Automated Reasoning: the big picture

Maria Paola Bonacina

Dipartimento di Informatica
Università degli Studi di Verona
Verona, Italy, EU

Talk given at the Dept. of Computer Science and Engineering
Chalmers University of Technology and Gothenburg University
As Opponent of Magnus Björk's PhD thesis defense
Gothenburg, Sweden, EU

11 May 2006

The field of automated reasoning: a brief overview

Theorem proving as a search problem

A taxonomy of theorem-proving strategies

A central problem in automated reasoning

S: set of *assumptions*
properties of the object of study
(e.g., system, circuit, program, data type, communication protocol, mathematical structure)

φ: *conjecture*
a property to be verified

Problem: does φ *follow* from S?

$$S \models^? \varphi$$

Automated reasoning and knowledge representation

Knowledge representation:
finding formalisms for S and φ to represent desired aspects of the analyzed systems

Automated reasoning:
studying and implementing *reasoning techniques* to solve the entailment problem ($S \models^? \varphi$)

Automated reasoning in first order logic

Representation formalism: first order logic (FOL)

Motivation: FOL provers applied successfully to, e.g.,

- *software and hardware verification*, e.g.,
  - cryptographic protocols
  - message-passing systems
  - software specifications
  - theorem proving support to model checking
- *proving non-trivial mathematical theorems* in, e.g.,
  - Boolean algebras
  - theories of rings, groups and quasigroups
  - many-valued logic

Automated reasoning: building proofs or models

$$S \models^{?} \varphi$$

- **Theorem proving:**
  finding a *proof* of φ from S and answer affirmatively
- **Model building:**
  finding a *model* of $S \cup \{\neg\varphi\}$, that is a *counter-example* for $S \models \varphi$, and answer negatively

Theorem proving: deduction or induction

$S \models \varphi$:
φ is true in all models (systems, worlds ...) where S is true

- **Deductive theorem proving:**

$$S \models \varphi$$

- **Inductive theorem proving:**

$$S \models \varphi\sigma$$

for all ground substitutions σ

Automated reasoning problems are very hard

In first order logic

- **Deductive theorem proving** is only semi-decidable
- **Inductive theorem proving** is not even semi-decidable
- **Model building** is not even semi-decidable

Automatic and interactive theorem proving

- **Automatic theorem proving:**
  the machine alone is expected to find a proof
- **Interactive theorem proving:**
  a proof is born out of the interaction between human and machine

Automatic deductive theorem proving

- **Automatic theorem proving:**
  deductive theorem proving
- **Interactive theorem proving:**
  induction, model generation and reasoning in higher-order logics

Refutational theorem proving

- *Direct proof:*
  deriving φ from S without making use of φ itself
- *Proof by way of contradiction or by refutation:*
  showing that $S \cup \{\neg\varphi\}$ generates a contradiction ($\perp$),
  $S \cup \{\neg\varphi\}$ is *inconsistent*, hence $S \models \varphi$

Too difficult to find a proof ignoring the conjecture:
theorem-proving methods work *refutationally*.

Decidable instances of reasoning problems

Decidable instances of reasoning problems do exist

Decidability may stem from imposing restrictions on

1. the logic
2. the form of admissible formulae
3. the theory presented by the assumptions

Examples of decidable instances

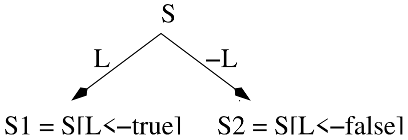
1. *propositional logic*: the SAT problem
2. *Bernays-Schönfinkel class*:

   $$\exists x_1, \ldots x_n . \forall y_1, \ldots y_m . P[x_1, \ldots x_n, y_1, \ldots y_m]$$

   where P is quantifier-free and function-free
3. *Presburger arithmetic* or theories of data structures, such as *lists* or *arrays*

SAT: Davis-Putnam-Logemann-Loveland procedure

- **Case analysis or splitting + unit propagation:**



- **Unit clause rule:** if L is a clause, only one branch
- **Pure literal rule:** if L is pure (only one sign), only one branch
- **Control:** depth-first search (DFS) with backtracking + refinements

SAT: Boolean Ring simplification

Let $+$ be exclusive *or* and juxtaposition be *and*:

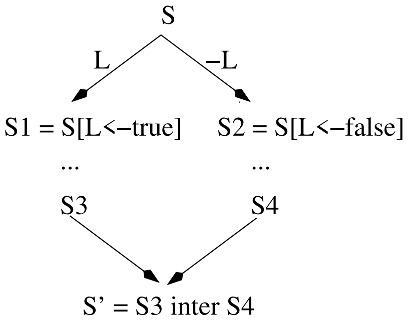
$$\begin{array}{ccc}
\mathbf{xx = x} & \mathbf{x0 = 0} & \mathbf{x1 = 1} \\
\mathbf{x + x = 0} & \mathbf{x + 0 = x} & -x = x \\
xy = yx & (xy)z = x(yz) & \\
x + y = y + x & (x + y) + z = x + (y + z) & \mathbf{x(y + z) = xy + xz}
\end{array}$$

$x \vee y$ is $xy + x + y$ and $\neg x$ is $x + 1$

- **Simplification** by equations in bold face as *rewrite rules*
- **Unique normal form**: 0, 1 or a Boolean polynomial
- **Distributivity:** exponential growth of the normal form
- **A solution:** DPLL + BR representation + BR simplification

SAT: Stålmarck's method

- **Same framework as DPLL (sort of)**
- **Dilemma rule:**



- **Control:** DFS with **iterative deepening** (DFID) to control how deep to go in the dilemma's branches

Back from SAT to FOL theorem proving

Semi-decidability:

No procedure is guaranteed to halt and

- return a positive answer and a proof whenever $S \cup \{\neg\varphi\}$ is inconsistent
- return a negative answer and a model whenever $S \cup \{\neg\varphi\}$ is consistent

The best one can have is a *semi-decision procedure*

Semi-decision procedures

A *semi-decision procedure* is guaranteed to halt and return a positive answer and a proof whenever $S \cup \{\neg\varphi\}$ is inconsistent.

However, if $S \cup \{\neg\varphi\}$ is consistent, the procedure is not guaranteed to halt.

Search for proofs

Intuition of the source of semi-decidability:

- Proofs are *finite*, if they exist, but
- There is an *infinite search space* of consequences where to look for a contradiction

A machine can explore only a *finite* part of this *infinite* space

Challenge: to find a proof using as little resources as possible

Theorem-proving strategies

- *Inference system:*
  set of *inference rules*
  defines the *search space* of all possible inferences
- *Search plan:*
  controls the application of the inference rules
  guides the search for a proof

Inference system + search plan = *theorem-proving strategy*

Since we are looking for a proof:
Proof system + search plan = *proof procedure*

From non-determinism to determinism

- *Inference system:*
  non-deterministic set of *inference rules*
- *Search plan:*
  determines the unique *derivation*, e.g.,

$$S_0 \vdash S_1 \vdash \ldots S_i \vdash S_{i+1} \vdash \ldots$$

  that the strategy computes from $S_0 = S \cup \{\neg\varphi\}$

A TP strategy or proof procedure is *deterministic*

S_i: *state*, e.g.: a set of clauses; a set of clauses and a tableau

Soundness and adequacy

- **Soundness:** if $S_i \vdash S_{i+1}$ then $S_i \models S_{i+1}$
- **Adequacy:** if $S_i \vdash S_{i+1}$ then $S_{i+1} \models S_i$

Adequacy was also called *monotonicity*:

$S_i \vdash S_{i+1}$ implies $Th(S_i) \subseteq Th(S_{i+1})$

where $Th(S) = \{\psi \mid S \models \psi\}$

Refutational completeness and fairness

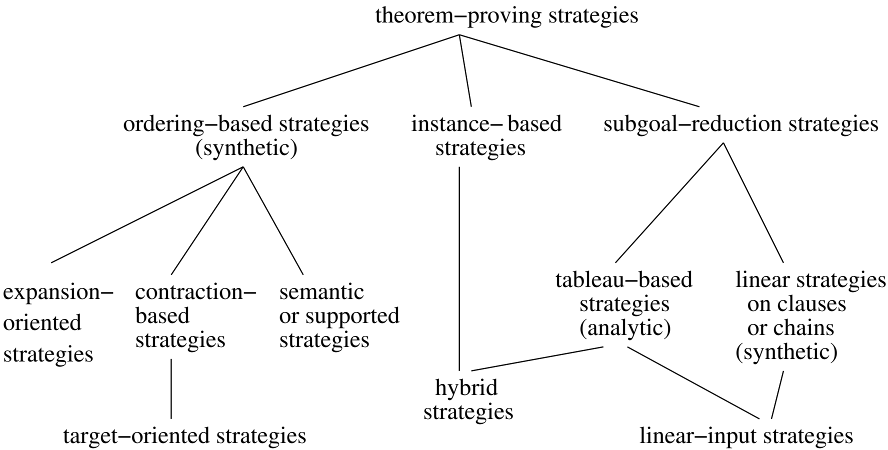
- **Refutational completeness:**
  if $S_0 = S \cup \{\neg\varphi\}$ is inconsistent,
  inference system generates at least a derivation
  $S_0 \vdash S_1 \vdash \ldots S_i \vdash S_{i+1} \vdash \ldots$
  such that S_k contains $\perp$ for some k
- **Fairness:**
  search plan considers *eventually* all inferences that may be necessary to generate such an S_k
- **Uniform fairness:**
  search plan considers *eventually* all *irredundant* expansion inferences
- **Formal definitions:** e.g., with well-founded *proof orderings*

Refutational completeness

If the inference system (proof system) is *refutationally complete* and the search plan is *fair*,
then the strategy (proof procedure) is *refutationally complete*:

if $S_0 = S \cup \{\neg\varphi\}$ is inconsistent,
the unique *derivation*
$S_0 \vdash S_1 \vdash \ldots S_i \vdash S_{i+1} \vdash \ldots$
computed by the strategy is such that S_k contains $\perp$ for some k

A taxonomy of theorem-proving strategies



Ordering-based strategies I

- **Expansion inference rule:**

$$\frac{A_1 \quad \dots \quad A_n}{B_1 \quad \dots \quad B_m}$$

where $m > 1$
e.g., *resolution* and *paramodulation*

- **Contraction inference rule:**

$$\frac{A_1 \quad \dots \quad A_n}{\overline{B_1 \quad \dots \quad B_m}}$$

where $m \geq 0$
e.g., *subsumption* and *equational simplification*

Ordering-based strategies II

- **Expansion inferences:**

$$\frac{S_i}{S_{i+1}} \ \ S_i \subset S_{i+1}$$

- **Contraction inferences:**

$$\frac{S_i}{S_{i+1}} \ \ S_i \not\subseteq S_{i+1} \quad S_{i+1} \prec S_i$$

  where $\prec$ is a well-founded ordering

- **Database of clauses:** *indexing* techniques

Subgoal-reduction based strategies I

- *Model elimination (ME)*
- *Linear resolution*
- *Matings*
- *Connections or matrices*

All eventually understood in the context of
clausal normalform tableaux
e.g., *ME-tableaux*

Subgoal-reduction based strategies II

- **Free-variable tableaux**
- **Clausal normalform tableaux:**
  - *Extension:* extend branch with fresh copy of clause
  - *Closure:* close branch with unifiable complementary literals + apply mgu
  - *(Strong) link condition:* extend only if (adjacent) complementary literals unify
- **Rigid variables**

Ordering-based and subgoal-reduction strategies I

	Ordering-based	Subgoal-reduction
Data	set of objects	one goal-object at a time
Proof attempts built	many implicitly	one at a time
Backtracking	no	yes
Contraction	yes	no

Ordering-based and subgoal-reduction strategies II

	Ordering-based	Subgoal-reduction
Visited search space	all generated clauses	all tried tableaux
Active search space	all kept clauses	current tableau
Generated proof	ancestor-graph of $\Box$	closed tableau

Instance-based strategies

- Forerunner: *Gilmore's multiplication method* (1960)
- Recent methods:
  - Generate ground instances of clauses in set to be refuted (e.g., by *hyperlinking*)
  - Apply a SAT solver and iterate
- More recent methods:
  - SAT solver as model generator
  - Generate ground instances to eliminate models

Hybrid strategies

Combine tableaux and instance generation, e.g.:

- Give up on instantiating rigid variables in the tableau
- Backtracking no longer needed
- Add instance generation, e.g., by *hyperlinking*

Intuitively, the information lost by not instantiating the tableau is generated as instances of clauses in the given set.