Logic and Theorem Proving
Current Activities

In Design

Logic and theorem proving are certainly used in real-world applications. It is extremely important to reason and deduct conclusions given information that does not directly prove such conclusions. For example, the team that built the Intel Itanium processor had to prove the reproduction of their logic and design using theorem proofs and logic. The conjunction between hardware and software systems requires great use of but not limited to: symbolic simulation, temporal logic model testing, and general theorem proving to formally verify the behaviors of a processor. I believe that logic and theorem proving are invaluable skills, apart from the purposes of this course. To solve problems, it is important to be able to reason through logical steps to ultimately come to a conclusion. This basic workflow is evident in all types of work including law, healthcare, and apparently, low level processor design.

https://www.cl.cam.ac.uk/~jrh13/slides/arw-04apr02/slides.pdf

Model checking

https://en.wikipedia.org/wiki/Model_checking

"Given a model of a system, exhaustively and automatically check whether this model meets a given specification. Typically, one has hardware or software systems in mind, whereas the specification contains safety requirements such as the absence of deadlocks and similar critical states that can cause the system to crash. Model checking is a technique for automatically verifying correctness properties of finite-state systems.

In order to solve such a problem algorithmically, both the model of the system and the specification are formulated in some precise mathematical language. To this end, the problem is formulated as a task in logic, namely to check whether a given structure satisfies a given logical formula. This general concept applies to many kinds of logics and suitable structures. A simple model checking problem is verifying whether a given formula in the propositional logic is satisfied by a given structure.

https://www.nsf.gov/news/news_summ.jsp?cntn_id=111102
Officials from the National Science Foundation's (NSF) Computer and Information Science and Engineering (CISE) directorate cheered this week's announcement that Edmund M. Clarke, E. Allen Emerson and Joseph Sifakis have won the 2007 A.M. Turing Award, frequently referred to as the 'Nobel Prize' of computing for their work on model checking. Clarke and Emerson have received funding and support from NSF throughout their distinguished careers.

https://cacm.acm.org/magazines/2009/11/48424-turing-lecture-model-checking-algorithmic-verification-and-debugging/fulltext

The website I found for the discussion is about automated theorem proving. ….

The link : http://tptp.cs.miami.edu/~tptp/OverviewOfATP.html

"What has Automated Theorem Proving been Really Useful for?

Many significant problems have been, and continue to be, solved using ATP. The fields where the most notable successes have been achieved are mathematics, software creation and verification, and hardware verification, and knowledge based systems.

The most exciting recent success in mathematics has been the settling of the Robbins problem by the ATP system EQP. In 1933 Herbert Robbins conjectured that a particular group of axioms form a basis for Boolean algebra, but neither he nor anyone else (until the solution by EQP) could prove this. The proof that confirms that Robbins' axioms are a basis for Boolean algebra was found October 10, 1996, after about 8 days of search by EQP, on an RS/6000 processor. This result was reported in the New York Times.

…. more in mathematics

Software creation is an economically important real world application of ATP. Although the use of ATP in software creation is in its infancy, there have already been some interesting results. The KIDS system developed at Kestrel Institute has been used to derive scheduling algorithms that have outperformed currently used algorithms. KIDS provides intuitive, high level operations for transformational development of programs from specifications. The Amphion project, sponsored by NASA, is used to determine appropriate subroutines to be combined to produce programs for satellite guidance. …

Software verification …

Hardware verification is the largest industrial application of ATP. …"

Knowledge representation can be time intensive

Theorem proving by using PVS

The theorem prover is a tool for logical reasoning, just as a calculator is an arithmetic tool. Theorem proof is not as mature or widely used as a calculator; using it requires considerable expertise. Theorem provers such as PVS can express any mathematics or computer science. This involves modeling, specification, definition of the constructs involved, and proof of their results interactively and automatically. **It can provide the highest level of correctness, but at a very high cost: the experts have put a lot of effort into it,** which is a safety-critical, safety-critical and mass-production system. The PVS specification language consists of a typed lambda calculus, "a functional programming language similar to Haskell or ML," but more expressive. Once we have defined a theory, we can prove any lemma and theorem it contains. Lemmas can be completed in any order; PVS tracks confirmed content. The user authenticates the PVS prover interactively by command.

The link https://www.cs.ru.nl/E.Poll/teaching/PVS/pvs_slides.pdf

Machine Learning to guide a theorem prover

Alpha Go and Theorem Proving

https://deepmind.com/research/alphago/
https://arxiv.org/pdf/1701.06972.pdf

AlphaGo is a program released by DeepMind that plays the game of Go. Recently, AlphaGo was able to beat one of the world's top Go players, using many unconvential moves that went against classical Go strategies. The application uses advanced tree search and deep neural networks to plan its moves. The program, given a set of axioms, needs to figure out how to make the 'best' move possible according to what it knows. The program obviously combines many different topics to produce its results, but it's using theorem proving to calculate the best end state and to figure out how to reach that end state.

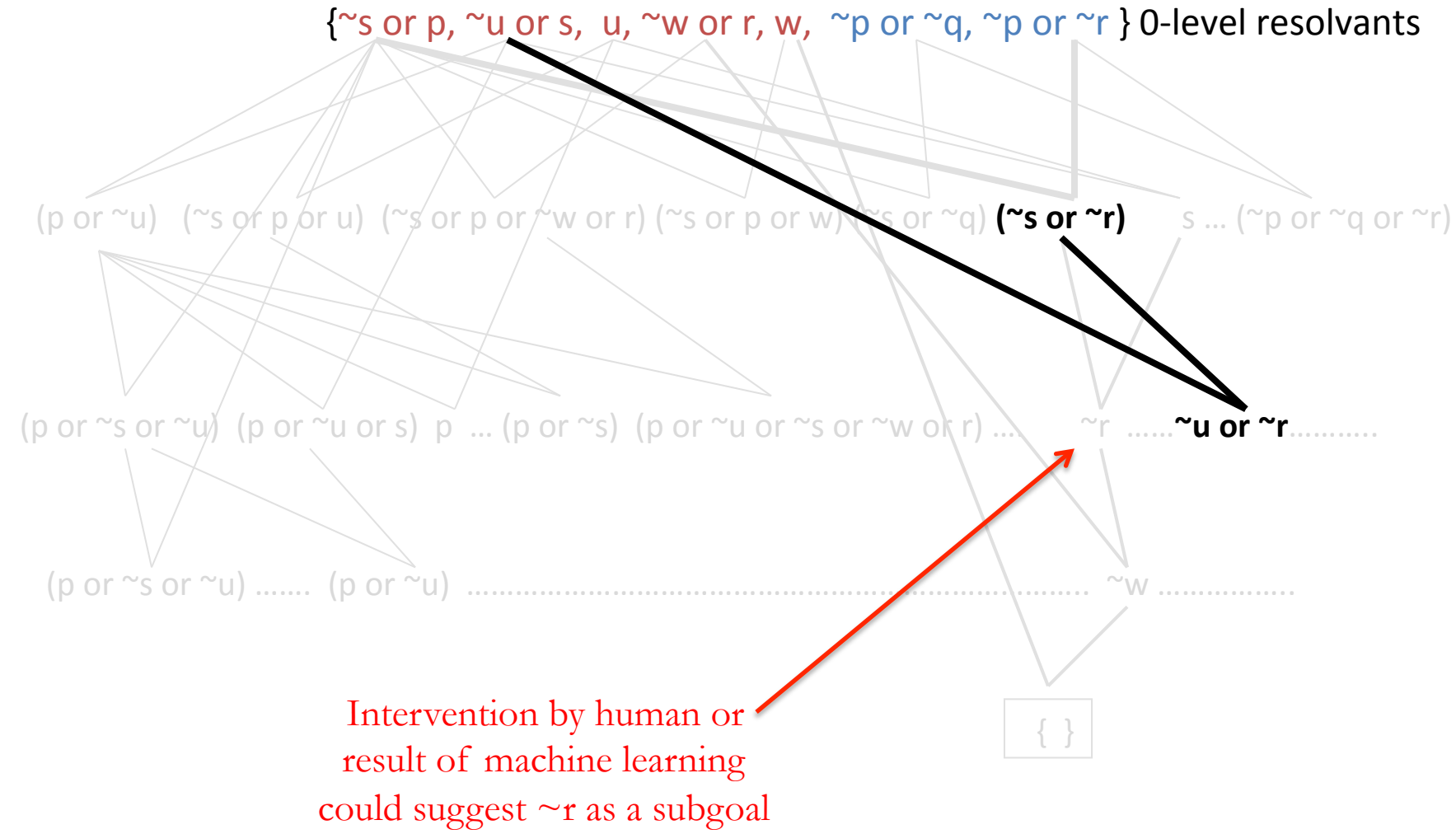
AITP Conference

The 2nd Conference on Artificial Intelligence and Theorem Proving (AITP) was held in in Austria. The purpose of the conference was to provide a forum for discussion that allowed individuals to present ideas regarding combinations of AI and reasoning methods over "large mathematical and scientific corpora." Topics included **big-data methods in theorem proving, collaboration between automated and interactive theorem proving, and combinations of learning-based and reasoning based methods.** I think it is interesting to note that the conference presented large-scale combination of mathematical theorem proving as an inevitable future for the field of AI, which seems to indicate the significance of logic and theorem proving in the field of AI.

Source:

"2nd Conference on Artificial Intelligence and Theorem Proving." AITP. http://aitp-conference.org/2017/

Interactive theorem proving is also an important focus in our textbook in the guise of asking questions of the theorem prover. A human expert may be able to ask questions of intermediate granularity (not too fine grained and not too course grained) in order to guide the theorem-prover in a way that mitigates a computational explosion on derived intermediate theorems, but


{~s or p, ~u or s, u, ~w or r, w, ~p or ~q, ~p or ~r } 0-level resolvants
(p or ~u) (~s or p or u) (~s or p or ~w or r) (~s or p or w) (~s or ~q) (~s or ~r) s ... (~p or ~q or ~r)
(p or ~s or ~u) (p or ~u or s) p ... (p or ~s) (p or ~u or ~s or ~w or r) ... ~r .....~u or ~r.....
(p or ~s or ~u) ..... (p or ~u) ..... ~w .....
{ }
Intervention by human or
result of machine learning
could suggest ~r as a subgoal

As of 2016, the Artificial Intelligence and Theorem Proving conference has showcased emerging academic (and some private) research into the role of Artificial Intelligence in automated Theorem Proving problems. Among the focus of these researchers this year was in improving the ability of the AI to better understand the problem domain; to this end Takuya Matsuzaki and Noriko Arai presented their research in "Machine comprehension of math problem text."

Another area of focus was theorem proving with tactics. **In summary, given a set of proofs, certain tactics to make progress will be more successful than others; learning to optimize the process to deliver more efficient results formed the basis of the work by Thibault Gauthier and Ramana Kumar worked to develop AI capable of using theorem proving technology to optimize tactics in a known problem space.** It seems plausible to me that this could be applied in the future to problems beyond explicitly specified proof end-states to more generalized process optimization. Any proposed **industry implementation would require extensive interfacing with domain experts, but the possibility to augment human process-design is apparent, as these processes also have specified options to improve performance**, and a specific end state (minimizing time, maximizing output). An optimization problem can be modeled as a proof that the proposed process represents the upper/lower bound of productivity. In practice, such a bound cannot be reached, necessitating the inclusion of a tolerance for practical application.

References:

3rd Conference on Artificial Intelligence and Theorem Proving, 3/25-3/30 2018, http://aitp-conference.org/2018/

Gauthier, Thibault, et al. "TacticToe: Learning to Reason with HOL4 Tactics." 2018, doi:10.29007/ntlb.

Machine comprehension of math problem text,
Takuya Matsuzaki and Noriko H. Arai
3rd Conference on Artificial Intelligence and Theorem Proving, Aussois, France 27 Mar 2018

MathWeb is a theorem proving application that combines current softwares into a single distributed, modularized too. **The basis of the system is a society of agents that perform mathematical tasks, with a central agent directing actions and information, and this allows for program distribution.** Since each agent encapsulates different services it is extremely modular and is easy to create new agents based on new software applications, which is the main component of "plug and play" architecture. The article focuses most on the system architecture than the system itself, which I think is an interesting choice. **The article is unique in that it highlights distributed and modular system architecture and delves into each design choice that went into making clean code**, which I think is a focus more projects should have, although I would have liked to see more about how the theorem proving was implemented by each agent or at least examine one representative agent.

Franke A., Kohlhase M. (1999) System Description: MathWeb, an Agent-Based Communication Layer for Distributed Automated Theorem Proving. In: Automated Deduction — CADE-16. CADE 1999. Lecture Notes in Computer Science, vol 1632. Springer, Berlin, Heidelberg

Is this related to Domain Splitting?

The ideas about using machine learning to guide theorem proving are example of a more general situation on learning search control

Aside: another example of learning search control arises when learning macro-operators

Consider a domain in which we have STRIPS operators
PUC, DC, PUM, DM, MC-OFF, MC-CS, MC-MR, MC-LAB, MCC-OFF, MCC-CS, MCC-MR, MCC-LAB

Lets say the average branching factor of the search is 4 (e.g., from a state in the planner search, there are approximately 4 neighbors)

When we learn macros like
PUC; MC-CS; DC and
PUC; MC-CS; DC; MC-OFF; MC-MR; PUM, MCC-MR; MCC-LAB; MCC-OFF; DM

We DECREASE the effective depth of search when a macro is applicable in virtual space, but we INCREASE the effective branching factor of search and

- this increases search cost, unless
- we use machine learning to decide under what conditions (macro-)operators are likely to be helpful (we must address the "utility problem" – what macros are helpful and when)

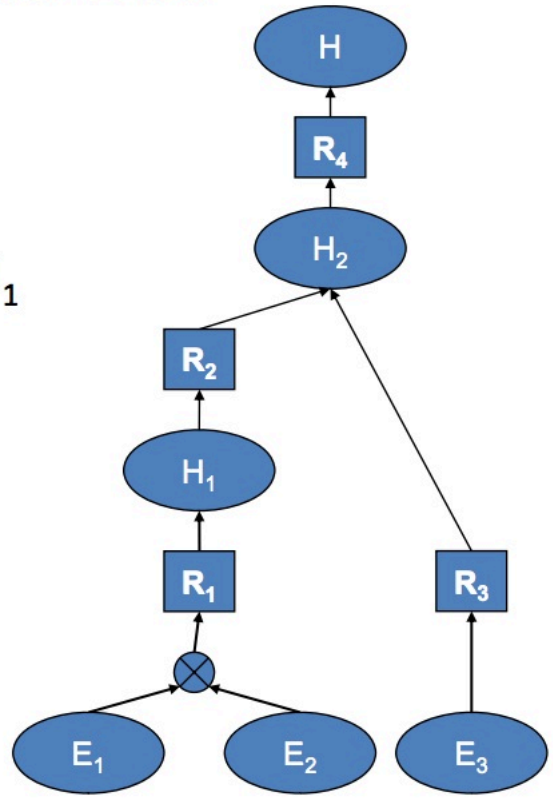
The same machinery for proving a theorem can be adapted to showing what is possible. This is most easily seen in diagnosis

sore-throat → flu (0.6)

car-won't-start → battery-dead (0.7)

Inference Tree

- Rules
  - R_1: IF E_1 AND E_2 THEN H_1
  - R_2: IF H_1 THEN H_2
  - R_3: IF E_3 THEN H_2
  - R_4: IF H_2 THEN H



http://www.dia.fi.upm.es/~mgremesal/MIR/slides/03%20-%20MIR%20-%20%20MYCIN%20(PF).pdf

Note Dr. Randy Miller and the INTERNIST-I computer-assisted medical diagnosis project

The same machinery for proving a theorem can be adapted to showing what is possible. This is most easily seen in diagnosis

RULE037

IF: 1) *The identity of the organism is not known with certainty*, and
2) The stain of the organism is gramneg, and
3) The morphology of the organism is rod, and
4) The aerobicity of the organism is aerobic

THEN: There is strongly suggestive evidence (.8) that the class of the organism is enterobacteriaceae

RULE145

IF: 1) The therapy under consideration is one of: cephalothin clindamycin erythromycin lincomycin vancomycin, and
2) Meningitis is an infectious disease diagnosis *for the patient*

THEN: It is definite (1) the therapy under consideration is not a potential therapy for use against the organism

RULE060

IF: The *identity of the organism* is bacteroides

THEN: I recommend therapy chosen from among the following drugs:

1 - clindamycin	(.99)
2 - chloramphenicol	(.99)
3 - erythromycin	(.57)
4 - tetracycline	(.28)
5 - carbenicillin	(.27)

Rules from the MYCIN system

http://www.aaaipress.org/Classic/Buchanan/Buchanan07.pdf

Other applications

Logic Theorist
Mitigation of adverse interactions in pairs of clinical practice guidelines using constraint logic programming
Online Logic Course (https://www.mooc-list.com/course/logic-language-and-information-2-coursera)
Foundation Series
Natural Language Processing
Interactive theorem proving conference (https://itp2018.inria.fr/)
Automated Logic Checking at Compile Time for Programming Languages