Automating Argumentation with Goal-directed Answer Set Programming

Gopal Gupta, Sarat Varanasi, Kinjal Basu, Zhuo Chen
Elmer Salazar, Farhad Shakerin, Serdar Erbatur, Fang Li, Huaduo Wang
The University of Texas at Dallas, Richardson, USA

Joaquín Arias
Universidad Rey Juan Carlos, Madrid, Spain

Brendan Hall, Kevin Driscoll
Honeywell Corp, Minneapolis, MN

Overview

- **What?** : formalize human thought process (automate commonsense reasoning)
- **Why?** : If we can formalize the human thought process, then ***everything*** can be automated (medical treatment, self-driving cars, automated s/w certification)
- Humans are sophisticated & effective thinkers: operate with just 20 watts of power:
  - and they can perform deductive, abductive and inductive reasoning
- Classical logics are limited: can perform reasoning for only well-founded or inductively constructed objects
- Humans employ both inductive (well-founded) and co-inductive (circular or assumption-based) reasoning; they also draw conclusions from failure of proofs (NAF)
- Negation-as-failure (NAF): a key concept for emulating the human thought process
- ASP: a knowledge representation and reasoning paradigm based on negation-as-failure
- s(CASP): Goal-directed implementation of ASP, crucial for automating argumentation

Formalizing Human Thought

- Formalizing the human thought process has been considered hard
  - History: Syllogisms, Boolean logic, Predicate Logic, ……. Other advanced logics
  - These logics have not been able to model the sophistication/effectiveness of human reasoning …
- Early problems of naïve set theory and predicate logic (Russell's paradox) led mathematicians and logicians to focus only on inductive sets & reasoning
  - ….. which is insufficient to model commonsense reasoning
- Classical systems of logic cannot reason about themselves (Tarski)
  - A logic cannot have its own truth predicate: need meta logic, meta meta logic, *ad infinitum*
  - Classical logic cannot reason about its proof failure, for instance
  - Kripke 1975 showed that a language can have its own truth predicate *and* consistency
  - Led to idea of co-induction and modeling of circular reasoning (that humans do employ)

Formalizing Human Thought

- "Informal" logic (IL) proposed due to lack of success of classical logic
  - Attempt to make the human thought process "precise"
  - Consists of inference rules (called arguments) and reasoning (proof).
  - An argument is an attempt to present evidence for a conclusion (or a claim)
  - Relies on premises that support the conclusion
- Excellent primer by Holloway & Wasson on IL for proving overarching properties
  - Following the primer, we show that there is 1-1 mapping between IL and ASP
- Why automate informal logic? Some arguments against automating IL:
  - Informal logic is meant for humans; argumentation should be done by humans
  - "automation more often deters rather than promotes thinking"

Formalizing Human Thought

- Reason for automating informal logic: Human abilities are limited:
  - Humans can handle only 4 variables at a time (Halford et al, *Psych. Sci.*, 2005)

"If the number of variables to be considered exceeds human processing capacity then the worker will drop his or her mental bundle and become unable to proceed."

"worker may revert to a simplified version of the task that does not take all aspects into account and therefore may make the wrong decision."

- Individual statements easy to comprehend; jointly they become hard to understand

Switch P turns on **if** switch S is not on.
Switch S turns on **if** switch R is not on.
Switch R turns on **if** switch P is not on.
Switch R turns on **if** switch S is not on.

- Who will go to Mexico?
- Formalization/Automation is hence important;
- We will do it via *answer set programming*, an extension of logic programming/Prolog

Logic Programming (Prolog)

Facts:

father(jim, john). mother(mary, john).

father(john, rob). mother(sue, rob).

…. ….

Rules:

parent(X, Y) if mother(X, Y).

parent(X, Y) if father(X, Y).

grandparent(X, Y) if parent(X, Z) **&** parent(Z, Y).

anc(X, Y) if parent(X, Y).

anc(X, Y) if parent(X, Z) **&** anc(Z, Y).

Queries:

?- anc(jim, rob). ?- anc(A, rob).

Logic Programming (Prolog)

Facts:

father(jim, john). mother(mary, john).

father(john, rob). mother(sue, rob).

…. ….

Rules:

parent(X, Y) ⇐ mother(X, Y).

parent(X, Y) ⇐ father(X, Y).

grandparent(X, Y) ⇐ parent(X, Z) ∧ parent(Z, Y).

anc(X, Y) ⇐ parent(X, Y).

anc(X, Y) ⇐ parent(X, Z) ∧ anc(Z, Y).

Queries:

?- anc(jim, rob). ?- anc(A, rob).

Logic Programming (Prolog)

Facts:

```
father(jim, john).    mother(mary, john).
father(john, rob).    mother(sue, rob).
....                  ....
```

Rules:

```
parent(X, Y) :- mother(X, Y).
parent(X, Y) :- father(X, Y).
grandparent(X, Y) :- parent(X, Z), parent(Z, Y).
anc(X, Y) :- parent(X, Y).
anc(X, Y) :- parent(X, Z), anc(Z, Y).
```

Queries:

```
?- anc(jim, rob).        ?- anc(A, rob).
```

Common Sense Reasoning

- Standard Logic Programming fails at performing human-style commonsense reasoning
- In fact, most formalisms have failed; problem: monotonicity
- Commonsense reasoning requires:
  1. Non-monotonicity: the system can revise its earlier conclusion in light of new information (contradictory information discovered later does not break down things as in classical logic)
     - If Tweety is a bird, it can fly; conclusion to be retracted if Tweety is found to be a penguin
  2. Draw conclusions from absence of information; humans frequently use this [default reasoning] pattern:
     - Can't tell if it is cold outside. If I see no one wearing a jacket, so it must not be cold
     - You text your friend in the morning; he does not respond; normally he responds right away. You may conclude: he must be taking a shower.
  3. Possible worlds semantics: be able to work with multiple worlds (non-inductive semantics)
- Negation as failure (NAF) allows modeling of #1 and #2
- Automating commonsense reasoning achievable with NAF *& multiple worlds semantics*

Classical Negation vs Negation as Failure

- Classical negation (CN)
  - represented as -p
  - e.g, -sibling(john, jim) %states that John and Jim are definitely not siblings
  - An explicit proof of falsehood of predicate p is needed
  - -murdered(oj, nicole) holds true only if there is an explicit proof of OJ's innocence (seen in Boston airport, Nicole's body was found in LA)
- Negation as failure (NAF)
  - represented as not(p)
  - e.g., not sibling(john, jim) %states that *no evidence* John and Jim are siblings
  - We try to prove a proposition p, if we fail, we conclude not(p) is true
  - No evidence of p then conclude not(p)
  - not(murdered(oj, nicole)) holds true if we fail to find any proof that OJ killed Nicole

FAIL TO PROVE (NAF) vs **EXPLICITLY PROVING FALSEHOOD (CN)**

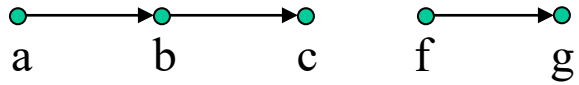
Failure of Classical Reasoning Methods for CSR

- Example: Transitive closure: edge(a, b). edge(b, c). edge(f, g).

  reach(X, Y) :- edge(X, Y).

  reach(X, Y) :- edge(X, Z), reach(Z, Y).

  ?- reach(a, f).



  Query fails (no proof); Under classical theorem proving we can't conclude that f is unreachable from node a.

- Need axioms for **un**reachability, only then we can conclude –reach(a, f).
  - That is, we have to explicitly define rules for –reach(X, Y).
- Failure is not of logic or the decision procedure, rather how we interpret the rules.
- Interpret implications as causal relations: A if B means A iff B

  reach(X, Y) $\leftrightarrow$ (edge(X, Y) $\vee$ (edge(X, Z), reach(Z, Y))).
- Now with NAF we can write: unreachable(X, Y) :- not reachable(X, Y).
- Note: when humans write "if A then B", they mean "A iff B" most of the time;
  - E.g.: breaks_object :- drop_object we automatically mean not_breaks_object :- not_drop_object.

Multiple Possible Worlds

- Classical logic only allows inductive semantics (based on least fixpoint)
- Non-inductive semantics (cyclical reasoning) not allowed; cyclical reasoning arises often

  eats_dinner(joe) :- eats_dinner(jill).

  eats_dinner(jill) :- eats_dinner(joe).
- Inductive semantics: both will not eat
- In reality two models: one in which both eat dinner, one in which none eats dinner
- In propositional realm, both models can be computed:

  $\text{eats\_dinner(joe)} \leftrightarrow \text{eats\_dinner(jill)}$
- Problem arises in the predicate realm

  $\forall X\ \forall Y$ eats_dinner(X) :- loves(X,Y), eats_dinner(Y).

  $\forall X\ \forall Y$ eats_dinner(Y) :- loves(Y,X), eats_dinner(X).
- Need to operationalize *coinduction* to execute such programs (supported by ASP)

Answer Set Programming (ASP)

- Prolog extended with NAF; Rules are of the form:

  p. (fact)

  p :- a_1,…, a_m, not b_1 ,…, not b_n. m, n ≥0 (rule)
- ASP is a popular formalism for non monotonic reasoning
- Applications of ASP to KR&R, planning, constrained optimization, etc.
- Semantics: lfp of a residual program obtained after "Gelfond-Lifschitz" transform
- Popular implementations: DLV, CLASP, etc. More than 30 years of research invested
- Example programs:

flies(X) :- bird(X), not abnormal_bird(X).

abnormal_bird(X) :- penguin(X).

bird(tweety). bird(sam).

Query: ?- flies(tweety) & ?- flies(sam) will succeed

Now add the fact: penguin(tweety).

Query: ?- flies(tweety) fails while ?- flies(sam) succeeds

(Default reasoning with exceptions)

teach_db(john) :- not teach_db(mary).

teach_db(mary) :- not teach_db(john).

(multiple possible worlds)

false :- teach_db(mary).

(constraints)

s(CASP): Goal-directed Execution of ASP

- Current ASP systems are highly sophisticated, but can only execute propositional programs as they rely on a SAT solver
- We know how to handle cyclical propositional programs, but not cyclical programs with predicates
- Discovery of coinductive logic programming (coLP) changed that: coLP gives operational semantics to cyclical programs (greatest fixpoint based computations)
- Culminated in the s(CASP) system developed at UT Dallas & IMDEA, Spain:
  - Supports predicates: no grounding is needed
  - Supports goal-directed or query-driven execution, just like Prolog
  - Supports coinduction & constructive negation
  - Supports constraints over dense domains such as real numbers
  - Supports both deductive & abductive reasoning; rules represent inductive inference
  - Freely available on github: https://gitlab.software.imdea.org/ciao-lang/sCASP

Automating Common Sense Reasoning

- If we automate CSR, we can automate the human thought process
- **Model any intelligent task that humans can accomplish**
- Achieve artificial general intelligence (AGI), at least
- Our goal should be to simulate an **unerring** human
- So far:
  - We need negation-as-failure to model the fact that humans operate in a world where lot of stuff is unknown
  - We need possible world semantics, as humans do not always reason inductively
- Answer Set Programming provides both these features, hence is an excellent candidate for automating common sense reasoning using the s(CASP) system

Deductive, Inductive, and Abductive Reasoning

- **Deductive Reasoning:** All birds fly. Tweety is a bird. Deductively infer: Tweety flies.

  flies(X) :- bird(X). bird(tweety). ?- flies(tweety). is true.

- **Abductive Reasoning:** inference to the best explanation; hypothesis generation

  Observed: flies(tweety). We know: flies(X) :- bird(X).

  surmise: bird(tweety)

- **Inductive Reasoning:**
  - We see a number of birds and they all fly; we induce that all birds fly.

    flies(X) :- bird(X),
  - However, never sure that all birds fly, so we leave room for exceptions

    flies(X) :- bird(X), not abnormal_bird(X).

    abnormal_bird(X) :- penguin(X).

Deductive, Inductive, and Abductive Reasoning

- Three pieces of information:

  (i) flies(X) :- bird(X). (ii) bird(tweety). (iii) flies(tweety).

  – Deduction: Given (i) and (ii), infer (iii) [sound]
  – Abduction: Given (i) and (iii), infer (ii) [inference to best explanation]
  – Induction: Given (ii) and (iii), infer (i) [learning a rule from data]

- Humans make these 3 types of inferences; all 3 also supported by ASP
  – Deduction: body of a rule holds, then infer the head; directly supported by ASP
  – Abduction: assumption-based reasoning; for each abducible predicate, p,
    - add rule: p :- p. i.e., p is true in one world and false in another.
    - Query will succeed in the appropriate world
  – Induction: default rules capture inductive inference (used for explainable AI)
    - They go one step further: distinguish between exceptions and noise

Holloway & Wasson's IL Primer

- Concepts in Holloway's & Wasson's primer can be directly mapped to ASP
  - Conclusion ≡ Claim ≡ Theorem ≡ Query
  - Argument ≡ Rule ≡ Clause
  - Premise (what an audience believes) ≡ Evidence or Assumption
    - Evidence ≡ Fact Assumption ≡ Abducible
  - Reasoning ≡ Proof
  - Binding ≡ Substitution
  - Defeater (support for not believing) ≡ Negated Goal (NAF)
- Example
  - Claim: ?- flies(tweety).
  - Argument: flies(X) :- bird(X).
  - Evidence: bird(tweety).
  - Defeater: penguin(tweety).
  - Refined argument: flies(X) :- bird(X), not penguin(X).

Precepts

- Primer presents a number of guiding principles called precepts:
  - **Locality:** cogency of a compound argument never exceeds the cogency of its weakest atomic argument: In ASP, head of a rule inferred only if entire body is true
  - **Depth:** argument decomposition must descend far enough to serve stake-holder objectives: easily modeled in ASP as user chooses level of abstraction
  - **Change:** context in which an argument is made may change: pertains to non-monotonicity of our knowledge that ASP supports
  - **Induction**: not all reasoning may be deductive: ASP allows inductive knowledge to be expressed (flies/1 rule above is based on inductive inference)
  - **Plausibility**: people have biases, beliefs, etc., that color their perception of world: belief, biases, etc., can be modeled with ASP; e.g., consider the rule: flies(X) :- bird(X), not ab(X).

    ab(X) can be interpreted in two ways: we can be aggressive in our reasoning or conservative

    1. ab(X) :- penguin(X). Aggressive: X is abnormal if X is a penguin; no knowledge of X means X flies
    2. ab(X) :- not –penguin(X). X is abnormal if no proof its not a penguin; no knowledge of X, then X does not fly

Possible Worlds (Example from Peter Norvig)

People can talk. (in all worlds)
talk(X) :- people(X).

Non-human animals can't talk. (in real world)
-talk(X) :- non_human_animal(X), rw.

Human-like cartoon characters can talk. (in cartoon world)
talk(X) :- human_like_cc(X), cw.

Fish can swim. (in all worlds)
swim(X) :- fish(X).

A fish is a non-human animal. (in real world)
non_human_animal(X) :- fish(X), rw.

Nemo is a human_like cartoon character. (in cartoon world)
human_like_cc(nemo) :- cw.

Nemo is a fish. (in all worlds)
fish(nemo).

cw :- not rw.
rw :- not cw.

Can nemo talk?
?- talk(nemo).
succeeds only in cartoon world (cw)

Can nemo swim?
?- swim(nemo).
succeeds in both worlds, cw and real world (rw)

Representing Knowledge with ASP

- Suppose Mary believes that John was in Holland Park some morning and that Holland Park is in London. Then Mary can deductively reason from these beliefs, to conclude that John was in London that morning.
- So the reasoning cannot be attacked. However, perfection remains unattainable since the argument is still fallible: its grounds may turn out to be wrong. For instance, Jan may tell us that he met John in Amsterdam that morning around the same time. We now have a reason against Mary's belief that John was in Holland Park that morning, since witnesses usually speak the truth. Can we retain our belief or must we give it up? The answer to this question determines whether we can accept that John was in London that morning.
- Maybe Mary originally believed that John was in Holland Park for a reason. Maybe Mary went jogging in Holland Park and she saw John. We then have a reason supporting Mary's belief that John was in Holland Park that morning, since we know that a person's senses are usually accurate. But we cannot be sure, since Jan told us that he met John in Amsterdam that morning around the same time. Perhaps Mary's senses betrayed her that morning?
- But then we hear that Jan has a reason to lie, since John is a suspect in a robbery in Holland Park that morning and Jan and John are friends. We then conclude that the basis for questioning Mary's belief that John was in Holland Park that morning (namely, that witnesses usually speak the truth and Jan witnessed John in Amsterdam) does not apply to witnesses who have a reason to lie. So our reason in support of Mary's belief is undefeated and we accept it.

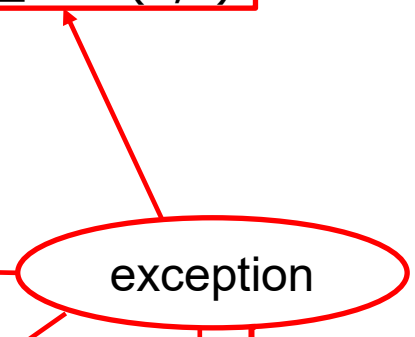
Representing Knowledge with ASP

- We map various statements into ASP, e.g., the argument (rule) for speaking truth:

speaks_truth(X, Y) :- person(X), person(Y), observer(E,X), theme(E,Y), not ab_speaks_truth(X,Y).
ab_speaks_truth(X, Y) :- may_lie(X,Y).

may_lie(X,Y) :- person(X), person(Y), not -lie(X,Y).
-lie(X,Y) :- person(X), person(Y), not conflict_interest(X,Y).

conflict_interest(X,Y) :- friends(X,Y), crime_suspect(Y), not ab_conflict_interest(X).
crime_suspect(X) :- person(X), robbery_suspect(X), not ab_crime_suspect(X).



- Likewise for Mary's infirmity

accurate_senses(X) :- person(X), not ab_accurate_senses(X).
ab_accurate_senses(X,sense) :- person(X), age(X, A), A = old.

Representing Knowledge with ASP

- Based on the assumptions we make, we can prove different claims:
  - Mary is infirm and has poor eyesight #abducible age(mary, old).
  - Jan and John are friends #abducible friends(john, jan).
  - John is a robbery suspect #abducible robbery_suspect(john).
- Assumptions are modeled as abducibles (abductive reasoning)
  - Given p $\Rightarrow$ q and an observation q, we *abduce* (*assume*) that p is true
  - #abducible p $\equiv$ p $\leftrightarrow$ p
- Now claims can be established automatically with requisite assumptions
- ?- claim(john_location_unknown).

  Assumptions: age(mary,old) $\wedge$ friends(jan,john) $\wedge$ robbery_suspect(john) %can't trust Mary or Jan

- ?- claim(john_not_in_london).

  Assumptions: age(mary,old) $\wedge$ not friends(jan, john). % trust Jan

  Assumptions: age(mary,old) $\wedge$ friends(jan,john) $\wedge$ not robbery_suspect(john) % trust Jan

- ?- claim(john_in_london).

  Assumptions: friends(jan,john) $\wedge$ robbery_suspect(john) %trust Mary

  Assumptions: age(mary, B | B $\neq$ old) $\wedge$ friends(jan,john) $\wedge$ robbery_suspect(john) %trust Mary

Justification tree in English can be produced

- s(CASP) can produce justification tree (proof tree) for a given claim:

John is in London at time 8AM, because
Mary saw John at 7:45AM, and
John was at holland_park at 7:45AM, and
holland_park is in london, and
Mary has sharp senses.
Sighting happened at time 7:45AM, because
John was at holland_park at 7:45AM.
Jan saw John at 7:45AM, and ...
John was at Amsterdam at 7:45AM, and
John is no longer in London at 7:45AM, because
Jan saw John at 7:45AM.
Meeting event involves John, because
Jan saw John at 7:45AM.
Meeting event involves John, justified above, and
Jan may lie about John, because
Jan has conflict of interest with John, because
we assume that Jan and John are friends, and
John is a crime suspect.

Finding Defeaters: Is the House Safe?

- Use ASP/s(CASP) to automatically finding defeaters for arguments (via abduction).
- Given a set of rules, we declare abducibles (assumptions)
- Find assumptions under which a proof will succeed, given that it failed (or vice versa)

```
turn_sprinkler_on :- fire.
sprinkler_on :- turn_sprinkler_on, not water_off.
turn_water_off :- water_on_floor, not people_present.
water_off :- turn_water_off.
```

Encode knowledge

```
house_burns_down :- fire, not sprinkler_on.
```

Encode requirements/violations

```
#abducible fire.    #abducible water_on_floor.    #abducible people_present.
```

Specify sensors as abducible

- Our defeater analysis will yield that one of the rules should be revised

```
turn_water_off :- water_on_floor, not people_present, not fire.
```

Conclusions

- Answer Set Programming: A very powerful paradigm for knowledge rep. & reasoning
- Can simulate complex reasoning, incl. common sense reasoning employed by humans
- Automating commonsense reasoning: ultimate goal of automation (model unerring human)
- Informal Logic *a la* Holloway and Wason's Primer can be directly modeled with ASP
- Goal-directed implementations such as s(CASP) are crucial to ASP's success
- Many interesting applications developed/being-developed
  - -- Natural Language Question Answering based on truly "understanding" the text
  - -- Automated Treatment of Heart Failure based on guidelines published by American College of Cardiology
  - -- Chatbots that "understand" user utterances
  - -- Automated Defeater Discovery (claim verification, done using Adelard's ASCE s/w assurance tool)
- Automating common sense reasoning: holy grail of artificial general intelligence (AGI)

Ack: National Science Foundation & DARPA ARCOS for research support

THANK YOU

QUESTIONS?

More information:
http://utdallas.edu/~gupta