Knowledge representation

lecture 7

Some applications
of first order provers

https://i.redd.it/4i4uv8qsh7yz.gif

Tanel Tammet

TTU

Lecture overview

Recall: three main kinds of reasoners (there ar e others ...)

An overview of using first order provers: where and how

Using provers in various kinds of mathematics

Nontrivial examples with Otter

Formal verification

Using provers for commonsense reasoning in natural language processing

Three main kinds

- First order aka predicate logic (for classical logic)
- Propositional logic
- SMT (satisfiability modulu theories): prop logic + limited, computable parts of first order logic

About propositional and SMT

Juhan Ernits will talk about the applications in the next lecture

First order logic applications

Very rarely used in actual commercial software development.

Why: part of AI, not something you can easily use for creating a user interface or reports etc.

First order logic and learning?

What makes logic hard: writing rules.

Normal people have really hard time encoding knowledge in formal rules.

Additionally, real-life rules are probabilistic/fuzzy, not certain.

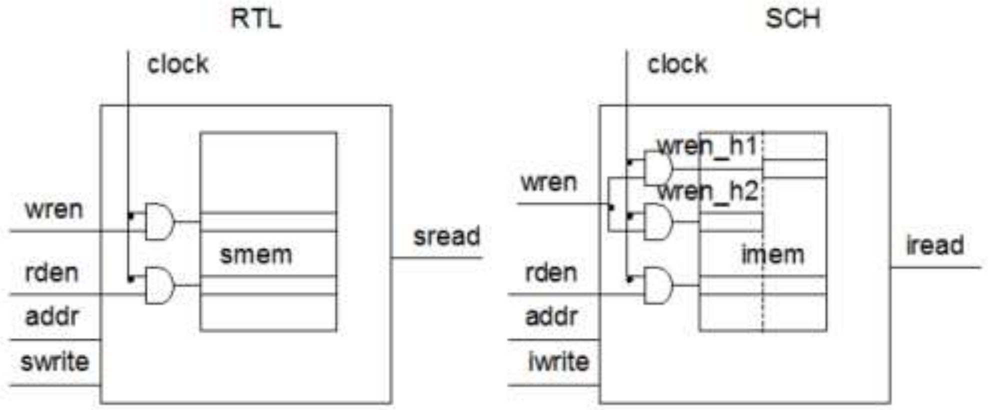
Hence, real hope or need is to start learning rules!

Once we have learned lots of rules, we could start using the rules, i.e. deriving new information.

Automated Reasoning

- What is Reasoning? Solving problems by syntactic manipulations.

  Hardware: Are these two hardware designs equivalent?



  Software: Does your program accesses unallocated memory?

  Math: Does this equation $(xy)^{-1} = y^{-1}x^{-1}$ hold in all groups?

  Knowledge management:
  Can we represent and analyse all available knowledge about human body ?

  Automated reasoning: can we solve all these problems automatically ?

The scale of automation required

Intel floating arithmetic bug cost $475 million.

Intel i7 Haswell-E 2,600,000,000 gates

Software bugs cost billions.

40 Mil LOC

Major companies: Intel, Microsoft, Airbus, NASA intensively use formal methods.

In mathematics

Erdős discrepancy problem proved by a SAT solver (2014):

INDEPENDENT

Computer cracks Erdős puzzle – but no human brain can check the answer

12GB proof

Largest math proof ever: Pythagorean triples problem was proved by a SAT solver (2016)

nature

NATURE | NEWS

Two-hundred-terabyte maths proof is largest ever

A computer cracks the Boolean Pythagorean triples problem — but is it really maths?

200TB proof

Applications of automated reasoning

Applications:

- software and hardware verification: Intel, Microsoft
- information management: biomedical ontologies, semantic Web, databases
- combinatorial reasoning: constraint satisfaction, planning, scheduling
- Internet security
- Theorem proving in mathematics

John McCarthy

"It is reasonable to hope that the relationship between computation and mathematical logic will be as fruitful in the next century as that between analysis and physics in the past." McCarthy, 1963.

Realities

Provers are – in an important sense – incredibly weaker than a brain.

Real mathematics (what mathematicians do) is very hard. Provers have managed to solve open problems only some rare cases.

Even mathematics is very hard to formalize, though doable.

Formalizing electronics is sometimes feasible, but really hard. In these cases prop solvers and SMT solvers work best.

We do not understand well how to fomalize common sense thinking.

Common sense thinking relies on an enormous amount of uncertain rules.

Hope!

Maybe we can learn commonsense rules using specialized learning techniques.

In parallel, maybe we can start automatically reading simple texts and automatically building statistical rules for commonsense reasoning.

After that we could use automated reasoning as a method for applying statistical rules we have learned.

Examples from mathematics

Can we axiomatize „all mathematics“ at once?

Two answers:

- Gödel taught us that mathematics is not finitely axiomatizable.
- However, very large sensible parts have been axiomatized. See eg set theory in first order logic:
  - https://en.wikipedia.org/wiki/Zermelo%E2%80%93Fraenkel_set_theory
  - http://www.cs.miami.edu/~tptp/cgi-bin/SeeTPTP?Category=Axioms&File=SET009^0.ax
  - http://www.cs.miami.edu/~tptp/cgi-bin/SystemOnTPTP look for SET...

TPTP

TPTP: the largest existing set of formalized first order problems

http://www.cs.miami.edu/~tptp/

along with

- converters
- an online tool capable of running all the top-level provers over the web interface.
- yearly prover competitions http://www.cs.miami.edu/~tptp/CASC/
- see also http://www.satcompetition.org/ for SAT competitions

Some nontrivial examples for otter

Let us examine

http://lambda.ee/w/images/0/03/Problems.tar.gz

Check these recommendations:

http://lambda.ee/wiki/Soovitusi_Otteri_otsingu_suunamiseks

Classic otter prover doing math

- Old results:

  http://www.mcs.anl.gov/research/projects/AR/new_results/

- Famous result: open problem - Robbins algebra ... – solved

  http://www.cs.unm.edu/~mccune/papers/robbins/

- „See Otter digging for algebraic pearls“:

  http://www.math.md/files/qrs/v10-n1/v10-n1-(pp95-114).pdf

Induction!?

Let us define the properties of a list member function m where c adds a new element to the list. n stands for empty list and a list [1,5] is represented as c(1, c(5, n))

```
-m(X,n).
m(X,c(X,Y)).
X=Z | m(X,Y) | -m(X,c(Z,Y)).
-m(X,Y) | m(X,c(Z,Y)).
```

Your goal is to prove that m has the expected property;

```
All x All y Exists z All u ((m(u,x) & m(u,y)) <=> m(u,z))
```

Induction!?

We need to prove two statements:

Induction base:

```
All y Exists z All u ((m(u,n) & m(u,y)) <=> m(u,z))
where n stands for empty list
```

Induction step:

```
All x ((All y Exists z All u ((m(u,x) & m(u,y)) <=> m(u,z)))
 =>
 (All h All y1 Exists z1 All u1 ((m(u1,c(h,x)) & m(u1,y1))
                        <=> m(u1,z1)))
```

NB! There is no „easy“ way to guess what is a required base and step.

Formal verification

Verification vs. Simulation

$$(x+1)^2 = x^2 + 2x + 1$$

X	$(X+1)^2$	X^2+2X+1
0	1	1
1	4	4
2	9	9
…	…	…

1	$(X+1)^2=(X+1)(X+1)$
2	$(X+1)(X+1)=(X+1)X+(X+1)1$
3	$(X+1)1=X+1$
4	$(X+1)X=XX+1X$
5	$XX=X^2$
…	….

Exhaustive Simulation Time

- Design: a 256-bit RAM.
- $2^{256} = 10^{80}$ possible combinations of initial states and inputs
- Assume:
  - Use all matter in our galaxy (10^{17} kg) to build computers.
  - Each computer is of the size of single electron (10^{-30} kg).
  - Each computer simulates 10^{12} cases per second.
  - we started at the time of the Big Bang about 10^{10} years ago

→ We would just have reached the 0.05% mark of completing our task

Detecting Design Faults

To check the new implementation for functional correctness, we need:

1. a reference description:
   - either a specification
   - or a previous “golden” implementation.
2. a new implementation, resulting from
   - a refinement (synthesis) of the specification
   - or an optimization of the reference implementation.
3. a correctness relation which has to be established between the two specifications
   - (e.g. behavioral correctness).

Hardware Verification

Definition: Hardware verification

is the proof that a circuit or a system (the implementation) behaves according to a given set of requirements (the specification).

Formal verification

uses *mathematical reasoning* to prove that an implementation satisfies a specification

Consideration of *all cases is implicit* in formal verification.

Formal Verification Methods

Equivalence Checking

Compares optimized/synthesized model against original model

Model Checking

Checks if a model satisfies a given property

Theorem Proving

Proves implementation is equivalent to specification in some formalism

What is Model Checking? (cont.)

Typically used during RTL code development to debug the RTL model prior to synthesis.

Used concurrently with and/or prior to simulation.

FormalCheck is the name of Cadence’s Model Checking tool.

Currently the dominant formal verification tool.

What is Model Checking?

Relies on exhaustive state space search.

- exhaustive state space search is guaranteed to terminate, as the model is finite.

Major challenge:

- To fight state-explosion problem

Can uncover subtle design errors

Can handle large state spaces (10^120)

Quicker to start testing

- as it does not require vectors or a testbench.

Successfully used to find bugs in published standards

What is Model Checking?

Model Checking (Property Checking):

- An automatic technique for verifying finite-state reactive systems, (such as sequential digital circuits or communication protocols).
  - A reactive FSM is an FSM whose inputs come from the environment.
- For checking that a desired property holds in a finite state model of a system
- Was pioneered by Edmund Clarke, professor in the CS Dept of CMU, in 1981
  - (E.M. Clarke and E.A. Emerson. "Synthesis of Synchronization Skeletons for Branching Time Temporal Logic", in Logic of Programs workshop, Yorktown Heights, NY, May 1981.).

Applicability

Wrong assumptions:

- Formal methods can guarantee the perfectness of systems

Facts:

- can significantly enhance the trust, but not able to guarantee flawlessness.

Reasons:

- Only makes correctness statements with regard to a formal specification which can be faulty itself.
- Faults in verification program.
- Only design faults but not fabrication faults or faults during system usage.

Some tools

Verification Languages:
- "e".
- Synopsys Vera :
- SystemC SCV

Equivalence Checkers:
- Cadence Verplex :
- Synopsys Formality :
- Mentor FormalPro :
- Prover eCheck :
- homebrew EC

Assertion Languages:
- IBM Sugar/PSL
- 0-in Checkerware
- Verplex OVL
- System Verilog SVA
- Synopsys Vera OVA

NLP processing

NLP processing

„Simple“ parts can be done using either:

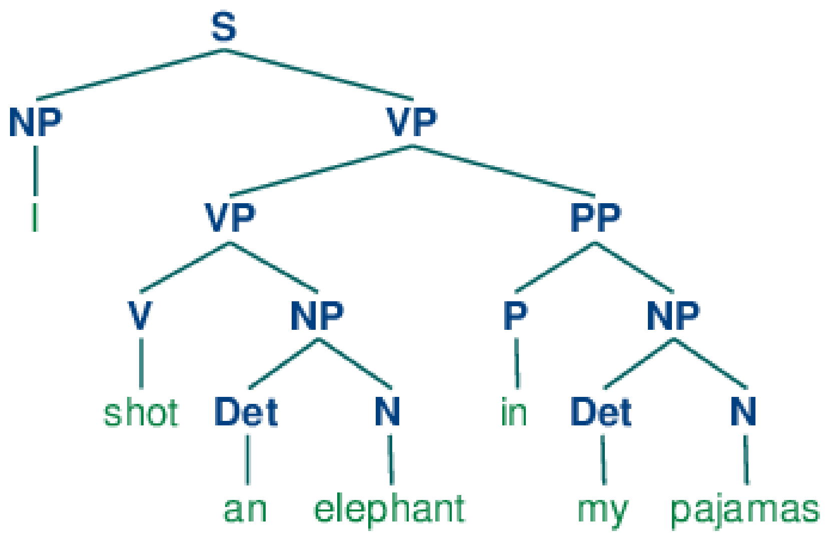
- „Simple“ rules (grammar, presence in Wikipedia, etc)
- Statistics and learned weight networks from large annotated text corpora

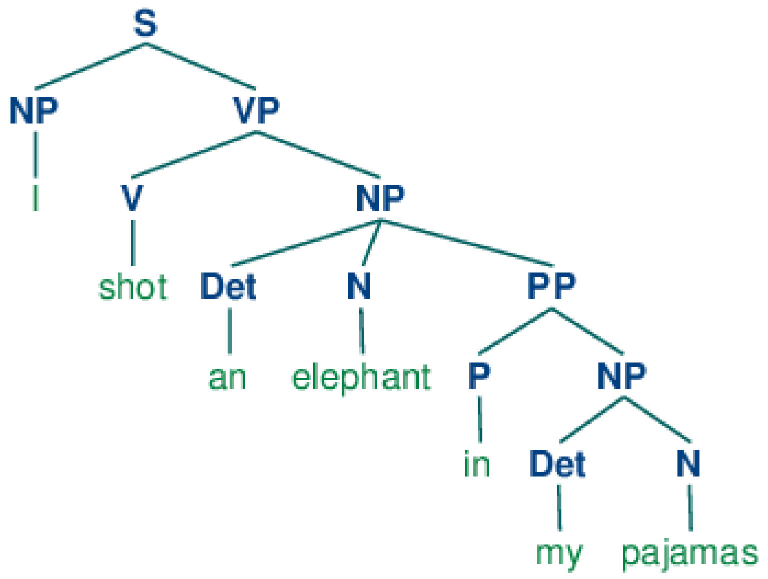
Hard parts require **commonsense understanding of the world**:

- Huge amounts of facts commonly known
- Huge amounts of probabilistic/fuzzy rules
- Weights for popularities, common usage patterns, etc

Ambiguity in grammar

Different possible parses of the sentence „I shot an elephant in my pajamas.“





Example

- Some roles prefer to be filled by certain kinds of NPs.
- This can give us useful features for classifying accurately:
  - "John ate the spaghetti with chopsticks." **(instrument)**

    "John ate the spaghetti with meatballs." **(patient)**

    "John ate the spaghetti with Mary."
    - Instruments should be tools
    - Patient of "eat" should be edible
  - "John bought the car for $21K." **(instrument)**

    "John bought the car for Mary." **(beneficiary)**
    - Instrument of "buy" should be Money
    - Beneficiaries should be animate (things with desires)
  - "John drove Mary to school in the van"

    "John drove the van to work with Mary.

NLP processing: textual entailment

Means: commonsense derivations from text.

Classical Entailment Definition

- Chierchia & McConnell-Ginet (2001):
  A text t entails a hypothesis h if h is true in every circumstance (possible world) in which t is true

- Strict entailment - doesn't account for some uncertainty allowed in applications

“Almost certain” Entailments

t: The technological triumph known as GPS … was incubated in the mind of Ivan Getting.

h: Ivan Getting invented the GPS.

Applied Textual Entailment

- A directional relation between two text fragments: *Text (t)* and *Hypothesis (h)*:

t entails h ($t \Rightarrow h$) if
humans **reading** *t* **will infer that** *h* is most likely **true**

- Operational (applied) definition:
  - Human gold standard - as in NLP applications
  - Assuming common background knowledge – which is indeed expected from applications

Probabilistic Interpretation

Definition:

- *t probabilistically entails h* if:
  - P(*h is true* | *t*) > P(*h is true*)
    - *t* increases the likelihood of *h* being true
    - ≡ Positive PMI – *t* provides information on *h*'s truth
- P(*h is true* | *t*): *entailment confidence*
  - The relevant entailment score for applications
  - In practice: "most likely" entailment expected

The Role of Knowledge

- For textual entailment to hold we require:
  - *text* AND *knowledge* $\Rightarrow$ *h*

  but
  - *knowledge* should not entail *h* alone
- Systems are not supposed to validate *h*'s truth regardless of *t* (e.g. by searching *h* on the web)

Gold standards

Annotated examples of text and sentence derivable from text:

Classical FRACAS example set:

https://nlp.stanford.edu/~wcmac/downloads/fracas.xml

Newer RTE example sets:

https://tac.nist.gov/data/RTE/index.html

Stanford Question Answering Dataset:

https://rajpurkar.github.io/SQuAD-explorer/

PASCAL Recognizing Textual Entailment (RTE) Challenges

EU FP-6 Funded PASCAL Network of Excellence
2004-7

Bar-Ilan University
MITRE
ITC-irst and CELCT, Trento
Microsoft Research

Generic Dataset by Application Use

- 7 application settings in RTE-1, 4 in RTE-2/3
  - QA
  - IE
  - "Semantic" IR
  - Comparable documents / multi-doc summarization
  - MT evaluation
  - Reading comprehension
  - Paraphrase acquisition
- Most data created from actual applications output
- RTE-2/3: 800 examples in development and test sets
- 50-50% YES/NO split

RTE Examples

	TEXT	HYPOTHESIS	TASK	ENTAIL-MENT
1	*Regan attended a ceremony in Washington to commemorate the landings in Normandy.*	*Washington is located in Normandy.*	IE	False
2	*Google files for its long awaited IPO.*	*Google goes public.*	IR	True
3	*….: a shootout at the Guadalajara airport in May, 1993, that killed Cardinal Juan Jesus Posadas Ocampo and six others.*	*Cardinal Juan Jesus Posadas Ocampo died in 1993.*	QA	True
4	*The SPD got just 21.5% of the vote in the European Parliament elections, while the conservative opposition parties polled 44.5%.*	*The SPD is defeated by the opposition parties.*	IE	True

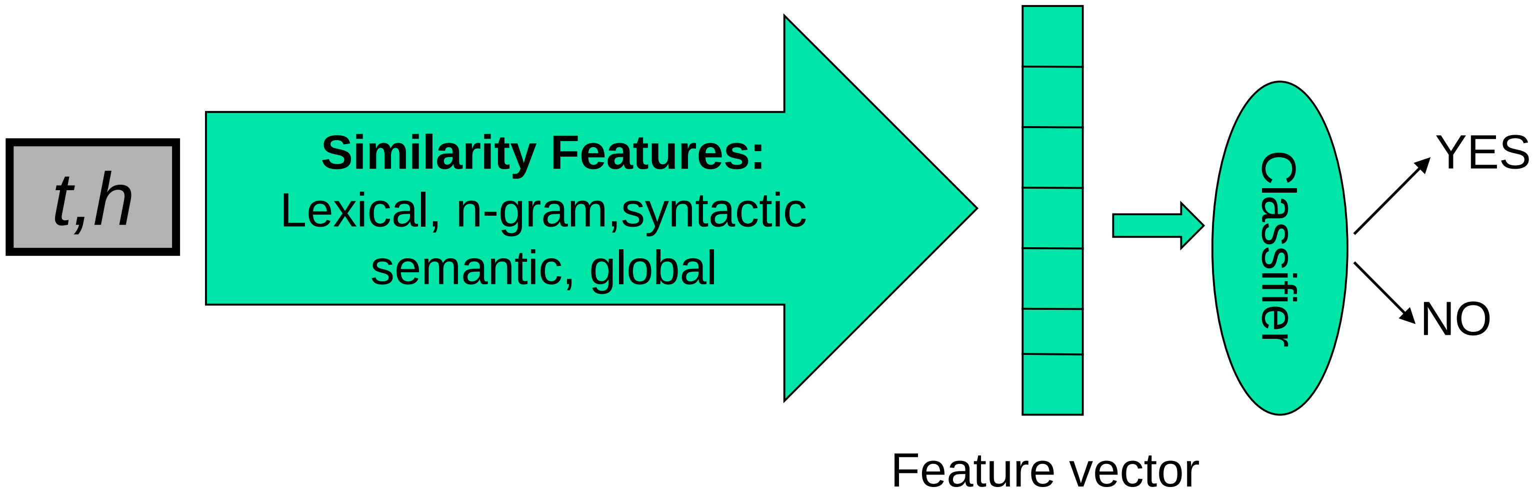
Participation and Impact

- Very successful challenges, world wide:
  - RTE-1 – 17 groups
  - RTE-2 – 23 groups
    - ~150 downloads
  - RTE-3 – 25 groups
    - Joint workshop at ACL-07
- High interest in the research community
  - Papers, conference sessions and areas, PhD's, influence on funded projects
  - Textual Entailment special issue at JNLE
  - ACL-07 tutorial

Methods and Approaches (RTE-2)

- Measure similarity match between *t* and *h* (*coverage* of *h* by *t*):
  - Lexical overlap (unigram, N-gram, subsequence)
  - Lexical substitution (WordNet, statistical)
  - Syntactic matching/transformations
  - Lexical-syntactic variations ("paraphrases")
  - Semantic role labeling and matching
  - Global similarity parameters (e.g. negation, modality)
- Cross-pair similarity
- Detect mismatch (for non-entailment)
- Interpretation to logic representation + logic inference

Dominant approach: Supervised Learning



- Features model similarity and mismatch
- Classifier determines relative weights of information sources
- Train on development set and auxiliary *t-h* corpora

RTE-2 Results

First Author (Group)	Accuracy	Average Precision
Hickl (LCC)	75.4%	80.8%
Tatu (LCC)	73.8%	71.3%
Zanzotto (Milan & Rome)	63.9%	64.4%
Adams (Dallas)	62.6%	62.8%
Bos (Rome & Leeds)	61.6%	66.9%
11 groups	58.1%-60.5%	
7 groups	52.9%-55.6%	

Average: 60%
Median: 59%