On Common Sense

Tanel Tammet
Tallinn University of Technology

with contributions from Dirk Draheim, Priit Järv, Martin Verrev
Overview of the talk

Introduction and context
End-to-end machine learning: examples and counterexamples
Hybrid systems: machine learning + rules
  ML extended with rules
  Rules extended with ML: our project as a case study
In artificial intelligence (AI), **commonsense reasoning** is a human-like ability to make presumptions about the type and essence of ordinary situations humans encounter every day.

These assumptions include judgments about the nature of physical objects, taxonomic properties, and peoples' intentions.
Artificial general intelligence (AGI) is the ability of an intelligent agent to understand or learn any intellectual task that a human being can. It is a primary goal of some artificial intelligence research and a common topic in science fiction and futures studies.

AGI can also be referred to as strong AI, full AI, or general intelligent action, although some academic sources reserve the term "strong AI" for computer programs that experience sentience or consciousness.
Famous A.I. winters

Beyond actual numbers, A.I. has followed a pattern of hype and disillusion for years. What does the future hold?
The narrow focus of the talk

Capturing meaning in natural language sentences: understanding what is said, answering questions, explaining answers, calculating plans of action.
IBM Watson winning Jeopardy in 2011
Example questions from Jeopardy with Watson

• It's just a bloody nose! You don't have this hereditary disorder once endemic to European royalty: Haemophilia

• You just need a nap! You don't have this sleep disorder that can make sufferers nod off while standing up: Narcolepsy

• Heitor Villa-Lobos dedicated his "12 Etudes" for this instrument to Andres Segovia: Guitar

• Paganini's "24 Capricci" set the standard for etudes for this instrument: Violin

• Rembrandt's Biblical Scene "Storm on the Sea of" this was stolen from a Boston museum in 1990: Galilee
Classic old-style approach

Three components needed:

• English to logic parser
• Logical reasoner
• Knowledge bases used by the reasoner

They all failed miserably!
Famous case study: CYC

Wikipedia

• **Cyc** is a long-term artificial intelligence project that aims to assemble a comprehensive ontology and knowledge base that spans the basic concepts and rules about how the world works.

• Douglas Lenat began the project in July 1984 ... since January 1995, has been under active development by the **Cycorp** company.

• The Cyc project has been described as "one of the most controversial endeavors of the artificial intelligence history". Catherine Havasi, CEO of Luminoso, says that Cyc is the predecessor project to IBM's Watson. Machine-learning scientist Pedro Domingos refers to the project as a "catastrophic failure".
A mostly failed limited approach: Semantic Web

• Idea: make it easy to publish, harvest, integrate and use structured knowledge on the web.
• Proposals worked out: RDF data model and OWL rule-like language

• RDF main idea: represent all data in a single three-column table of <object id> <property> <value> triplets like this:

    client_1  name  „John Brown“
    client_1  balance  200
What happened with the Semantic Web?

Negative:
• Mostly academic research leading to over-complicating even simple things.
• People were not really keen to publish data or rules on the web.

Positive:
• Attempt to make it a bit simpler in a closely related area: linked data
• JSON-LD language giving RDF interpretation to JSON
• http://schema.org repository of suggested type/property names
End-to-end deep learning for Q&A

Main systems used by people for examples & research:

• Derivates of google **BERT**: RoBERTa, DistilBERT, DeBERTa, ...
• OpenAI **GPT-3**
End-to-end learning: pro and con

Surprisingly effective (ca ... 75 ... % correct) on a wealth of examples.

*Superglue:*

Barq’s – Barq’s is an American soft drink. Its brand of root beer is notable for having caffeine. Barq’s, created by Edward Barq and bottled since the turn of the 20th century, is owned by the Barq family but bottled by the Coca-Cola Company. It was known as Barq’s Famous Olde Tyme Root Beer until 2012.

Question: is barq’s root beer a pepsi product?

Answer: No
More benchmark examples

*Superglue:*
My body cast a shadow over the grass. What’s the CAUSE for this?
1. The sun was rising.
2. The grass was cut.

*OpenBookQA:*
Which of these would let the most heat travel through?
1. a new pair of jeans.
2. a steel spoon in a cafeteria.
3. a cotton candy at a store.
4. a calvin klein cotton hat.
Suspicions

• Maybe the results are not good enough?
• Maybe the systems learn superficial hints / artifacts?
• Maybe the results are not generalizable?
• What happens if they attack really new benchmarks?
• ...

<table>
<thead>
<tr>
<th>Category</th>
<th>Tasks</th>
<th>Zero-shot</th>
<th>One-shot</th>
<th>Few-shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completion prediction</td>
<td>Lambada</td>
<td>0.766*</td>
<td>0.731*</td>
<td>0.872*</td>
</tr>
<tr>
<td>Reading comprehension</td>
<td>BoolQ</td>
<td>0.782</td>
<td>0.825</td>
<td>0.848</td>
</tr>
<tr>
<td>Reading comprehension</td>
<td>RACE-h</td>
<td>0.479</td>
<td>0.484</td>
<td>0.479</td>
</tr>
<tr>
<td>Commonsense reasoning</td>
<td>PiQA</td>
<td>0.820*</td>
<td>0.810*</td>
<td>0.832*</td>
</tr>
<tr>
<td>Commonsense reasoning</td>
<td>HellaSwag</td>
<td>0.802</td>
<td>0.802</td>
<td>0.824</td>
</tr>
<tr>
<td>Commonsense reasoning</td>
<td>WinoGrande</td>
<td>0.730</td>
<td>0.737</td>
<td>0.789</td>
</tr>
<tr>
<td>Natural language inference</td>
<td>ANLI-R2</td>
<td>0.366</td>
<td>0.397</td>
<td>0.396</td>
</tr>
<tr>
<td>Natural language inference</td>
<td>HANS</td>
<td>0.607</td>
<td>0.649</td>
<td>0.702</td>
</tr>
<tr>
<td>Word sense disambiguation</td>
<td>WiC</td>
<td>0.486</td>
<td>0.513</td>
<td>0.585</td>
</tr>
</tbody>
</table>
What Will it Take to Fix Benchmarking in Natural Language Understanding?

Samuel R. Bowman, New York University
George E. Dahl, Google Research, Brain Team

... Ample evidence has emerged that the systems that have topped these leaderboards can fail dramatically on simple test cases that are meant to test the very skills that the leaderboards focus on (McCoy et al., 2019; Ribeiro et al., 2020).

This result makes it clear that our systems have significant room to improve. However, we have no guarantee that our benchmarks will detect these needed improvements when they're made.

... The recently collected ANLI (Nie et al., 2020a) dataset illustrated this by adversarially collecting difficult examples where current state-of-the-art models fail. Recently, Brown et al. (2020) found that GPT-3 performs not much above chance on ANLI, noting that "NLI is still a very difficult task for language models and [it is] only just beginning to show signs of progress" (Brown et al., 2020, p.20).
The president advised the doctor. Did the doctor advise the president?

GPT-3: The doctor advised the president.

The artists that supported the senators shouted. Did the senators shout?

GPT-3: The senators did not shout.

The athlete who the banker called advised the scientist. Did the scientist advise the banker?

GPT-3: The scientist did not advise the banker.

Before the presidents ran, the tourist shouted. Did the presidents run?

GPT-3: No, the presidents didn't run.
Exploring GPT-3: some famous benchmarks

Is it easier for human's to survive in an ocean, cave, town or alone?

It is easier for human's to survive in a town.

Metal is a thermal conductor. Which of these would let the most heat travel through? A new pair of jeans, a steel spoon in a cafeteria, a cotton candy at a store or a calvin klein cotton hat?

A steel spoon in a cafeteria would let the most heat travel through.

The large ball crashed right through the table because it was made of steel. What was made of steel?

The large ball.

The large ball crashed right through the table because it was made of styrofoam. What was made of styrofoam?

The table was made of styrofoam.
Exploring GPT-3

Can beer drink?
Yes, beer is a beverage.

Can a bottle drink?
No, a bottle cannot drink.

Can water drink?
No, water cannot drink.

Can lemonade drink?
Yes, lemonade can be drunk.
Exploring GPT-3

John is a man or not a man?
I cannot answer that question.
John has a car and does not have a car?
John has a car.
John is nicer than Mike. Mike is nicer than Eve. Who is nicer than John?
Eve is nicer than John.
Exploring GPT-3

Elephants are animals. John is an elephant. John is an animal?
Yes, John is an animal.

Foos are animals. John is a foo. John is an animal?
No, John is not an animal.

All foorers are animals. John is a foorer. John is an animal?
Yes
Exploring GPT-3

Birds fly. Baby birds can not fly. John is a baby bird. Mike is a bird. Can Mike fly?
Yes, Mike can fly.

Birds fly. Baby birds can not fly. John is a baby bird. Mike is a bird. Can John fly?
No, John cannot fly.
Exploring GPT-3

Penguins are birds who do not fly. Birds fly. John is a penguin. John flies?
False. Penguins are birds who do not fly.

John is a bird. Folkameers are birds who can not fly. Birds can fly. John is not a Folkameer. John can fly?
No, John is not a Folkameer.
Exploring GPT-3

Penguins cannot fly. Most birds can fly. Most things cannot fly. Birds are things. **John is a bird.** Can John fly?

No, John cannot fly.

Penguins cannot fly. Most birds can fly. Most things cannot fly. Birds are things. **John is a penguin.** Can John fly?

No, John cannot fly.

Penguins cannot fly. Most birds can fly. Most things cannot fly. Birds are things. **John is a thing.** Can John fly?

No, John cannot fly.

Penguins cannot fly. Most birds can fly. Most things cannot fly. Birds are things. **John is a bird.** Why cannot John fly?

John cannot fly because he is not a bird.
Two birds sat on a tree. Three more birds came to sit on the tree. How many birds are sitting on the tree now?

There are five birds sitting on the tree now.

Ten birds sat on a tree. Seven more birds came to sit on the tree. How many birds are sitting on the tree now?

There are now ten birds sitting on the tree.
Exploring GPT-3: Allen AI examples

Metals conduct electricity. Insulators do not conduct electricity. If something is made of iron then it is metal. Nails are made of iron. Nails conduct electricity?

Nails are made of iron, but they are not good conductors of electricity.

Metals conduct electricity. Insulators do not conduct electricity. If something is made of iron then it is metal. Nails are made of plastic. Plastic is an insulator. Nails conduct electricity?

Nails do not conduct electricity.
Exploring GPT-3: Allen AI examples

The circuit has a switch.
The circuit has a bell.
The switch is off.
If the circuit has the switch and the switch is on then the circuit is complete.
If the circuit does not have the switch then the circuit is complete.
If the circuit is complete and the circuit has the light bulb then the light bulb is glowing.
If the circuit is complete and the circuit has the bell then the bell is ringing.
If the circuit is complete and the circuit has the radio then the radio is playing.
The bell is ringing?
GPT-3: Yes
Exploring GPT-3

A robot hand can lift one block at a time.

Birds can fly.
Penguins are birds which cannot fly.
John is a bird.
The robot hand cannot hold more than one block.
The robot hand can put a block on another block or a table.
A block can be on another block.
No more than one block can be on another block.
If a block X is on another block Y, the robot hand cannot lift the block Y.
There are three blocks on the table: a red block, a blue block, a yellow block.

Q: Can John fly?
GPT-3: No
Q: Why cannot John fly?
GPT-3: John cannot fly because he is a penguin.
Can we fix end-to-end learning?

Nobody knows so far.

Some arguments against:

....
Explainability?

Not much success explaining the reasoning behind the answers.
Real neurons vs neural nets

- The number of cells in a brain is similar to the number of transistors in a computer.
- Neurons are cells. Cells are extremely complex small animals.
- https://www.youtube.com/watch?v=hb7tjqhfDus
- One neuron has ca 7000 of synapses. Each synapse does learn. There are many different ways synapses communicate: electrical impulses, chemistry, rna exchange, ...
Brain has a genetic pre-determined structure

• Just observe a newborn calf.

https://www.youtube.com/watch?v=kUPX6p_btJ0

• Wikipedia: universal grammar (UG), in modern linguistics, is the theory of the genetic component of the language faculty, usually credited to Noam Chomsky. The basic postulate of UG is that there are innate constraints on what the grammar of a possible human language could be. When linguistic stimuli are received in the course of language acquisition, children then adopt specific syntactic rules that conform to UG.
Most landmark A.I. systems are hybrid

... except for object recognition and story/picture generation

• Siri and Google Assistant
• Most automated driving systems
• AlphaGo and AlphaZero
• AlphaFold
Can we pre-build some necessary structure?

Like making the system internally know things like:

• There are objects, and they are mostly persistent.
• Objects are in space and time and follow some basic physics.
• Objects have properties.
• Objects have relations between each other.
• Relations have some basic logical properties.
• Some things are dangerous, some are desirable

...
Hybrid systems: ML + rules

Two approaches

• Inject rules into a machine learning model (extremely active field)

• Add machine learning to a rule-based system (only a few groups)
Inject rules into machine learning

- Immense number of possible ways to do that
- People are actively experimenting
- Typically we get small gains over pure end-to-end transformers
For example, try to calculate saliency and then count

Conversational Multi-Hop Reasoning with Neural Commonsense Knowledge and Symbolic Logic Rules

Forough Arabshahi
Facebook

Jennifer Lee
Facebook

Antoine Bosselut
EPFL

Yejin Choi
University of Washington

Tom Mitchell
Carnegie Mellon University

2021
If it is not used for hair, a round brush is an example of what? (art supplies)
Compared results with the same system

### Table 3: Test Accuracy comparison on OpenBookQA
Experiments are controlled using the same seed LM for all LM+KG methods.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AristoRoBERTa (no KG)</td>
<td>78.4</td>
</tr>
<tr>
<td>+ RGCN</td>
<td>74.6</td>
</tr>
<tr>
<td>+ GconAttn</td>
<td>71.8</td>
</tr>
<tr>
<td>+ RN</td>
<td>75.4</td>
</tr>
<tr>
<td>+ MHGRN</td>
<td>80.6</td>
</tr>
<tr>
<td>+ QA-GNN</td>
<td>82.8</td>
</tr>
<tr>
<td><strong>GREASELM (Ours)</strong></td>
<td><strong>84.8</strong></td>
</tr>
</tbody>
</table>

### Table 4: Test accuracy comparison to public OpenBookQA model implementations.
*UnifiedQA (11B params) and T5 (3B) are 30x and 8x larger than our model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc.</th>
<th># Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALBERT (Lan et al., 2020) + KB</td>
<td>81.0</td>
<td>~235M</td>
</tr>
<tr>
<td>HGN (Yan et al., 2020)</td>
<td>81.4</td>
<td>≥355M</td>
</tr>
<tr>
<td>AMR-SG (Xu et al., 2021)</td>
<td>81.6</td>
<td>~361M</td>
</tr>
<tr>
<td>ALBERT + KPG (Wang et al., 2020)</td>
<td>81.8</td>
<td>≥235M</td>
</tr>
<tr>
<td>QA-GNN (Yasunaga et al., 2021)</td>
<td>82.8</td>
<td>~360M</td>
</tr>
<tr>
<td>T5* (Raffel et al., 2020)</td>
<td>83.2</td>
<td>~3B</td>
</tr>
<tr>
<td>T5 + KB (Pirtoaca)</td>
<td>85.4</td>
<td>≥11B</td>
</tr>
<tr>
<td>UnifiedQA* (Khashabi et al., 2020)</td>
<td>87.2</td>
<td>~11B</td>
</tr>
<tr>
<td><strong>GREASELM (Ours)</strong></td>
<td><strong>84.8</strong></td>
<td>~359M</td>
</tr>
</tbody>
</table>
One big issue: good commonsense KBs?

Some of the most well-known:

- wordnet
- Dbpedia
- wikidata
- yago
- babelnet
- conceptnet
- atomic
- nell
- framenet
- cyc
- adimen-sumo
- TPTP
- schema.org
Typical KG contents: simple triplets like
Yet another active area: improve KBs

Conversational Multi-Hop Reasoning with Neural Commonsense Knowledge and Symbolic Logic Rules

Forough Arabshahi
Facebook

Jennifer Lee
Facebook

Antoine Bosselut
EPFL

Yejin Choi
University of Washington

Tom Mitchell
Carnegie Mellon University

2021
Case study: add machine learning to a rule-based system

Our group at Taltech: T.Tammet, P.Järv, D.Draheim, M.Verrev

Three components:

• **A semantic parser**: English to full first order logic
• **A “softened” commonsense reasoner** GK
• **Improved versions of existing commonsense knowledge bases**

Plus a minor thing: **readable explanations** in English
A semantic parser for English

Why hard:

• A large number of possible syntactic parse trees
• Roles of objects in the sentence hard to understand: subject, object, using, helper, place, time, .... (“A man opened the door” vs “A key opened the door”)
• Coreference resolution and friends (“He saw it”)
• How to represent context/space/time etc?

....
Classic example

Ambiguity
Many frameworks for semantic parsing:

- Abstract Meaning Representation (AMR)
- Universal Conceptual Cognitive Annotation (UCCA)
- Elementary Dependency Structures (EDS)
- Discourse Representation Structures (DRS)
- Universal Decompositional Semantics (UDS)

But no good parsers, i.e. suitable for our purposes!
Our approach to semantic parsing: DIY

• We take the output of the Stanza Universal Dependencies Parser (Stanford)
• And then convert the result to 1\textsuperscript{st} order logic enhanced with confidences and exceptions.

Stanza uses machine learning on labelled sentences to determine the correct parse.

The UD framework gives fairly detailed information about the roles of the parts of the sentence.

We get ca 95% of the Hans adversarial benchmark right: the remaining 5% are Stanza misinterpretations.
Automated reasoner

- Almost all serious work in automated reasoning is either:
  - Research in logic
  - Improving AR for verification (SMT solvers etc)
  - Improving AR for mathematics

- Very little effort has ever gone to automating common sense.
Our commonsense reasoner GK (Graph Knowledge)

GK is built on top of our high-performance classical reasoner GKC

http://logictools.org/gk

CASC prover competition ("the world championship for such systems") on FOL during FLOC 2022:

Our base system GKC for classical FOL
Note: it can be pretty hard to find solutions

• Finding a solution is semidecidable: no guarantee that we can show that a solution does not exist.

• Even small problems can be hard. A solvable tiny example from group theory, ca 57 steps in a proof:

\[
multiply(A, \text{inverse}(multiply(B, multiply(C, multiply(multiply(inverse(C), inverse(multiply(D, B))), A)))))) = D.
multiply(\text{inverse}(a1), a1) \neq multiply(\text{inverse}(b1), b1).
\]
Automated reasoning needs a wealth of capabilities for commonsense A.I. to be usable for a hybrid AR + ML system

- Find answers, i.e. substitutions
- Survive contradictions
- **Estimate confidences**
- Handle defeasible reasoning
- Handle analogy-based reasoning
- Manage context
- Integrate with NLP systems
- Integrate with knowledge bases

*Implemented in the GK reasoner*

*CADE 2021 paper*

*Work in progress*
Practical context: NLP Q&A / dialogue

• Suppose we use an AR component as a part of an NLP Q&A / dialogue system.
• Most facts and rules are uncertain: some more, some less.
• We need to estimate our confidence in the results of the derivations.

Using real probabilities does not seem to be realistic for this scenario.
Current approaches outside fuzzy logic

• Programming tools for real probabilities with distributions, etc
• Propositional logic + probabilities: Bayesian networks
• Logic programming with probabilities: grounding a la Problog
• Markov logic networks, also grounding

Systems for the last two cannot handle

1.0 :: p(a).  
1.0 :: p(i(a,b)).  
1.0 :: p(Y) :- p(X), p(i(X,Y)).  
query( p(b) ).
Sources of confidence

We follow the subjective interpretation of probability as a degree of belief, originating from Ramsey and De Finetti.

We assume that the confidence in a fact or rule in our common sense knowledge base typically arises from:

- a large number of human users via crowd-sourcing like ConceptNet
- NLP-analyzed scraped text from the web like NELL
- machine learned rules like the work on improving KBs
- combining different knowledge bases with weights like Quasimodo
- assumptions accompanying a question
Assigning confidences to statements

• To each FOL statement $S$ we will assign both a confidence $c$ and a set $L$ of unique identifiers of (non-derived) input statements used for deriving this statement: a triple $<S, c, L>$.  
• Each $S$ in the triple is a clause. 
• Lists of such triples are then treated as sets. 
• The dependency lists $L$ are used in the formula estimating the cumulated confidence. 
• NB! For each single FOL clause $S$ there may be many different derivable triples $<S, c, L>$ for different $c$ and $L$, stemming from different derivation trees of $S$. 
Interpretation of confidences

• Confidence estimates the lower limit of the probability of a statement: 
  \(<S, c, L>\) means “statements L support the claim that probability(S) ≥ c”.

• An example, from which we derive bird(a) : 0:682

  <bird(X), 0.1, L1>
  <bird(a), 0.8, L2>
  <bird(a), 0.9, L3>
  <not bird(a), 0.3, L4>
Preliminaries

• Extends the resolution method: underlying FOL kept intact.
• The exact maximal confidence for derived statements is not calculated.
• Assume the question posed is in one of two forms:
  (1) Is the statement Q true?
  (2) Find values V for existentially bound variables in Q so that Q is true.

exist \( X_1, \ldots, X_n \) \((Q(X_1, \ldots, X_n) \& \text{not answer}(X_1, \ldots, X_n))\)

• A clause consisting of only answer predicates is called an “answer clause”
Main points of the algorithm

• Contradictions not containing an answer clause are discarded.
• The proof search does not stop whenever an answer clause is found, but will continue to look for new answer clauses until a predetermined time limit is reached.
• Resolution / paramodulation rules decrease the confidence of the result by multiplying of the confidences of arguments.
• Different independent derivations of the same answer clause are combined to cumulatively increase the confidence of the result.
• Attempt to find proofs for negations of the instantiated query answers.
• The difference of confidences of positive / negative query are final numbers.
Time limits are crucial

Let $t$ be total time given.
Spend $t / 2$ time for finding positive answers, combined to $N$ answers.
Spend $(t / 2) / N$ time for searching proofs for negations of answers.

**Optimistic approach:** more time for positive search.
**Pessimistic approach:** more time for negative search.
Algorithm 1 CONFER algorithm

Input: Common sense knowledge base KB, question Q, time limit t.
Output: Set of answers R with attached confidences.

1: Let R=\{\}.
2: Find a set of initial positive answers with confidences and dependencies
   \( IPA=\{\langle A_1, c_1, L_1 \rangle, ..., \langle A_p, c_p, L_p \rangle \} \) for Q from KB using c-resolution with the time limit \( t/2 \).
3: Calculate a set of cumulative positive answers \( CPA=\{\langle B_1, d_1, E_1 \rangle, ..., \langle B_r, d_r, E_r \rangle \} \) from IPA.
4: Let \( i = 1 \).
5: while \( i <= r \) do
6:     Form the negated question \( NQ_i \) from \( \neg Q \) with a substition \( s \) given by \( B_i \).
7:     Find a set of initial negative answers \( N_i \) with confidences and dependencies for
        \( NQ_i \) from KB using c-resolution with the time limit \( t/(2 \times r) \).
8:     if \( N_i \) is empty then
9:         Let \( nc_i = 0 \).
10:    else
11:        Calculate the cumulative negative confidence \( nc_i \) from \( N_i \).
12:    end if
13:    Add a pair \( \langle B_i, (c_i - nc_i) \rangle \) to \( R \).
14:    Let \( i = i + 1 \).
15: end while
16: For each pair \( \langle B_i, d_i \rangle, \langle B_j, d_j \rangle \) in \( R \) where \( i \neq j \), \( B_i = B_j \) and \( d_i \geq d_j \), remove
    \( B_j : d_j \) from \( R \).
17: Remove from \( R \) all elements \( \langle B_i, d_i \rangle \) where \( d_i \leq 0 \).
18: return the set of answers with confidences \( R \).
Basic confidence calculation for resolution steps

Resolution and paramodulation: multiply the confidences of the premises. Factorization: unchanged. Question clauses have a confidence 1.

Example: from
0.8:: bird(tweety).
0.9:: bird(X) => canfly(X).
0.7:: canfly(X) => fast(X).
1.0:: fast(X) => answer(X).

we derive
0.72:: canfly(tweety).
0.504:: fast(tweety).
0.504:: answer(tweety).
A clause $C$ can be derived in different ways, giving two different derivations $d_1$ and $d_2$ with confidences $c_1$ and $c_2$.

In case the derivations $d_1$ and $d_2$ are independent, we could apply the standard formula $P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$

We will estimate the independence $i$ of two derivations $d_1$ and $d_2$ simply as

$$1 - \frac{\text{number of shared input clauses of } d_1 \text{ and } d_2}{\text{total number of input clauses in } d_1 \text{ and } d_2}$$

Thus, if no clauses are shared between $d_1$ and $d_2$, then $i = 1$
and if all the clauses are shared, then $i = 0$.  

Cumulative confidence: independence
Cumulative confidence calculation

Given two derivations \( d_1 \) and \( d_2 \) of the search result \( C \) with confidences \( c_1 \) and \( c_2 \), calculate the updated confidence of \( C \) as

\[
\max(c_1 + c_2*i*h, \quad c_1*i*h + c_2) - c_1*c_2*i*h
\]

where

- independence of derivations \( i \) is defined as above,
- \( h \) is the heuristic estimate of the independence of the total set of input clauses from 1 for total independence to 0 for total dependence.
Cumulative confidence calculation: intuition

If d1 and d2 do not share non-question input clauses and all the input clauses are mutually independent, $i*h = 1$ and the formula turns into

$$c_1 + c_2 - (c_1 * c_2).$$

If d1 and d2 have the same non-question input clauses or the total set of input clauses is mutually totally dependent, $i*h = 0$ and the formula turns into

$$\max(c_1, c_2).$$
Cumulative confidence: example

Example: from

0.8:: bird(tweety).
0.9:: bird(X) => canfly(X).
0.7:: canfly(X) => fast(X).
0.8:: inair(tweety).
0.9:: inair(X) => fast(X).
1.0:: fast(X) => answer(X).

we derive

0.72:: canfly(tweety).
0.504:: fast(tweety).
0.504:: answer(tweety).
0.72:: fast(tweety).
0.72:: answer(tweety).

Derivations d1 and d2 do not share clauses, thus i=1. Let heuristic be 1 as well. Then we get cumulative confidence

$$0.504 + 0.72 - (0.504 \times 0.72) = 0.86112.$$ 

Now let heuristic $h=0.5$ (partial independence). Then we get cumulative confidence

$$\max(0.504 + 0.72 \times 0.5 - (0.504 \times 0.72 \times 0.5),$$

$$0.504 \times 0.5 + 0.72 - (0.504 \times 0.72 \times 0.5)) = \max(0.68256, 0.79056) = 0.79056$$

Setting $h$ to 0 (total dependence) leads to cumulative confidence

$$\max(0.504, 0.72)= 0.72$$
The systems we use for comparison

ProbLog2
• A leading system for probabilistic logic programs
• Live web site https://dtai.cs.kuleuven.be/problog/

Alchemy 2
• A leading system for Markov Logic
• Versions from https://github.com/PhDP/alchemy2
• Three different algorithms: MC-SAT, exact and approximate probabilistic theorem proving.
Negation examples: expected and unexpected

Consider

0.5 :: bird(a).
0.5 :: not bird(a).
query( bird(a) ).

• Confer gives confidence 0, which we interpret as “no information”, not as “false”.
• The Problog2 system gives 0.25, explained by one of the authors as “an atom (head) is satisfied if any of the rules that make it true fire and none of the rules that make it false fire, hence 0.5 * (1 – 0.5) = 0.25”
• The three different algorithms of the Alchemy 2 system give answers 0.015, 0 and 0.082, respectively.
Negation: consistencies and inconsistencies

Consider
0.5::bird(a).
0.5::not bird(a).
0.9:: flies(X) :- bird(X).
query(flies(_)).

• Confer gives us 0.45: inconsistent with the result of the previous example.
• ProbLog2 gives 0.225: unintuitive, but consistent with the unintuitive result of ProbLog2 in the previous example.
• The three algorithms of Alchemy 2 give us 0.047, 0 and 0.98.
Social network

0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :- influences(Y,X), smokes(Y).
query(smokes(carl)).

• Confer: 0.1201, cumulating values 0.096 and 0.08
• Problog2: 0.1376
• Alchemy2 algorithms: 0.135, 0 and 0.741
Earthquake

person(john).
person(mary).
0.7::burglary.
0.2::earthquake.
0.9::alarm :- burglary, earthquake.
0.8::alarm :- burglary, \+earthquake.
0.1::alarm :- \+burglary, earthquake.
0.8::calls(X) :- alarm, person(X).
0.1::calls(X) :- \+alarm, person(X).
evidence(calls(john),true).
evidence(calls(mary),true).
query(burglary).
query(earthquake).

<table>
<thead>
<tr>
<th>query</th>
<th>CONFER</th>
<th>CONFER +</th>
<th>CONFER -</th>
<th>Problog</th>
<th>Alchi</th>
<th>Alche</th>
<th>Alcha</th>
</tr>
</thead>
<tbody>
<tr>
<td>burglary</td>
<td>0.8713</td>
<td>0.97650</td>
<td>0.1051</td>
<td>0.9819</td>
<td>0.709</td>
<td>0</td>
<td>0.905095</td>
</tr>
<tr>
<td>earthquake</td>
<td>0.1648</td>
<td>0.8854</td>
<td>0.7206</td>
<td>0.2268</td>
<td>0.204</td>
<td>0</td>
<td>0.888</td>
</tr>
</tbody>
</table>
Performance

<table>
<thead>
<tr>
<th>Problem</th>
<th>CONFER</th>
<th>CONFER 1.0</th>
<th>gkc pure</th>
<th>gkc full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steamroller</td>
<td>0.0018</td>
<td>0.0015</td>
<td>0.001</td>
<td>0.06</td>
</tr>
<tr>
<td>Dreadbury</td>
<td>0.0017</td>
<td>0.0011</td>
<td>0.001</td>
<td>0.06</td>
</tr>
<tr>
<td>Lukasz 0</td>
<td></td>
<td>0.0916</td>
<td>0.093</td>
<td>0.22</td>
</tr>
<tr>
<td>Lukasz 1</td>
<td></td>
<td>0.913</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lukasz 2</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lukasz 3</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lukasz 4</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSR025+5</td>
<td>0.0004</td>
<td>0.0001</td>
<td>0.0001</td>
<td>4.5</td>
</tr>
<tr>
<td>CSR035+5</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.07</td>
<td>4.6</td>
</tr>
<tr>
<td>CSR045+5</td>
<td>3.418</td>
<td>1.4</td>
<td>1.3</td>
<td>5.8</td>
</tr>
<tr>
<td>CSR055+5</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>4.5</td>
</tr>
</tbody>
</table>
Automated reasoning needs a wealth of capabilities for commonsense A.I. to be usable for a hybrid AR + ML system

• Find answers, i.e. substitutions
• Survive contradictions
• Estimate confidences
• Handle defeasible reasoning
• Handle analogy-based reasoning
• Manage context
• Integrate with NLP systems
• Integrate with knowledge bases

Implemented in the GK reasoner

IJCAR 2022 paper

Work in progress
A small example of real NL input / output

Birds can fly. Penguins are birds. Penguins cannot fly.
Patrick is a penguin and Billie is a bird.
Mickey is perhaps a bird. Theo is possibly a penguin.

Who can fly?
Billie and maybe Mickey. (GPT3: Billie can fly)

Who cannot fly?
Patrick and likely Theo. (GPT3: Theo is possibly a penguin and cannot fly)

Patrick can fly?
No. (GPT3: False)

Billie can swim?
Unknown. (GPT3: Billie can swim if she is a bird)
Explanation example

Billie:

Sentences used:
(1) Birds can fly.
(2) Patrick is a penguin and Billie is a bird.
(3) Who can fly?

Statements inferred:
(1) If X is a bird, then X can fly, except when X cannot fly. Why: sentence 1.
(2) Billie is a bird. Why: sentence 2.
(3) Billie can fly, except when Billie cannot fly. Why: statements 1, 2.
(4) If X can fly, then X has a property in the question. Why: sentence 3.
(5) Billie has a property in the question, except when Billie cannot fly. Why: statements 3, 4.
(6) If X has a property in the question, then X is an answer. Why: the question.
(7) Billie is an answer, except when Billie cannot fly. Why: statements 5, 6.

Exceptions checked and not holding: Billie cannot fly.
A more complex example from Allen A.I.

The circuit has a switch.
The circuit has a bell.
The switch is off.

If the circuit has the switch and the switch is on then the circuit is complete.
If the circuit does not have the switch then the circuit is complete.
If the circuit is complete and the circuit has the light bulb then the light bulb is glowing.
If the circuit is complete and the circuit has the bell then the bell is ringing.
If the circuit is complete and the circuit has the radio then the radio is playing.

The bell is ringing?
GK: No.  GPT-3: Yes
The standard example of default logic

penguin(p)
bird(b)
∀x. penguin(x) ⇒ bird(x)
∀x. penguin(x) ⇒ ¬ fly(x)

And the default rule  

bird(x) : fly(x) ⊨ fly(x)

we encode this as  

¬ bird(x) V block(0, neg(fly(x))) V fly(x)
The standard example of default logic

Goal: \( \text{fly}(x) \Rightarrow \text{ans}(x) \)
The prover runs for \( N \) seconds, deriving clauses, including some answer clauses:

....
ans(b)
...
ans(p) V block(0,neg(fly(p)))
....

Now check whether \( \neg \text{fly}(p) \) is provable, i.e. \( \text{fly}(p) \) gives a contradiction:
it does, without any blockers in the result.

Hence \( b \) is the sole answer.
Finite domains vs a non-omniscient system

Since FOL is not decidable, it is in general impossible to guarantee that some blocker literal is not derivable.

Hence, the standard approach for handling default logic has been creating a large ground instance KBg of the KB, and then performing decidable propositional reasoning on the KBg. This is what ASP systems do.
Finite domains vs a non-omniscient system

Our approach is to avoid grounding and accept that the system is not logically omniscient:

- Justification checking of blockers (i.e. exceptions) is delayed until a first-order proof is found; after that recursively deepening checks are performed with diminishing time limits.
- GK first produces a potentially large number of different candidate proofs and then enters a recursive checking phase.
- The results produced by GK depend on the time limits and are not stable in the general case.
Set time limit $L$ and a trust threshold confidence $C_t$

1. Calculate **positive evidence** with a total confidence $C_p$
   - Collect $N_p$ proofs for the query during $L/4 - \epsilon$ time
   - Attempt to **recursively invalidate each of 1 ... $N_p$ proofs** giving $(L/4 - \epsilon) / N_p$ time
     - ... A proof is invalidated if some blocker is proved with a confidence over $C_t$

2. Calculate **negative evidence** with a total confidence $C_n$: symmetric to the procedure above

3. Combine positive / negative evidence
   - If $C_p > C_n$, give a validated combined proof of the query with the confidence $C_p - C_n$,
   - else a validated combined proof of the negated query with the confidence $C_n - C_p$
Priorities: an ordering of default rules

The concept of priorities for default rules has been well investigated, with several mechanisms proposed. Defaults are typically ordered by specificity:

Default rules for a more specific class of objects should take priority over rules for more general classes.

For example,

• Birds (who typically do fly) are physical objects.
• Physical objects typically do not fly.
• Hence we have contradictory default rules.
• Since birds are a subset of physical objects, the flying rule of birds should have a higher priority than the non-flying rule of physical objects.
Encoding priorities

- We encode priority information as a first argument of the blocker literal, offering several ways to determine priority: an integer, a taxonomy class number, a string in a taxonomy or a combination of these with an integer.

- For automatically using specificity we employ taxonomy classes: a class has a higher priority than those above it on the taxonomy branch.

- To enable more fine-grained priorities, an integer can be added to the term like $("bird", 2)$ generating a lexicographic order.
Fine-grained priorities are important

Levels like the “Physical objects cannot fly”, “Birds can fly” and “Penguins cannot fly” are solved using taxonomy-based priorities.

But consider “Birds can fly” and “Baby birds cannot fly”: there is no class taxonomy involved.

We need to be able to attach a potentially arbitrarily deep list of lexicographically comparable priority information to each exception. Consider “Birds can eat meat” and “Baby birds cannot eat raw meat”.

The main modification to the FOL reasoner

Limit subsumption and simplifications with the requirement:

A triple $T_1 = \langle A_1, c_1, L_1 \rangle$ consisting of a clause $A_1$, confidence $c_1$ and a dependency list $L_1$ subsumes a triple $T_2 = \langle A_2, c_2, L_2 \rangle$ if and only if

- $A_1$ subsumes $A_2$ in the standard sense,
- For blockers $\text{block}(s_1,t)$ in $A_1$ subsuming a blocker $\text{block}(s_2,g)$ in $A_2$, $s_2$ is not stronger than $s_1$.
- If $L_2$ contains a goal clause, $L_1$ also contains a goal clause
- $c_1 \geq c_2$ and
- $L_1$ is a (non-strict) subset of $L_2$
Current ASP limits

Consider a standard example in ASP syntax

\[
\begin{align*}
bird(b1). \\
penguin(p1).
bird(X) & : penguin(X). \\
flies(X) & : bird(X), \text{ not } -flies(X). \\
-flies(X) & : penguin(X).
\end{align*}
\]

Ask \texttt{flies(b1) and flies(p1)}.

All default-solving ASP systems can solve these very quickly.
Adding function symbols

When we add the rules

bird(father(X)) :- bird(X).
penguin(father(X)) :- penguin(X).

Then no ASP systems we tried terminate on the previous queries. s(CASP) terminates, when the previous formulation is modified as

flies(X) :- bird(X), not abs(X).
abs(X) :- penguin(X).
Adding transitivity

When we instead add facts and rules

\[ \text{father}(b1,b2). \]
\[ \text{father}(p1,p2). \]
\[ \vdots \]
\[ \text{father}(bN-1,bN). \]
\[ \text{father}(pN-1,pN). \]
\[ \text{ancestor}(X,Y) : - \text{father}(X,Y). \]
\[ \text{ancestor}(X,Y) : - \text{ancestor}(X,Z), \text{ancestor}(Z,Y). \]

for a large \( N \), \( s(\text{CASP}) \) does not terminate and two other systems become slow for \( \text{flies}(b1) \): ca 8 seconds for \( N = 500 \) and ca 1 minute for \( N = 1000 \). GK solves the same question with \( N = 1000 \) under half a second and with \( N = 100000 \) under three seconds
A simple multi-level example with GK

["flyingpenguin", "?:X1"], "=>", ["penguin", "?:X1"]],
["penguin", "?:X1"], "=>", ["bird", "?:X1"]],
["bird", "?:X1"], "=>", ["organism", "?:X1"]],
["organism", "?:X1"], "=>", ["object", "?:X1"]],

["penguin", "?:X1"], "<=>", ["penguin", ["father", "?:X1"]]],
["bird", "?:X1"], "<=>", ["bird", ["father", "?:X1"]]],
["flyingpenguin", "?:X1"], "<=>", ["flyingpenguin", ["father", "?:X1"]]],

["penguin", "?:X1"], "=>", ["fly", "?:X1"], ["$block", "$", "penguin", 3, ["not", ["fly", "?:X1"]]]],
["bird", "?:X1"], "=>", ["-fly", "?:X1"], ["$block", "$", "penguin", 2, ["fly", "?:X1"]]]],
["organism", "?:X1"], "=>", ["fly", "?:X1"], ["$block", "$", "organism", ["fly", "?:X1"]]]],

["flyingpenguin", "?:X1"], "<=>", ["fly", "?:X1"], ["$block", "$", "penguin", 3, ["not", ["fly", "?:X1"]]]],
["bird", "?:X1"], "<=>", ["-fly", "?:X1"], ["$block", "$", "bird", ["not", ["fly", "?:X1"]]]],

["organism", "?:X1"], "<=>", ["fly", "?:X1"], ["$block", "$", "organism", ["fly", "?:X1"]]]],

["flyingpenguin", "fp"],
[penguin", "p"],
["bird", "b"],
["organism", "o"],

//@question": ["fly", ["father", "?:X"]]
//@question": ["fly", ["father", "?:X"]]
//@question": ["fly", ["father", ["father", "p"]]]
]
Proof search trace outline

= top level: starting main proof search for positive evidence for 2250.000000 mseconds =

**** depth 0 run 1 starts with strategy

{"result": "answer found",
"answers": [
{
"answer": true,
"blockers": [["$block","$","bird"], ["$not","fly","father","father","p"]]]
},

"confidence": 1,
...

= top level: starting to check blockers =

**** depth 1 run 1 starts with strategy

{"result": "answer found",
"answers": [
{
"answer": true,
"blockers": [["$block","$","penguin",2], ["fly","father","father","p"]]]
},

"confidence": 1,
...
**** depth 2 run 1 starts with strategy
search terminated without proof.

= top level: starting to search negative evidence for 2250.000000 mseconds=

"result": "answer found",
"answers": [
{
"answer": true,
"blockers": [["$block","$","penguin",2],["fly","father","father","p"]],
"confidence": 0,
"negative proof":
[
[1, 12, ["in", "frm_9", "axiom", 1, []], ["$block","$","penguin",2],["fly","?X"]], ["penguin","?X"], ["fly","?X"]],
[2, 5, ["in", "frm_5", "axiom", 1, []], ["penguin","father","?X"], ["penguin","?X"]],
[3, 16, ["in", "frm_13", "axiom", 1, []], ["penguin","p"]],
[4, 47461, ["mp", [2,1], 3, "fromaxiom", 1, [5,16]], ["penguin","father","p"]],
[5, 5, ["in", "frm_5", "axiom", 1, []], ["penguin","father","?X"], ["penguin","?X"]],
[6, 47465, ["mp", [4,5,1], "fromaxiom", 1, [5,16]], ["penguin","father","father","p"]],
[7, 20, ["in", "$auto_negated_question", "goal", 1, []], ["fly","father","father","p"]],
[8, 47540, ["mp", [1,1], 6, 7, "fromgoal", 1, [12,5,16]],
[["$block","$","penguin",2],["fly","father","father","p"]]]]]}
Proof search trace outline and final proof

*** blocker check for negative evidence ****

**** depth 1 run 1 starts with strategy search terminated without proof.

*** final result combination and printout ****
= showing final result =
"result": "answer found",
"answers": [ 
  ( "answer": false,
    "blockers": ["$block","$","penguin",2],
                ["fly",["father","father","p"]]]],
  "confidence": 1,
  "negative proof": [
  [1, 12, ["in", "frm_9", "axiom", 1, []], [["$block","$","penguin",2],
                        ["fly","?:X"]], ["-penguin","?:X"],
                        ["-fly","?:X"]],
  [2,  5, ["in", "frm_5", "axiom", 1, []], ["penguin","father","?:X"]],
  [3, 16, ["in", "frm_13", "axiom", 1, []], ["penguin","p"]],
  [4, 47461, ["mp", [2,1], 3, "fromaxiom", 1, [5,16]],
              [["penguin","father","p"]]]],
  [5,  5, ["in", "frm_5", "axiom", 1, []], ["penguin","father","?:X"]],
  [6, 47465, ["mp", 4, [5,1], "fromaxiom", 1, [5,16]],
              [["penguin","father","p"]]]],
  [7, 20, ["in", "$auto_negated_question", "goal", 1, []],
              [["fly","father","father","p"]]]],
  [8, 47540, ["mp", [1,1], 6, 7, "fromgoal", 1, [12,5,16]],
              ["$block","$","penguin",2],
              ["fly","father","father","p"]]]]
]}
Analogical reasoning: similar to defaults

queen(mary)
king(X) => rich(X)
king(X) => male(X)
queen(X) => female(X)

queen is similar to king

? rich(mary)
? male(mary)
Using Quasimodo with GK

- Quasimodo is a large commonsense KB enhanced with numeric confidences, built by a group in Max Planck.
- Size is ca 200 megabytes, contains over million rules/facts.
- We have converted Quasimodo and added default conditions to most rules.
- Used by reading the whole KB into shared memory along with the Wordnet taxonomy classes.
- GK is able to solve simple examples using default logic and the whole Quasimodo.
- It can also use a large converted subset of ConceptNet along with Quasimodo.
## Quasimodo data

<table>
<thead>
<tr>
<th>Subject</th>
<th>Relation</th>
<th>Object</th>
<th>$s$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>bird</td>
<td>can</td>
<td>flying</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>bird</td>
<td>fly</td>
<td>over the acropolis</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>birds eye</td>
<td>be protected</td>
<td>from winds when flying</td>
<td>0</td>
<td>0.38</td>
</tr>
<tr>
<td>penguin</td>
<td>can</td>
<td>fly</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>penguin</td>
<td>can</td>
<td>fly</td>
<td>0</td>
<td>0.42</td>
</tr>
<tr>
<td>penguin</td>
<td>has_property</td>
<td>unable to fly</td>
<td>0</td>
<td>0.76</td>
</tr>
</tbody>
</table>

Table 1: Excerpt of Quasimodo data
Conversion example:
Penguins are birds, penguins cannot fly:

\[ \text{IsA(bird, penguin)} : 0.96 \]

A known relation, also present in Table 1. The number 84487 is the identifier of "penguin" from the taxonomy graph. This rule means that we have 0.99 confidence that penguins do not fly, unless there is some more specific penguin that has this capability:

\[ \text{IsA(penguin, X)} \rightarrow (\neg \text{Capability}(\text{fly}, X)) \lor \\
\text{Sblock(84487, Capability(fly, X))) : 0.99} \]

An generic subject-verb-object relation "eat" with no blocker. This rule contains a compound noun "leopard seal" encoded as one term:

\[ \text{IsA(leopard seal, X)} \rightarrow \text{SVO(penguin, eat, X)} : 0.92 \]
A conversion example with basic NL parsing

Example 4 (Fragment parsing). The negative Quasi-modo fact ”Northern Ireland, have, rugby team” is encoded as:

$$\text{IsA(northern}_{-}\text{ireland,} X) \rightarrow \neg\text{HasA(sk}(X), X) \wedge$$

$$\text{IsA(team, sk}(X)) \wedge$$

$$\text{Property(rugby, sk}(X))$$

”Northern Ireland” is a named entity, encoded as one term. The fragment ”rugby team” is split by the parser. The rule is skolemized because new variables introduced to the right hand side must be existentially bound to a left hand side variable.
Work to do

Theory: completeness questions:
• Define sensible completeness criterias
• Prove the completeness of the algorithm for several subclasses

Practice: employ machine learning and simpler statistics:
• Similarity reasoning: word/relation similarity measures and exceptions
• Clause selection using NLP semantics and co-occurrence embeddings
• Use machine learning to speed up search
• Improve NLP parsing
• Modify, improve and integrate existing commonsense KBs