLB More Sports Lifestyle

Tennis / US Open Australian Open Wimbledon Roland-Garros Novak Djokovic Roger Federer Rafael Nadal

Australian Open

Novak Djokovic's deportation dramas overshadow Australian Open

Novak Djokovi

But **Immigration Minister Alex Hawke** used his ministerial discretion to cancel the **34-year-old Serb's visa** on public interest grounds - announcing it at around 6 p.m. - only three days before play is set to first major tournament of the year.

second

Djokovic. Novak Djokovic recognises mistakes in Australia travel documents

Australian Open

s(LAW): Patterns to translate law into ASP

Requirement For Applying

- Disjunction — S/he obtains a school place **if one** of the following common requirements are met
- Conjunction — **In addition**, some of the specific requirements must be met

Exceptions For Applying — Students coming from non-bilingual schools, **need** to accredit B1

Ambiguity — **In case of** *force majeure*, students may be reassigned to a school from another district

This pattern generates two models:

- One where `force_majeure` is assumed to hold.
- Another model where there is **no** evidence that `force_majeure` holds.

Discretion To Act — The School Council **may add** another complementary criterion

Unknown Information — It **may be unclear** whether the documents we have are valid or not

- `-evidence/1`: The *strong* negation (`-`) is used to specify that we have evidences supporting the falsehood of some information.

s(LAW): The framework

ArticleESO.pl

- Contains the legislation rules in Fig. 1 following the patterns described.

ArticleESO.pre.pl

- Contains the natural language patterns to provide readable justifications.
- The directive `#pred` defines the natural language patterns, e.g.:

```
#pred obtain_place :: 's/he may obtain a school place'.
```

- Additionally, we can obtain a readable code in NL by invoking `scasp --code --human`.

StudentXX.pl

- Last module in Fig. 2 encodes the evidences of a student and links the previous modules.
- The code `XX` corresponds to the 'id' of each student (from 01 to 06).
- Table 1 shows the data corresponding to the candidates and the conclusion generated by s(LAW) for the query `?-obtain_place`.
  - Students 01, 03, 04, and 05 obtain a place at the school while students 02 and 06 do not.

s(LAW): The framework - ArticleESO.pl

```
obtain_place :-
    met_requirement,
    not exception.
met_requirement :-
    met_common_requirement,
    met_specific_requirement.
%% Common requirements:
met_common_requirement :-
    large_family.

met_common_requirement :-
    recipient_social_benefits.
recipient_social_benefits :-
    renta_minima_insercion.
recipient_social_benefits :-
    ingreso_minimo_vital.

met_common_requirement :-
    disability_status.
disability_status :-
    disabled_parent.
disability_status :-
    disabled_sibling.
%%  Specific requirements:
met_specific_requirement :-
    sibling_enroll_center.
met_specific_requirement :-
    legal_guardian_work_center.
met_specific_requirement :-
    relative_former_student.
met_specific_requirement :-
    school_proximity.
school_proximity :-
    same_education_district.
school_proximity :-
    not same_education_district,
    force_majeure.    % Ambiguity

force_majeure :-
    not n_force_majeure.
n_force_majeure :-
    not force_majeure.
%% Exceptions:
exception :-
    come_non_bilingual,
    want_bilingual_section(Course),
    not accredit_english_level(Course).

accredit_english_level('1st ESO') :-
    b1_certificate.
accredit_english_level('2nd ESO') :-
    b1_certificate.
accredit_english_level('3rd ESO') :-
    b2_certificate.
accredit_english_level('4th ESO') :-
    b2_certificate.
%% Discretion To Act:
obtain_place :-
    not met_requirement,
    met_complementary_criterion(CC).
obtain_place :-
    met_requirement, exception,
    met_complementary_criterion(CC).

met_complementary_criterion(CC) :-
    school_criteria(CC),
    purpose(CC), not unlawful(CC),
    not n_met_complementary_criterion(CC).
n_nmet_complementary_criterion(CC) :-
    not met_complementary_criterion(CC).

purpose(CC) :- promote_diversity(CC).
unlawful(CC) :- sex_discrimination(CC).
unlawful(CC) :- race_discrimination(CC).
unlawful(CC) :- religion_discrimination(CC).

school_criteria(foreign_student) :-
    foreign_student.
school_criteria(specific_etnia) :-
    specific_etnia.
promote_diversity(foreign_student).
promote_diversity(specific_etnia).
race_discrimination(specific_etnia).
```

Figure: Translation of the procedure for awarding school places under s(LAW).

s(LAW): The framework - Student01.pl

```
#include('ArticleESO.pl').
#include('ArticleESO.pred.pl').

come_non_bilingual.
want_bilingual_section('2nd ESO').

evidence(large_family).
evidence(renta_minima_insercion).
evidence(sibling_enroll_center).
evidence(same_education_district).
evidence(b1_certificate).
-evidence(foreign_student).
-evidence(specific_etnia).
```

Figure: Data of student 01.

Evaluation: Reasoning and Deduction with Real Use-Cases

Table: Case of different students evaluated using s(LAW).
Note: '+' is a positive evidence, '−' is a negative evidence, '?' means unkown.

	Student01	Student02	Student03	Student04	Student05	Student06
`large_family`	+	+	+	−	−	−
`renta_minima_insercion`	+	+	+	?	−	−
`sibling_enroll_center`	+	+	−	+	−	−
`same_education_district`	+	+	−	+	−	−
`b1_certificate`	+	−	+	?	−	−
`foreign_student`	−	−	−	−	+	−
`specific_etnia`	−	−	−	−	−	+
`?- obtain_place`	**yes**	no	**yes**	**yes**	**yes**	no

Evaluation: A Priori Deduction

- Student 01: S/he meets the requirements and has B1:

```
{ obtain_place, large_family, sibling_enroll_center, come_non_bilingual,
  want_bilingual_section(2nd ESO), b1_certificate }
```

Evaluation: A Priori Deduction

- Student 01: S/he meets the requirements and has B1:

```
{ obtain_place, large_family, sibling_enroll_center, come_non_bilingual,
  want_bilingual_section(2nd ESO), b1_certificate }
```

... and the corresponding justification in NL.

```
s/he may obtain a school place, because
    a common requirement is met, because
        s/he is part of a large family.
    a specific requirement is met, because
        s/he has siblings enrolled in the center.
    there is no evidence that an exception applies, because
        s/he came from a non-bilingual public school, and
        s/he wish to study 2nd ESO in the Bilingual Section, and
        s/he accredit required level of English for 2nd ESO, because
            in the four skills certificate level b1.
```

Figure: Justification in Natural Language for the evaluation of `student01.pl`.

Evaluation: A Posteriori Deduction

- Student 06: Justification for `?-not obtain_place`.

Evaluation: A Posteriori Deduction

- Student 06: Justification for `?-not obtain_place`.

```
there is no evidence that s/he may obtain a school place, because
    there is no evidence that a common requirement is met, because
        there is no evidence that s/he is part of a large family, and
        there is no evidence that s/he is a recipient of the RMI, and
        there is no evidence that a parent or sibling has disability status.
    there is no evidence that the criterion foreign_student is met, because
        there is no evidence that s/he meets the criteria foreign_student, because
            there is no evidence that s/he is a foreign student.
    there is no evidence that the criterion specific_etnia is met, because
        s/he meets the criteria specific_etnia, because
            s/he belongs to a specific etnia.
        specific_etnia follows the purpose of the procedure, because
            specific_etnia promotes the diversity.
        specific_etnia is illegal, because
            specific_etnia discriminates based on race.
```

Figure: Justification in Natural Language for the evaluation of `student06.pl`.

Conclusions

- s(LAW) is capable of modeling discretion to act, ambiguity and unknown information.
  - It exhibits the property of modelling vague concepts.

Conclusions

- s(LAW) is capable of modeling discretion to act, ambiguity and unknown information.
  - It exhibits the property of modelling vague concepts.
- The deduction based on s(LAW) allows:
  - The consideration of different conclusions (multiple models):
  - The reasoning about the set of these conclusions/models.

Conclusions

- s(LAW) is capable of modeling discretion to act, ambiguity and unknown information.
  - It exhibits the property of modelling vague concepts.
- The deduction based on s(LAW) allows:
  - The consideration of different conclusions (multiple models):
  - The reasoning about the set of these conclusions/models.

Future work

- Modeling the complete legislation by tabulation for the criteria.
- Exploit the underlying constraint solver of s(CASP).
- Extend s(LAW) considering "Epistemic Specifications" [15]:
  - What is true in all/some models, models sharing assumptions...

Building Information Modeling Using Constraint Logic Programming.
Joaquín Arias, Seppo Törmä, Manuel Carro, and Gopal Gupta (2022).

Spatial Reasoner: Introduction [Arias et al. 2022]

- Building Information Modeling (BIM) represents the 3D geometry and properties (costs, materials, process, etc.), of buildings as digital models.
- For each building, architects and engineers create specifics models.
  - These specifics models must be shared and combined.
  - Automated tools are needed to check the integrity of the merged model.
  - In addition, the models must comply with building regulations.
- The design and construction of a building is a sequence of decisions.

Spatial Reasoner: Introduction [Arias et al. 2022]

- Building Information Modeling (BIM) represents the 3D geometry and properties (costs, materials, process, etc.), of buildings as digital models.
- For each building, architects and engineers create specifics models.
  - These specifics models must be shared and combined.
  - Automated tools are needed to check the integrity of the merged model.
  - In addition, the models must comply with building regulations.
- The design and construction of a building is a sequence of decisions.

Automated BIM tools must:

- Combine geometrical reasoning and symbolic/conceptual knowledge.
- Reason in presence of vague concepts and incomplete information.
- Deal with the ambiguity present in regulatory codes and standards.

Spatial Reasoner for Building Information Modeling

Logic programming-based tools meet many of the requirements:

- The following examples overcome some limitations of IFC-based tools:
  - The query language, BimSPARQL [44].
  - Model checkers for safety [45] or acoustic rules [31], and BIMRL [40].
  - A translator of building regulation, KBimCode [21].
  - A tool based on *clingo*, ASP4BIM [25].
- However, they have limitations in meeting all requirements.

Spatial Reasoner for Building Information Modeling

Logic programming-based tools meet many of the requirements:

- The following examples overcome some limitations of IFC-based tools:
  - The query language, BimSPARQL [44].
  - Model checkers for safety [45] or acoustic rules [31], and BIMRL [40].
  - A translator of building regulation, KBimCode [21].
  - A tool based on *clingo*, ASP4BIM [25].
- However, they have limitations in meeting all requirements.

Our Proposal: Spatial Reasoner

- Use tools integrating Constraint Logic Programming with ASP to model dynamic information and restrictions in BIM models.
- Shift from BIM verification to BIM refinement and to facilitate the implementation of new specifications, construction standards, etc.

Spatial Reasoner under s(CASP): Contributions

- A framework, based on Constraint Answer Set Programming (CASP), that allows unified geometrical and non-geometrical information.
- The prototype of a preliminary 3D reasoner under Prolog with CLP(Q/R) that we evaluate with several BIM models.
- The outline of an alternative implementation of this spatial reasoner under CASP, using s(CASP) [3], a goal-directed implementation.

Spatial Reasoner under s(CASP): Contributions

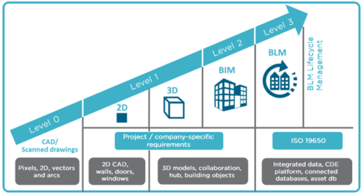
- A framework, based on Constraint Answer Set Programming (CASP), that allows unified geometrical and non-geometrical information.
- The prototype of a preliminary 3D reasoner under Prolog with CLP(Q/R) that we evaluate with several BIM models.
- The outline of an alternative implementation of this spatial reasoner under CASP, using s(CASP) [3], a goal-directed implementation.

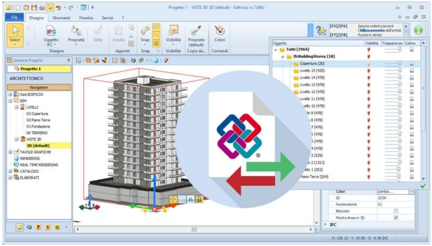
Evidence of advantages of s(CASP) in evaluating BIM models:

- It has the *relevance* property,
- It can generate justifications for negative queries, and
- It makes representing and reasoning with ambiguities easier.

Spatial Reasoner: Background, BIM + IFC

- Building information modeling (BIM):
  - Combine geometrical information with: costs, materials, process, etc.
  - Allow cost estimations, quantify takeoffs, energy analysis, etc.
  - Goal: achieve consistent of digital models:
    - shared with architects, engineers...
    - throughout the life cycle of a facility.
- The UK Government requires Level 2 of BIM maturity for any public project.
- BIM authoring tools: Revit, ArchiCAD, Tekla Structures, Allplan...
- Common data model: Industry Foundation Classes (IFC) [9].





Modeling and manipulating 3D objects: CLP(Q/R)

- Convex shapes are represented using linear equations.
- CLP(Q/R) [20] can be used to solve the resulting linear constraints.
- Using CLP(Q/R) objects are represented as a list of convex shapes:

```
box(point(Xa,Ya,Za), point(Xb,Yb,Zb), [convex([X,Y,Z])]) :-
    X#>=Xa, X#<Xb, Y#>=Ya, Y#<Yb, Z#>=Za, Z#<Zb.
```

Modeling and manipulating 3D objects: CLP(Q/R)

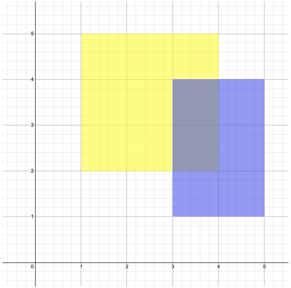
- Convex shapes are represented using linear equations.
- CLP(Q/R) [20] can be used to solve the resulting linear constraints.
- Using CLP(Q/R) objects are represented as a list of convex shapes:

```
box(point(Xa,Ya,Za), point(Xb,Yb,Zb), [convex([X,Y,Z])]) :-
    X#>=Xa, X#<Xb, Y#>=Ya, Y#<Yb, Z#>=Za, Z#<Zb.
```

- Operations: union, intersection, complement, and subtraction:

```
obj(r1, [convex([X,Y])]) :- X#>=1, X#<4, Y#>=2, Y#<5.
obj(r2, [convex([X,Y])]) :- X#>=3, X#<5, Y#>=1, Y#<4.
```

```
?- obj(r1,Sh1), obj(r2,Sh2),sh_intersection(Sh1, Sh2, Int).
   Int = [convex([A,B])], A#>=3, A#<4, B#>=2, B#<4 ?
?- obj(r1,Sh1), obj(r2,Sh2),sh_subtraction(Sh1, Sh2, Sub).
   Sub = [convex([A,B]),convex([C,D])],
         A#>=1,A#<3,B#>=2,B#<5, C#>=3,C#<4,D#>=4,D#<5 ?
```



Modeling and manipulating 3D objects: s(CASP) I

- The representation of the convex shapes are part of the program:

```
convex(r1, X, Y) :- X#>=1, X#<4, Y#>=2, Y#<5.
convex(r2, X, Y) :- X#>=3, X#<5, Y#>=1, Y#<4.
```

- They are handled as part of the constraint store of the program.
- A non-convex object is represented with several clauses, one for each convex shape:

Modeling and manipulating 3D objects: s(CASP) I

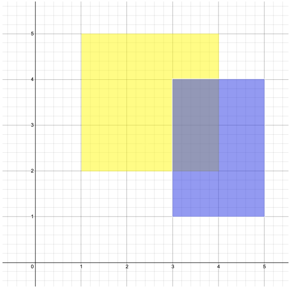
- The representation of the convex shapes are part of the program:

```
convex(r1, X, Y) :- X#>=1, X#<4, Y#>=2, Y#<5.
convex(r2, X, Y) :- X#>=3, X#<5, Y#>=1, Y#<4.
```

- They are handled as part of the constraint store of the program.
- A non-convex object is represented with several clauses, one for each convex shape:

```
?- shape_intersect(r1,r2,Int).
   Int = convex([A | { A#>=3, A#<4 }, B | { B#>=2, B#<4 }]) ?

?- shape_subtract(r1,r2,Sub).
   Sub = convex([A | { A#>=1, A#<3 }, B | { B#>=2, B#<5 }]) ? ;
   Sub = convex([A | { A#>=3, A#<4 }, B | { B#>=4, B#<5 }]) ?
```



Modeling and manipulating 3D objects: s(CASP) II

Operations on a 2D space using s(CASP)

```
% Union = ShA ∪ ShB
shape_union(IdA, IdB, convex([X,Y])) :-  convex(IdA,X,Y).
shape_union(IdA, IdB, convex([X,Y])) :-  convex(IdB,X,Y).
% Intersection = ShA ∩ ShB
shape_intersect(IdA, IdB, convex([X,Y])) :- convex(IdA,X,Y), convex(IdB,X,Y).
% Complement = ¬ ShA
shape_complement(IdA, convex([X,Y])) :- not convex(IdA,X,Y).
% Subtract = ShA ∩ ¬ ShB
shape_subtract(IdA, IdB, convex([X,Y])) :- convex(IdA,X,Y), not convex(IdB,X,Y).
```

- 9 lines of code instead of 39 lines.
- Spatial operations are translated into logical operations.

(Non)-monotonic iteration in BIM models

- Consider the design process of a room and the fire safety norm below:
  - If a gas boiler is used, the ventilation must be natural.
    window surface area is at least 10% of the floor area.
  - If the boiler is electric, the ventilation could be natural or mechanical.

(Non)-monotonic iteration in BIM models

- Consider the design process of a room and the fire safety norm below:
  - If a gas boiler is used, the ventilation must be natural.
    window surface area is at least 10% of the floor area.
  - If the boiler is electric, the ventilation could be natural or mechanical.

- Possible timeline:
  1. Initially, the shared BIM model has no ventilation or boiler restrictions.
  2. The architect reduces the size of the window (surface less that 10%).
  3. ALERT: An electric boiler is selected.
  4. At the same time, the engineer selects a gas boiler for efficiency.
  5. ALERT: Ventilation must be natural.
  6. ERROR: when attempting to merge both models, an inconsistency is detected.

(Non)-monotonic iteration in BIM models

- Consider the design process of a room and the fire safety norm below:
  - If a gas boiler is used, the ventilation must be natural.
    window surface area is at least 10% of the floor area.
  - If the boiler is electric, the ventilation could be natural or mechanical.

- Possible timeline:
  1. Initially, the shared BIM model has no ventilation or boiler restrictions.
  2. The architect reduces the size of the window (surface less that 10%).
  3. ALERT: An electric boiler is selected.
  4. At the same time, the engineer selects a gas boiler for efficiency.
  5. ALERT: Ventilation must be natural.
  6. ERROR: when attempting to merge both models, an inconsistency is detected.

- A naive approach to handle the ERROR would broadcast the inconsistency.

(Non)-monotonic iteration in BIM models

- Consider the design process of a room and the fire safety norm below:
  - If a gas boiler is used, the ventilation must be natural.
    window surface area is at least 10% of the floor area.
  - If the boiler is electric, the ventilation could be natural or mechanical.

- Possible timeline:
  1. Initially, the shared BIM model has no ventilation or boiler restrictions.
  2. The architect reduces the size of the window (surface less that 10%).
  3. ALERT: An electric boiler is selected.
  4. At the same time, the engineer selects a gas boiler for efficiency.
  5. ALERT: Ventilation must be natural.
  6. ERROR: when attempting to merge both models, an inconsistency is detected.

- A naive approach to handle the ERROR would broadcast the inconsistency.

Proposal:

- A continuous integration reasoner:
  - that determines who is the expert whose opinion prevails
  - makes a decision based on that.
  - notifies the other party to confirm the adjustments.

(Non)-monotonic iteration in BIM models (cont.)

Encoding of a Continuous Integration reasoner

```
%% BIM Continuous Integration
valid_data(P,Data) :-
     data(P,Data),
     not canceled(P, Data).

canceled(P, Data) :-
     higher_confidence(P1, P),
     data(P1, Data1),
     inconsistent(Data, Data1).
higher_confidence(PHi, PLo) :-
     PHi.>.PLo.
%% Example
inconsistent(boiler(gas),
              ventilation(artificial)).
inconsistent(ventilation(artificial),
              boiler(gas)).
data(1,ventilation(X)).
data(2,ventilation(natural)).
data(2,boiler(gas)).
% data(3,ventilation(artificial)).
% data(3,boiler(electrical)).
% data(4,boiler(gas)).
```

Evaluation I: Comprehensibility (reason in presence of vague concepts)

- Reason about various scenarios simultaneously (or not).

```
% Hotel with 8 rooms - only the size of 3 of them is known.
room(r1).       room(r2).       room(r3).       room(r4).
room(r5).       room(r6).       room(r7).       room(r8).
size(r1, 25).          size(r2, 5).          size(r3, 15).
% Uncertain whether rooms of 10 to 20 m2 are small or not.
evidence(Room, small) :- size(Room,Size), Size#<10.
-evidence(Room, small) :- size(Room,Size), Size#>20.
% Explicit evidence for / against or generate two models.
small(Room) :- evidence(Room,small).
-small(Room) :- -evidence(Room,small).
small(Room) :- not evidence(Room,small), not -small(Room).
-small(Room) :- not -evidence(Room,small), not small(Room).
% Inferring conclusions from evidence and/or assumptions.
room_is(Room,big) :- room(Room), -small(Room).
room_is(Room,small) :- room(Room), small(Room).
```

Evaluation I: Comprehensibility (reason in presence of vague concepts)

- Reason about various scenarios simultaneously (or not).

```
% Hotel with 8 rooms - only the size of 3 of them is known.
room(r1).       room(r2).       room(r3).       room(r4).
room(r5).       room(r6).       room(r7).       room(r8).
size(r1, 25).         size(r2, 5).          size(r3, 15).
% Uncertain whether rooms of 10 to 20 m2 are small or not.
evidence(Room, small) :- size(Room,Size), Size#<10.
-evidence(Room, small) :- size(Room,Size), Size#>20.
% Explicit evidence for / against or generate two models.
small(Room) :- evidence(Room,small).
-small(Room) :- -evidence(Room,small).
small(Room) :- not evidence(Room,small), not -small(Room).
-small(Room) :- not -evidence(Room,small), not small(Room).
% Inferring conclusions from evidence and/or assumptions.
room_is(Room,big) :- room(Room), -small(Room).
room_is(Room,small) :- room(Room), small(Room).
```

- For `?- room_is(Room,Size)`:
  - s(CASP) returns **14** partial models.
  - clingo returns **64** models.

Evaluation I: Comprehensibility (reason in presence of vague concepts)

- Reason about various scenarios simultaneously (or not).

```
% Hotel with 8 rooms - only the size of 3 of them is known.
room(r1).      room(r2).      room(r3).      room(r4).
room(r5).      room(r6).      room(r7).      room(r8).
size(r1, 25).         size(r2, 5).         size(r3, 15).
% Uncertain whether rooms of 10 to 20 m2 are small or not.
evidence(Room, small) :- size(Room,Size), Size#<10.
-evidence(Room, small) :- size(Room,Size), Size#>20.
% Explicit evidence for / against or generate two models.
small(Room) :- evidence(Room,small).
-small(Room) :- -evidence(Room,small).
small(Room) :- not evidence(Room,small), not -small(Room).
-small(Room) :- not -evidence(Room,small), not small(Room).
% Inferring conclusions from evidence and/or assumptions.
room_is(Room,big) :- room(Room), -small(Room).
room_is(Room,small) :- room(Room), small(Room).
```

- For `?- room_is(Room,Size)`:
  - s(CASP) returns **14** partial models.
  - clingo returns **64** models.

Considering 16 rooms

- s(CASP) returns **30** partial models.
- clingo returns **16384** models.

Evaluation I: Comprehensibility (reason in presence of vague concepts)

- Reason about various scenarios simultaneously (or not).

```
% Hotel with 8 rooms - only the size of 3 of them is known.
room(r1).      room(r2).      room(r3).      room(r4).
room(r5).      room(r6).      room(r7).      room(r8).
size(r1, 25).          size(r2, 5).          size(r3, 15).
% Uncertain whether rooms of 10 to 20 m2 are small or not.
evidence(Room, small) :- size(Room,Size), Size#<10.
-evidence(Room, small) :- size(Room,Size), Size#>20.
% Explicit evidence for / against or generate two models.
small(Room) :- evidence(Room,small).
-small(Room) :- -evidence(Room,small).
small(Room) :- not evidence(Room,small), not -small(Room).
-small(Room) :- not -evidence(Room,small), not small(Room).
% Inferring conclusions from evidence and/or assumptions.
room_is(Room,big) :- room(Room), -small(Room).
room_is(Room,small) :- room(Room), small(Room).
```

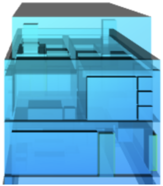
- For `?- room_is(Room,Size)`:
  - s(CASP) returns **14** partial models.
  - clingo returns **64** models.

Considering 16 rooms

- s(CASP) returns **30** partial models.
- clingo returns **16384** models.

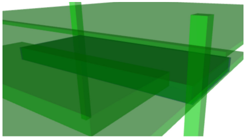
This exponential explosion in the # models generated by clingo...
... reduces the comprehensibility.

Evaluation II: Geometrical and non-geometrical information.



(a) Duplex_Q1.html

(b) Duplex_Q2.html

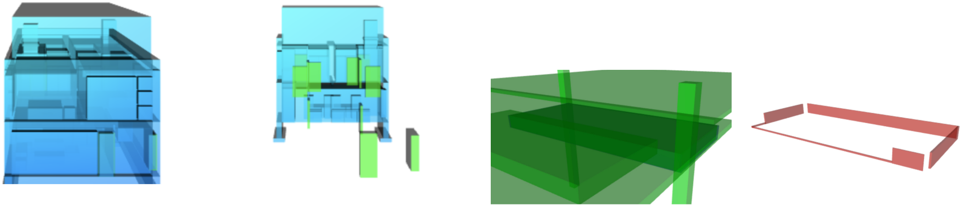


(c) Office_Q1.html

(d) Office_Q2.html

Query Duplex_Q1: The doors are in green and the rest of the objects are in blue.

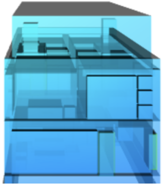
Evaluation II: Geometrical and non-geometrical information.



(a) Duplex_Q1.html (b) Duplex_Q2.html (c) Office_Q1.html (d) Office_Q2.html

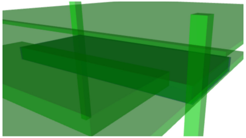
Query Duplex_Q2: imposes the constraints `Ya#<-4` to select certain doors, and `Y#>=-7, Y#<-4` to create a space (unbounded in the axis x and z) that defines a *slice* of the model.

Evaluation II: Geometrical and non-geometrical information.



(a) Duplex_Q1.html

(b) Duplex_Q2.html



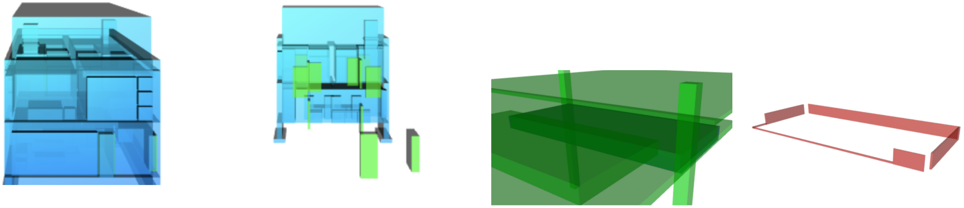
(c) Office_Q1.html

(d) Office_Q2.html

Constraints can be used in s(CASP) to reason about unbounded spaces

- Finer constraints, such as `Ya#<-4.002`, can be used without performance impact.
- That is in general not the case with other ASP systems.

Evaluation II: Geometrical and non-geometrical information.



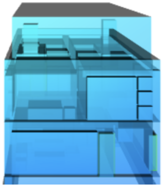
(a) `Duplex_Q1.html` (b) `Duplex_Q2.html` (c) `Office_Q1.html` (d) `Office_Q2.html`

Query Office_Q1/Q2: selects objects of type *IfcBeam* in the Architecture model that are not covered by objects in the Structural BIM model.

(c) shows the objects that intersect the beam.

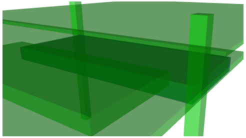
(d) shows the uncovered parts drawn in red.

Evaluation II: Geometrical and non-geometrical information.



(a) Duplex_Q1.html

(b) Duplex_Q2.html



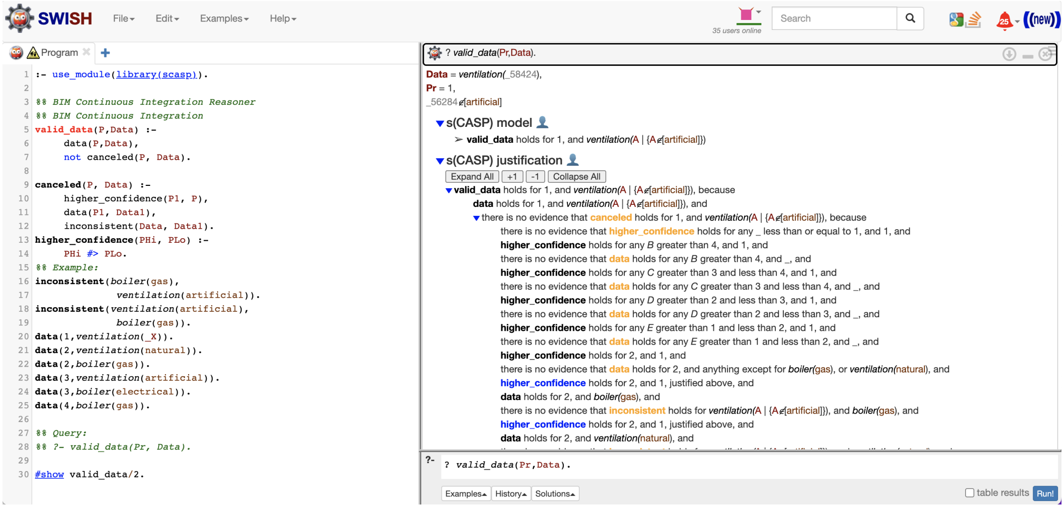
(c) Office_Q1.html

(d) Office_Q2.html

Performance Query Office_Q1/Q2

- Finds the first beam with uncovered parts in only 0.104 sec.
- Selects 691 beams out of 3639 objects in the architecture model and detected 511 beams not covered by the more than 1300 objects in the structure model in 48 sec.

Evaluation III: Explainability of the inferred conclusions.



Evaluation III: Explainability of the inferred conclusions.

playground for s(CASP)

Embed Docs ? Star 135

New Open Save Examples Load ▶ Share!

```
%% BIM Continuous Integration Reasoner
%% BIM Continuous Integration
valid_data(P,Data) :-
      data(P,Data),
      not canceled(P, Data).

canceled(P, Data) :-
      higher_confidence(P1, P),
      data(P1, Data1),
      inconsistent(Data, Data1).
higher_confidence(PHi, PLo) :-
      PHi.>.PLo.
%% Example
inconsistent(boiler(gas),
             ventilation(artificial)).
inconsistent(ventilation(artificial),
             boiler(gas)).
data(1,ventilation(X)).
data(2,ventilation(natural)).
data(2,boiler(gas)).
data(3,ventilation(artificial)).
data(3,boiler(electrical)).
data(4,boiler(gas)).

%% ?- valid_data(Pr, Data).

#show valid_data/2.
```

```
?- load('/draft.pl').

yes
?- valid_data(Pr,Data).

{ valid_data(1,ventilation(Var2 | {Var2 \= artificial})) }
Pr = 1
Data = ventilation(Var2 | {Var2 \= artificial}) ?

yes
?-
```

Justification: Expand All +1 -1 Collapse All

```
▾ valid_data(1,ventilation(Var2 | {Var2 \= artificial})) :-
    data(1,ventilation(Var2 | {Var2 \= artificial})),
  ▾ not canceled(1,ventilation(Var2 | {Var2 \= artificial})) :-
    ▾ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial})) :-
      ▾ forall(Var15,forall(Var16,not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),Var15,Var16))) :-
        ▸ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),Var3 | {Var3 #=< 1},Var17) :-
        ▸ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),Var18 | {Var18 #> 4},Var5) :-
        ▸ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),Var21 | {Var21 #> 3,Var21 #< 4},Var7) :-
        ▸ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),Var24 | {Var24 #> 2,Var24 #< 3},Var9) :-
        ▸ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),Var27 | {Var27 #> 1,Var27 #< 2},Var11) :-
        ▸ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),2,Var12 | {Var12 \= boiler(gas),Var12 \=
          ventilation(natural)}) :-
        ▸ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),2,boiler(gas)) :-
        ▸ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),2,ventilation(natural)) :-
        ▸ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),3,Var13 | {Var13 \= boiler(electrical),Var13 \=
          ventilation(artificial)}) :-
        ▸ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),3,boiler(electrical)) :-
        ▸ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),3,ventilation(artificial)) :-
        ▸ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),4,Var14 | {Var14 \= boiler(gas)}) :-
        ▸ not o_canceled_1(1,ventilation(Var2 | {Var2 \= artificial}),4,boiler(gas)) :-
global_constraint.
```

Conclusions

- We have highlighted the advantages of a well-founded approach to:
    - Represent geometrical and non-geometrical building information (including specifications, codes, and/or guidelines) as digital models.
    - Handle changes to the models during their design, construction, and/or facility time (removing, adding, or changing objects and properties).
- The use of CLP, and more specifically s(CASP), makes it possible to:
    - Realize commonsense reasoning including geometrical data.
    - Represent knowledge involving vague and/or unknown information.

Conclusions

- We have highlighted the advantages of a well-founded approach to:
  - Represent geometrical and non-geometrical building information (including specifications, codes, and/or guidelines) as digital models.
  - Handle changes to the models during their design, construction, and/or facility time (removing, adding, or changing objects and properties).
- The use of CLP, and more specifically s(CASP), makes it possible to:
  - Realize commonsense reasoning including geometrical data.
  - Represent knowledge involving vague and/or unknown information.

Future work

- Shift from BIM verification to BIM refinement.
- Develop non-monotonic model refinement methods.
- Integrate logical reasoning in BIM Software.

Explainable AI w/ Default Rules: FOLD Family of Algorithms.

Gopal Gupta's lab (specially Farhad Shakerin, and Huaduo Wang).

Explainable AI w/ Default Rules

- Idea is to learn rules from data, i.e., express patterns in data as rules.
- These rules are represented as default rules with exceptions.
- Since we use default rules, our representation of knowledge learned is very close to how humans will represent, understand and learn patterns from data.
- Many advantages:
  - Unlike other Machine Learning techniques, distinguishes noise from exception.
  - Represents concepts with fewer number of rules.
  - Supported by a powerful formalism (Answer Set Programming).

FOLD Family of Algorithms [Gupta et al. 2023]

- FOLD family of algorithms are distinct machine learning algorithms
  - FOLD, FOLD-R, LIME-FOLD, SHAP-FOLD... [Shakerin and Gupta 2019]
- FOLD-R++ algorithm: Performs binary classification [Wang and Gupta 2022a]
  - both categorical and numerical data
  - no data prep required, uses prefix sum computation for fast execution;
  - competitive with XGBoost and Neural Networks;
- FOLD-RM algorithm: Performs multi-category classification [Wang et al. 2022]
  - requires no data preparation; uses prefix sum computation for fast execution;
  - competitive with XGBoost and Neural Networks;
- FOLD-LTR: Learning to rank, but explainable

FOLD-SE

Improved FOLD-R++ and FOLD-RM with scalable interpretability. [Wang and Gupta 2022b]

Current s(CASP) applications/users

- Legal Industry.
- Architecture, Engineering and Construction.
- Healthcare Industry.
- Avionics Domain.

...more are still to come.

Current s(CASP) applications/users

- Legal Industry.
- Architecture, Engineering and Construction.
- Healthcare Industry.
- Avionics Domain.

...more are still to come.

THANKS!

Bibliography I

[1] Arias, J., Chen, Z., Carro, M., and Gupta, G. (2019). **Modeling and Reasoning in Event Calculus Using Goal-Directed Constraint Answer Set Programming**. In: *Pre-Proc. of the 29th Int'l. Symposium on Logic-based Program Synthesis and Transformation*.

[2] Arias, Joaquín, Carro, Manuel, Chen, Zhuo, and Gupta, Gopal (2020). **Justifications for Goal-Directed Constraint Answer Set Programming**. In: *Proceedings 36th International Conference on Logic Programming (Technical Communications)*. Vol. 325. EPTCS. Open Publishing Association, pp. 59–72. DOI: 10.4204/EPTCS.325.12.

[3] Arias, Joaquín, Carro, Manuel, Salazar, Elmer, Marple, Kyle, and Gupta, Gopal (2018). **Constraint Answer Set Programming without Grounding**. In: *Theory and Practice of Logic Programming* 18.3-4, pp. 337–354. DOI: 10.1017/S1471068418000285.

Bibliography II

[4] Arias, Joaquín, Moreno-Rebato, Mar, Rodriguez-García, Jose A., and Ossowski, Sascha (2021). **Modeling Administrative Discretion Using Goal-Directed Answer Set Programming**. In: *Advances in Artificial Intelligence, CAEPIA 20/21*. Vol. 12882. LNCS. Springer, pp. 258–267. DOI: 10.1007/978-3-030-85713-4_25.

[5] Arias, Joaquín, Törmä, Seppo, Carro, Manuel, and Gupta, Gopal (2022). **Building Information Modeling Using Constraint Logic Programming**. In: *Theory and Practice of Logic Programming* 22.5, pp. 723–738. DOI: 10.1017/S1471068422000138. URL: https://doi.org/10.1017/S1471068422000138.

[6] Balduccini, Marcello, Magazzeni, Daniele, and Maratea, Marco (2016). **PDDL+ Planning via Constraint Answer Set Programming**. In: *9th Workshop on Answer Set Programming and Other Computing Paradigms*. http://arxiv.org/abs/1609.00030.

Bibliography III

[7] Basu, Kinjal, Varanasi, Sarat, Shakerin, Farhad, Arias, Joaquin, and Gupta, Gopal (2021a). **Knowledge-driven Natural Language Understanding of English Text and its Applications**. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 14, pp. 12554–12563. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17488.

[8] Basu, Kinjal, Wang, Huaduo, Dominguez, Nancy, Li, Xiangci, Li, Fang, Varanasi, Sarat Chandra, and Gupta, Gopal (2021b). **CASPR: A Commonsense Reasoning-based Conversational Socialbot**. In: *4th Proceedings of Alexa Prize (Alexa Prize 2021)*.

[9] BuildingSMART (2020). **Industry Foundation Classes (IFC)**. Available at: https://technical.buildingsmart.org/standards/ifc/. Accessed on July, 2020.

[10] Cabalar, Pedro, Fandinno, Jorge, and Fink, Michael (2014). **Causal Graph Justifications of Logic Programs**. In: *Theory and Practice of Logic Programming* 14.4-5, pp. 603–618. DOI: 10.1017/S1471068414000234.

Bibliography IV

[11] Chen, Zhuo, Marple, Kyle, Salazar, Elmer, Gupta, Gopal, and Tamil, Lakshman (2016). **A Physician Advisory System for Chronic Heart Failure Management Based on Knowledge Patterns**. In: *Theory and Practice of Logic Programming* 16.5-6, pp. 604–618. DOI: 10.1017/S1471068416000429.

[12] Chittaro, Luca and Montanari, Angelo (1996). **Efficient Temporal Reasoning in the Cached Event Calculus**. In: *Computational Intelligence* 12, pp. 359–382. DOI: 10.1111/j.1467-8640.1996.tb00267.x.

[13] Cobbe, Jennifer (2019). **Administrative law and the machines of government: judicial review of automated public-sector decision-making**. In: *Legal Studies* 39.4, pp. 636–655.

[14] Fox, Maria and Long, Derek (2002). **PDDL+: Modeling continuous time dependent effects**. In: *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*. Vol. 4, p. 34.

Bibliography V

[15] Gelfond, Michael (1994). **Logic programming and reasoning with incomplete information**. In: *Annals of mathematics and artificial intelligence* 12.1, pp. 89–116.

[16] Gelfond, Michael and Lifschitz, Vladimir (1988). **The Stable Model Semantics for Logic Programming**. In: *5th International Conference on Logic Programming*, pp. 1070–1080. DOI: 10.2307/2275201. URL: http://www.cse.unsw.edu.au/~cs4415/2010/resources/stable.pdf.

[17] — (1993). **Representing Action and Change by Logic Programs**. In: *The Journal of Logic Programming* 17.2-4, pp. 301–321.

[18] Gupta, Gopal, Wang, Huaduo, Basu, Kinjal, Shakerin, Farhad, Salazar, Elmer, Varanasi, Sarat Chandra, Padalkar, Parth, and Dasgupta, Sopam (2023). **Logic-based explainable and incremental machine learning**. In: *Prolog: The Next 50 Years*. Springer, pp. 346–358.

Bibliography VI

[19] Hall, Brendan, Varanasi, Sarat Chandra, Fiedor, Jan, Arias, Joaquín, Basu, Kinjal, Li, Fang, Bhatt, Devesh, Driscoll, Kevin, Salazar, Elmer, and Gupta, Gopal (2021). **Knowledge-Assisted Reasoning of Model-Augmented System Requirements with Event Calculus and Goal-Directed Answer Set Programming**. In: *Proc. 8th Workshop on Horn Clause Verification and Synthesis*.

[20] Holzbaur, C. (1995). **OFAI CLP(Q,R) Manual, Edition 1.3.3**. Tech. rep. TR-95-09. Vienna: Austrian Research Institute for Artificial Intelligence.

[21] Lee, H., Lee, J.K., Park, S., and Kim, I. (2016). **Translating building legislation into a computer-executable format for evaluating building permit requirements**. en. In: *Automation in Construction* 71, pp. 49–61. DOI: 10.1016/j.autcon.2016.04.008.

[22] Lee, Joohyung and Meng, Yunsong (2013). **Answer Set Programming Modulo Theories and Reasoning about Continuous Changes**. In: *23rd Int'l. Joint Conference on Artificial Intelligence*, pp. 990–996.

Bibliography VII

[23] Lee, Joohyung and Palla, Ravi (2009). **System F2LP – Computing Answer Sets of First-Order Formulas**. In: *Logic Programming and Nonmonotonic Reasoning*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 515–521. DOI: 10.1007/978-3-642-04238-6_51.

[24] — (2012). **Reformulating the Situation Calculus and the Event Calculus in the General Theory of Stable Models and in Answer Set Programming**. In: *Journal of Artificial Intelligence Research* 43, pp. 571–620.

[25] Li, Beidi, Teizer, Jochen, and Schultz, Carl (2020). **Non-monotonic Spatial Reasoning for Safety Analysis in Construction**. In: *Proceedings of the 22nd International Symposium on Principles and Practice of Declarative Programming*, pp. 1–12.

[26] Marple, Kyle, Salazar, Elmer, and Gupta, Gopal (2017). **Computing Stable Models of Normal Logic Programs Without Grounding**. In: *arXiv* 1709.00501. arXiv: 1709.00501. URL: http://arxiv.org/abs/1709.00501.

Bibliography VIII

[27] Martínez, Agustí Cerrillo i (2019). **El derecho para una inteligencia artificial centrada en el ser humano y al servicio de las instituciones: Presentación del monográfico.** In: *IDP: Revista de Internet, Derecho y Politica* 30.

[28] Morris, Jason (2021). **Constraint answer set programming as a tool to improve legislative drafting: a rules as code experiment**. In: *ICAIL*. ACM, pp. 262–263.

[29] Moyle, Steve, Allott, Nicholas, and Manslow, John (2023). **Modelling Cyber Defenses using s (CASP)**. In: *ICLP'23 Workshop on Goal-directed Execution of Answer Set Programs*.

[30] Mueller, Erik T. (2008). **Chapter 17: Event Calculus**. In: *Handbook of Knowledge Representation*. Ed. by Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. Vol. 3. Foundations of AI. Elsevier, pp. 671 –708. DOI: 10.1016/S1574-6526(07)03017-9.

Bibliography IX

[31] Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van De Walle, R., and Van Campenhout, J. (2011). **A semantic rule checking environment for building performance checking**. en. In: *Automation in Construction* 20.5, pp. 506–518. DOI: 10.1016/j.autcon.2010.11.017.

[32] Pontelli, Enrico, Son, Tran Cao, and El-Khatib, Omar (2009). **Justifications for Logic Programs under Answer Set Semantics**. In: *Theory and Practice of Logic Programming* 9.1, pp. 1–56. DOI: 10.1017/S1471068408003633.

[33] Ramakrishna, Shashishekar, Górski, Łukasz, and Paschke, Adrian (2016). **A dialogue between a lawyer and computer scientist: the evaluation of knowledge transformation from legal text to computer-readable format**. In: *Applied Artificial Intelligence* 30.3, pp. 216–232.

[34] Schulz, Claudia and Toni, Francesca (2016). **Justifying Answer Sets Using Argumentation**. In: *Theory and Practice of Logic Programming* 16.1, pp. 59–110. DOI: 10.1017/S1471068414000702.

Bibliography X

[35] Sergot, Marek J., Sadri, Fariba, Kowalski, Robert A., Kriwaczek, Frank, Hammond, Peter, and Cory, H Terese (1986). **The British Nationality Act as a logic program**. In: *Communications of the ACM* 29.5, pp. 370–386.

[36] Shakerin, Farhad and Gupta, Gopal (2019). **Induction of Non-Monotonic Logic Programs to Explain Boosted Tree Models Using LIME**. In: *AAAI 2019*, pp. 3052–3059. DOI: 10.1609/aaai.v33i01.33013052.

[37] Shanahan, Murray (1999). **The Event Calculus Explained**. In: *Artificial Intelligence Today*. Springer, pp. 409–430. DOI: 10.1007/3-540-48317-9_17.

[38] — (2000). **An Abductive Event Calculus Planner**. In: *The Journal of Logic Programming* 44.1-3, pp. 207–240.

[39] Solé, Juli Ponce (2019). **Inteligencia artificial, Derecho administrativo y reserva de humanidad: algoritmos y procedimiento administrativo debido tecnológico**. In: *Revista general de Derecho administrativo* 50.

Bibliography XI

[40] Solihin, Wawan (2015). **A simplified BIM data representation using a relational database schema for an efficient rule checking system and its associated rule checking language**. PhD thesis. Georgia Institute of Technology.

[41] Wang, Huaduo and Gupta, Gopal (2022a). **FOLD-R++: A Scalable Toolset for Automated Inductive Learning of Default Theories from Mixed Data**. In: *International Symposium on Functional and Logic Programming*. Springer, pp. 224–242. DOI: 10.1007/978-3-030-99461-7_13.

[42] — (2022b). **FOLD-SE: Scalable Explainable AI**. In: *arXiv preprint arXiv:2208.07912*.

[43] Wang, Huaduo, Shakerin, Farhad, and Gupta, Gopal (2022). **FOLD-RM: A scalable, efficient, and explainable inductive learning algorithm for multi-category classification of mixed data**. In: *Theory and Practice of Logic Programming* 22.5, pp. 658–677. DOI: 10.1017/S1471068422000205.

Bibliography XII

[44] Zhang, Chi, Beetz, Jakob, and Vries, Bauke de (2018). **BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data**. In: *Semantic Web* 9.6, pp. 829–855.

[45] Zhang, S., Teizer, J., Lee, J.K., Eastman, C.M., and Venugopal, M. (2013). **Building Information Modeling (BIM) and Safety: Automatic Safety Checking of Construction Models and Schedules**. en. In: *Automation in Construction* 29, pp. 183–195. DOI: 10.1016/j.autcon.2012.05.006.