Automating Qualitative Reasoning with s(CASP)

Gopal Gupta[1] , **Elmer Salazar**[1] , **Joaquin Arias**[2]
[1]The University of Texas at Dallas
[2]CETINIA, Universidad Rey Juan Carlos
{gopal.gupta, elmer.salazar}@utdallas.edu, joaquin.arias@urjc.es

Abstract

Automating qualitative reasoning has been an active area of research. Qualitative reasoning closely relates to human commonsense reasoning whose automation has been extensively researched since the inception of AI. Significant progress has been made in human commonsense reasoning automation and effective technologies such as answer set programming (ASP) have been developed. In this paper, we give an overview of the s(CASP) goal-directed predicate ASP system and show how it can be effectively used for qualitative reasoning.

1 Introduction

Qualitative Reasoning (QR) automates reasoning about the continuous physical world. The idea is to qualitatively reason over physical quantities such as space, time, and quantity to predict outcomes and plan. Qualitative values (e.g., high, low, fast, slow, steady) are used instead of exact numerical quantities. QR can be thought of as an instance of commonsense reasoning, and in this paper we show that solutions developed for automating human-style commonsense reasoning will also work for QR.

Humans deal with incomplete, imprecise, or uncertain scenarios in their day to day lives. Commonsense reasoning must take this incompleteness, imprecision, and uncertainty into account, which makes its automation extremely hard. Recent advances in techniques such as answer set programming (ASP), however, have resulted in considerable progress [Gelfond and Kahl, 2014; Brewka *et al.*, 2011]. Goal-directed predicate ASP systems such as s(CASP) [Arias *et al.*, 2018] have pushed the envelope even further. Our goal in this paper is to show that the s(CASP) system is an excellent vehicle for automating qualitative reasoning.

The s(CASP) system [Arias *et al.*, 2018; Arias *et al.*, 2017] supports predicates, constraints over non-ground variables, uninterpreted functions, and, most importantly, a top-down, query-driven execution strategy for ASP. The s(CASP) system supports constructive negation based on a disequality constraint solver, and unlike Prolog's negation as failure and ASP's default negation, `not p(X)` can return bindings for `X` on success, i.e., bindings for which the call `p(X)` would have failed. The s(CASP) system provides support for full Prolog, however, in addition, it also supports constraints over reals, coinductive (circular, or assumption-based) reasoning, dual rules, and support for universally quantified variables. More details can be found elsewhere [Marple *et al.*, 2017; Arias *et al.*, 2018; Arias *et al.*, 2022a]. The s(CASP) system is available freely as part of Ciao-Prolog[1] and SWI-Prolog.[2]

2 Emulating Human Thinking

Logic programming was conceived as a language for problem-solving, AI, and emulating human thinking [Kowalski, 1979; Kowalski, 2011]. However, Prolog's inability to effectively model incomplete information limited its use for AI and emulating human reasoning, which became an impetus for significant subsequent research by various groups [Gunning *et al.*, 2010; Dietz Saldanha *et al.*, 2021; Gelfond and Kahl, 2014; Kowalski, 2011]. Negation-as-failure (NAF) is an important component in these research efforts and ASP is an important effort in this direction. ASP extends logic programming with stable model semantics-based NAF [Brewka *et al.*, 2011; Gelfond and Kahl, 2014]. ASP allows reasoning with incomplete information through NAF and default rules. ASP also supports integrity constraints, and non-inductive semantics in which multiple models (worlds) are admitted. Thus, ASP is a paradigm that comes close to supporting commonsense reasoning and emulating the human thought process [Gelfond and Kahl, 2014; Gupta *et al.*, 2022]. It is our strong belief that the path to automating commonsense reasoning in a practical manner goes through Answer Set Programming realized via Prolog-like implementations of predicate ASP such as the s(CASP) system [Arias *et al.*, 2018; Marple *et al.*, 2017]. By Prolog-like, we mean that predicates and first order terms are supported, and execution is query-driven, carried out in a top-down manner. Thus, the power of ASP, i.e., negation based on stable model semantics, is supported within Prolog.

Our focus in this paper is automating qualitative reasoning. Qualitative reasoning (QR) is, essentially, a specialized form of commonsense reasoning, and we show in this paper that QR can be performed with ASP, specifically using the s(CASP) system.

[1]See https://gitlab.software.imdea.org/ciao-lang/sCASP.
[2]See https://swish.swi-prolog.org/example/scasp.swinb.

2.1 Deduction, Abduction, and Induction

A significant part of qualitative reasoning can be emulated with default rules, integrity constraints, and assumption-based reasoning [Gupta *et al.*, 2022; Gupta *et al.*, 2023]. ASP obviously supports deduction. Default rules can be viewed as inductive generalizations, and assumption-based reasoning can be viewed as abduction. Thus, the three major modes of reasoning—deduction, abduction, and induction—are naturally supported within ASP and s(CASP). Details of how deduction and abduction are supported in s(CASP) can be found elsewhere [Gupta *et al.*, 2023; Gupta *et al.*, 2022]. Inductive generalizations represented as ASP's default rules can also be automatically generated [Wang and Gupta, 2024]. Many other concepts of qualitative reasoning, such as counterfactual generation, inconsistency detection, causal reasoning, constraint relaxation, deontic modal reasoning, etc., are also representable in ASP and s(CASP) [Gupta *et al.*, 2024].

2.2 Commonsense Knowledge in s(CASP)

As stated earlier, a large portion of the human thought process can be largely emulated by supporting (i) default rules with exceptions and preferences, (ii) integrity constraints, and (iii) multiple possible worlds. As explained earlier, default rules with exceptions express inductive generalizations, and are used by humans for making deductions. Similarly, multiple possible worlds help in abduction, or assumption-based reasoning. Integrity constraints allow us to prune unfruitful paths in our abductive reasoning process. Unfruitful paths in the deductive reasoning process are pruned by adding conditions to rule bodies that can be regarded as local constraints.

Default Reasoning with Exceptions and Preferences: Humans use *default reasoning* [Gelfond and Kahl, 2014] to jump to conclusions. These conclusions may be revised later in light of new knowledge. For example, if we are told that Tweety is a bird, and then asked whether Tweety flies, we will immediately answer, yes, it does. However, later if we are told that Tweety is a penguin, we will withdraw the conclusion about Tweety's flying ability, labeling Tweety as an *exception*. Thus, human reasoning is non-monotonic in nature, meaning that conclusions may be withdrawn as new knowledge becomes available. Humans use this sort of *default reasoning* to jump to conclusions all the time, and if they find the assumptions made to jump to this conclusion to be incorrect, they revise their conclusion. Multiple default conclusions can be drawn in some situations, and humans will use additional reasoning to *prefer* one default over another. Thus, default rules with exceptions and preferences capture most of the deductive reasoning we perform. (More details on default reasoning can be found elsewhere [Gelfond and Kahl, 2014; Kowalski, 2011]). It should be noted that expert knowledge is nothing but a set of default rules about a specialized topic [Gelfond and Kahl, 2014]. This includes the knowledge needed to perform QR.

Classical logic is unable to model default reasoning and non-monotonicity in an elegant way. We need a formalism that is non-monotonic and can support defaults to model commonsense reasoning. ASP is such a formalism. ASP supports both NAF (not p) as well as *strong negation* (−p), where p is a proposition or a predicate. A strongly negated predicate has to be explicitly defined, just as positive predicates are. Combining these two forms of negations results in nuanced reasoning closer to how humans reason:

1. p: denotes that p is *definitely* true.
2. not −p: denotes that p *maybe* true (i.e., no evidence that p is false).
3. not p ∧ not −p: denotes that p is unknown (i.e., no evidence of either p or −p being true).
4. not p: denotes that p *may* be false (no evidence that p is true).
5. −p: denotes that p is *definitely* false.

The above insight can be used, for example, to model the exceptions to Tweety's ability to fly in two possible ways. Consider the rules:

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

which state that if we know nothing about a bird, X, we conclude that it flies. This is in contrast to the rules:

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

which states that a bird can fly only if we can *explicitly* rule out that it is a penguin. So in the latter case, if we know nothing about a bird, we will conclude that it *does not* fly. Which of the two rules one will use depends on how conservative or aggressive one wants to be in jumping to the (default) conclusion. We can think of aggressive *vs* conservative choice as making a judgment call, and using the appropriate corresponding rule. Note that exceptions can have exceptions, which in turn can have their own exceptions, and so on. Automation of such judgment calls is crucial for QR.Defaults, exceptions, exceptions to exceptions, and so on, allow humans to perform reasoning elegantly in an *elaboration tolerant* manner [Baral, 2003; Gelfond and Kahl, 2014].

Integrity Constraints: ASP can also model integrity constraints elegantly. An integrity constraint is a rule of the form:

$$\texttt{false} :- p_1, p_2, \ldots, p_n.$$

which states that the conjunction of p_1, p_2, through p_n is false (the keyword false is often omitted). Integrity constraints elegantly model global invariants or restrictions that our knowledge must satisfy, e.g., p and −p cannot be true at the same time, denoted as :- p, −p.

Humans indeed use integrity constraints in their everyday reasoning: as restrictions (two humans cannot occupy the same spot) and invariants (a human must breath to stay alive). Note that integrity constraints are *global* constraints, in that they eliminate possible worlds. Unfruitful paths during deductive reasoning are eliminated by adding appropriate conditions to the rule-bodies. Note that in ASP, integrity constraints may also arise due to *odd loops over negation* (OLON), i.e., rules of the form:

$$p(\vec{t}) :- G, \texttt{not } p(\vec{t}).$$

where $p(\vec{t})$ is a predicate and G is a conjunction of goals. In absence of an alternative proof for $p(\vec{t})$, the only admissible model for the above rule is $p(\vec{t})$ = false, G = false, which amounts to the global constraint that G must be false.

Note that rules in answer set programs are *completed*, i.e., an *if* is interpreted as an *iff*.

Possible Worlds: Humans can represent *multiple possible worlds* simultaneously in their minds and reason over each. For example, in the real world, birds do not talk like humans, while in a cartoon world, birds (cartoon characters) can talk. Humans can maintain the distinction between various worlds in their minds and reason within each one of them. These multiple worlds may have aspects that are common (birds can fly in both the real and cartoon worlds) and aspects that are disjoint (birds can talk only in the cartoon world). ASP and s(CASP) support multiple possible worlds.

3 Examples and Applications

Next, we give an example where we use s(CASP) for qualitative reasoning. Consider the following set of statements.

1. If it rains and traffic is heavy, the trip will be delayed.
2. If there's a delay and you leave late, you'll miss the concert.
3. If you miss the concert, everyone will be disappointed.
4. The weather forecast gives a high chance of rain.
5. Highway cameras report current traffic as moderate but predict it will become heavy if it rains.
6. You have a choice: leave early or late. Leaving early reduces the chance of delay but makes everyone get up earlier (which reduces morale).
7. High morale is important for group harmony.
8. Everyone values attending the concert more than sleeping in.

The s(CASP) encoding of these statements is shown below:

```
1  delay :- rain, heavy_traffic.
2  miss_concert :- delay, late_departure.
3  everyone_disappointed :- miss_concert.
4  rain :- high_chance_of_rain.
5  high_chance_of_rain.
6  heavy_traffic :- rain.
7  #abducible early_departure.
8  late_departure :- not early_departure.
9  delay :- late_departure.
10 low_morale :- early_departure.
11 group_harmony :- high_morale.
12 high_morale :- not miss_concert.
13 -high_morale :- everyone_disappointed.
14 sleeping_more :- late_departure.
15 :- not group_harmony.
16 defeat_low_morale :- low_morale,
17       not defeat_low_morale.
18 defeat_low_morale :- high_morale.
```

Qualitative concepts such as late, delay, high chance of rain, etc., are used to perform reasoning. The code `#abducible early_departure` declares that `early_departure` proposition could be abduced, i.e., assumed to be either true or false (it is realized by creating the corresponding even loop over negation [Gupta *et al.*, 2022] rules). Alternatively, we can think of it as declaring that either we can assume `early_departure` to be true, or false, depending on which truth value will ensure the success of the query posed. The constraint in line 15 forces `group_harmony` to be true. Line 16-17 declare that `high_morale` preempts `low_morale` by making use of an odd loop over negation. Given this program, if we pose the query `?- high_morale` after loading this program on the s(CASP) system, we will obtain the following answer set (an answer set contains the set of predicates that must be true for the query to hold; it describes a possible world entailed by the rules):

{ group_harmony, high_morale, not miss_concert, delay, rain, high_chance_of_rain, heavy_traffic, not late_departure, early_departure, not -group_harmony, low_morale, defeat_low_morale }.

Note that a justification for a given query can also be obtained using the s(CASP) system. Several advanced applications that automate commonsense reasoning using ASP and s(CASP) have been developed. A few are described below. Some of them make use of qualitative reasoning.

Medical Treatment Auotmation: Most prominent is the CHeF system [Chen *et al.*, 2016] which emulates the expertise of a cardiologist to automatically generate treatment for congestive heart failure. Our studies show that the CHeF system performs on par with a cardiologist. This idea has been exploited by the Finnish company Forsante Oy to develop s(CASP)-based systems for medical treatment automation that have received regulatory approval in the Europe.

Automated Legal Reasoning The Rules-as-Code project in Canada has used s(CASP) to automate legal reasoning [Morris, 2023]. It has also been used as a back-end of the Logical English system [Kowalski *et al.*, 2023], also aimed at automation of legal reasoning.

Natural Language Understanding (NLU) & Chatbots: A combination of machine learning and commonsense reasoning can be used for NLU as well. The idea is to generate predicates from text using large language models such as GPT-3 [Brown *et al.*, 2020] via the use of *in-context learning* or *fine-tuning*. These predicates represent the meaning of the sentence, i.e., its *deep structure*. Commonsense reasoning can then be performed over these predicates to draw further conclusions, ask for missing information, and check for consistency of the information in the sentence. This method has been used to develop interactive reliable chatbots [Zeng *et al.*, 2024b; Zeng *et al.*, 2024a] for tasks such as an order-taker at a fast-food drive-through window. The technique has also been used by a company (CorroHealth) for near-100% accurate extraction of medical billing codes from textual description of electronic health records [Seeley, 2025].

Automatic Verification of Requirement Specs: Requirements written by humans for cyber-physical systems are represented in s(CASP) using the event calculus formalism. This modeling allowed us, for example, to discover timing errors in FDA-endorsed specification of the PCA pump [Vasícek *et al.*, 2024]. This effort shows that s(CASP) can also be used for precise quantitative modeling.

Building Information Modeling: Qualitative spatial reasoning can be performed using s(CASP) for modeling physical building design [Arias *et al.*, 2022b].

References

[Arias *et al.*, 2017] Joaquín Arias Arias, Kyle Marple, Elmer Salazar, Manuel Carro, and Gopal Gupta. s(CASP): A goal-driven predicate asp system, 2017. https://gitlab.software.imdea.org/ciao-lang/sCASP.

[Arias *et al.*, 2018] Joaquín Arias, Manuel Carro, Elmer Salazar, Kyle Marple, and Gopal Gupta. Constraint Answer Set Programming without Grounding. *TPLP*, 18(3-4):337–354, 2018.

[Arias *et al.*, 2022a] Joaquín Arias, Manuel Carro, Zhuo Chen, and Gopal Gupta. Modeling and reasoning in event calculus using goal-directed constraint answer set programming. *TPLP*, 22(1):51–80, 2022.

[Arias *et al.*, 2022b] Joaquín Arias, Seppo Törmä, Manuel Carro, and Gopal Gupta. Building information modeling using constraint logic programming. *Theory Pract. Log. Program.*, 22(5):723–738, 2022.

[Baral, 2003] C. Baral. *Knowledge representation, reasoning and declarative problem solving*. Cambridge University Press, 2003.

[Brewka *et al.*, 2011] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, and Others. Language models are few-shot learners. In *Proc. NeurIPS*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[Chen *et al.*, 2016] Zhuo Chen, Kyle Marple, Elmer Salazar, Gopal Gupta, and Lakshman Tamil. A physician advisory system for chronic heart failure management based on knowledge patterns. *Theory Pract. Log. Program.*, 16(5-6):604–618, 2016.

[Dietz Saldanha *et al.*, 2021] E.A. Dietz Saldanha, S. Hölldobler, and L.M. Pereira. Our themes on abduction in human reasoning: A synopsis. In *Abduction in Cognition and Action: Logical Reasoning, Scientific Inquiry, and Social Practice*, pages 279–293, 2021.

[Gelfond and Kahl, 2014] Michael Gelfond and Yuliya Kahl. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: An Answer Set Programming Approach*. Cambridge Univ. Press, 2014.

[Gunning *et al.*, 2010] David Gunning, Vinay K. Chaudhri, Peter Clark, Benjamin Grosof, and Others. Project halo update - progress toward digital aristotle. *AI Mag.*, 31(3):33–58, 2010.

[Gupta *et al.*, 2022] Gopal Gupta, Elmer Salazar, Sarat Chandra Varanasi, Kinjal Basu, Farhad Shakerin, Joaquín Arias Arias, Fang Li, and Huaduo Wang. Automated commonsense reasoning. In *Proc. GDE'22*, 2022. https://utdallas.edu/~gupta/csr-scasp.pdf.

[Gupta *et al.*, 2023] Gopal Gupta, Elmer Salazar, Farhad Shakerin, Joaquín Arias, Sarat Chandra Varanasi, Kinjal Basu, Huaduo Wang, Fang Li, Serdar Erbatur, Parth Padalkar, Abhiramon Rajasekharan, Yankai Zeng, and Manuel Carro. Prolog: Past, present, and future. In *Prolog: The Next 50 Years*, volume 13900 of *LNCS*, pages 48–61. Springer, 2023.

[Gupta *et al.*, 2024] Gopal Gupta, Elmer Salazar, and Joaquín Arias. Computational thinking with logic programming. In *ICLP'25 Workshop Proc: PEG 2.0*, volume 3799 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2024.

[Kowalski *et al.*, 2023] Robert Kowalski, Jacinto Davila, Galileo Sartor, and Miguel Calejo. Logical english for law and education. In *Prolog - The Next 50 Years*, number 13900 in LNCS. Springer, 2023.

[Kowalski, 1979] Robert A. Kowalski. *Logic for Problem Solving*. North Holland, 1979.

[Kowalski, 2011] Robert A. Kowalski. *Computational Logic and Human Thinking*. Cambridge University Press, 2011.

[Marple *et al.*, 2017] Kyle Marple, Elmer Salazar, and Gopal Gupta. Computing stable models of normal logic programs without grounding, 2017. arXiv:1709.00501.

[Morris, 2023] Jason Morris. Blawx: User-friendly goal-directed answer set programming for rules as code. In *Proc. Prog. Lang. and the Law (ProLaLa)*, 2023.

[Seeley, 2025] David Seeley. Ai 'reasoning' for medical coding automation. *Dallas Innovates*, 2025.

[Vasícek *et al.*, 2024] Ondrej Vasícek, Joaquín Arias, Jan Fiedor, Gopal Gupta, Brendal Hall, Bohuslav Krena, Brian Larson, Sarat Chandra Varanasi, and Tomás Vojnar. Early validation of high-level system requirements with event calculus and answer set programming. *Theory Pract. Log. Program.*, 24(4):844–862, 2024.

[Wang and Gupta, 2024] Huaduo Wang and Gopal Gupta. FOLD-SE: an efficient rule-based machine learning algorithm with scalable explainability. In *Proc. PADL'24*, volume 14512 of *LNCS*, pages 37–53. Springer, 2024.

[Zeng *et al.*, 2024a] Yankai Zeng, Abhiramon Rajasekharan, Kinjal Basu, Huaduo Wang, Joaquín Arias, and Gopal Gupta. A reliable common-sense reasoning socialbot built using llms and goal-directed ASP. *Theory Pract. Log. Program.*, 24(4):606–627, 2024.

[Zeng *et al.*, 2024b] Yankai Zeng, Abhiramon Rajasekharan, Parth Padalkar, Kinjal Basu, Joaquín Arias, and Gopal Gupta. Automated interactive domain-specific conversational agents that understand human dialogs. In *International Symposium on Practical Aspects of Declarative Languages*, pages 204–222. Springer, 2024.