Consequence-Finding

CS 157 Computational Logic
Lecture 18
November 19, 2009
Autumn 2009
Eric Kao

Proof-finding vs. Consequence-finding

Proof-finding:

Given premises and a goal, determine whether the premises entail the goal.

Consequence-finding:

Given premises, find the sentences which the premises entail.

Uses of Consequence-finding

Finding novel theorems

Knowledge Compilation

Predicting program behavior (specification generation)

Abduction

- Residues
- Diagnostics
- Requirements generation
- ...

...

Abduction - Generating hypotheses

Given a set of premises Σ and a goal φ,
for which sentences δ is it the case that
$\Sigma \cup \{\delta\} \models \varphi$?

Equivalently,
for which sentences δ is it the case that
$\Sigma \cup \{\sim\varphi\} \models \sim\delta$ and

We saw this in generating residues

Consequence-finding

Given a set of sentence:

{ p | q,
~q }

find all consequences.

How many consequences are there?

Ans: **Infinitely many**

How many non-tautological consequences are there?

Ans: **Infinitely many**

Consequence-finding: definition

Given a set of sentences Σ, normalized in clausal form,
find a set of sentences Π such that:

- $\Sigma \models \Pi$
- for any clause c such that $\Sigma \models c$ and c is not a tautology,
  - there exists a clause d in Π such that
    - d subsumes c.

We denote such a set Π as Con[Σ]

Example:

For Σ = { p | q, ~q }, which sets qualify as Con[Σ]?

{ p | q, ~q **}**

{ p | q, ~q, p }

{ p | q, ~q, p, p | ~q}

{ p | q, ~q, p, q**}**

{ p, ~q }

Example:

For Σ = { p(X) | q(X), ~q(a) }, which sets qualify as Con[Σ]?

{ p(X) | q(X), ~q(a) }

{ p(X) | q(X), p(a), ~q(a) }

{ p(a), ~q(a) }

{ p(X) | q(X), p(X), ~q(a) }

Finiteness

Given a finite set of clauses Σ, does there always a finite Con[Σ]?

In propositional logic?

yes

Finiteness

In first-order logic?

not always.

Σ = { ~p(X1,X2) | ~p(X2,X3) | p(X1,X3) }
(rule form: p(X1,X2) & p(X2,X3) => p(X1,X3))

Con[Σ] = {
~p(X1,X2) | ~p(X2,X3) | p(X1,X3)
~p(X1,X2) | ~p(X2,X3) | ~p(X3,X4) | p(X1,X4)
~p(X1,X2) | ~p(X2,X3) | ~p(X3,X4) | ~p(X4,X5) | p(X1,X5)
... ... }

Finiteness

Given an input Σ, if there is a finite set Con[Σ], we expect to find that finite set as output.

If Con[Σ] cannot be finite, we may want to enumerate it.

Method for consequence finding

Truth table?

Generate and test
"min terms"

Karnaugh maps

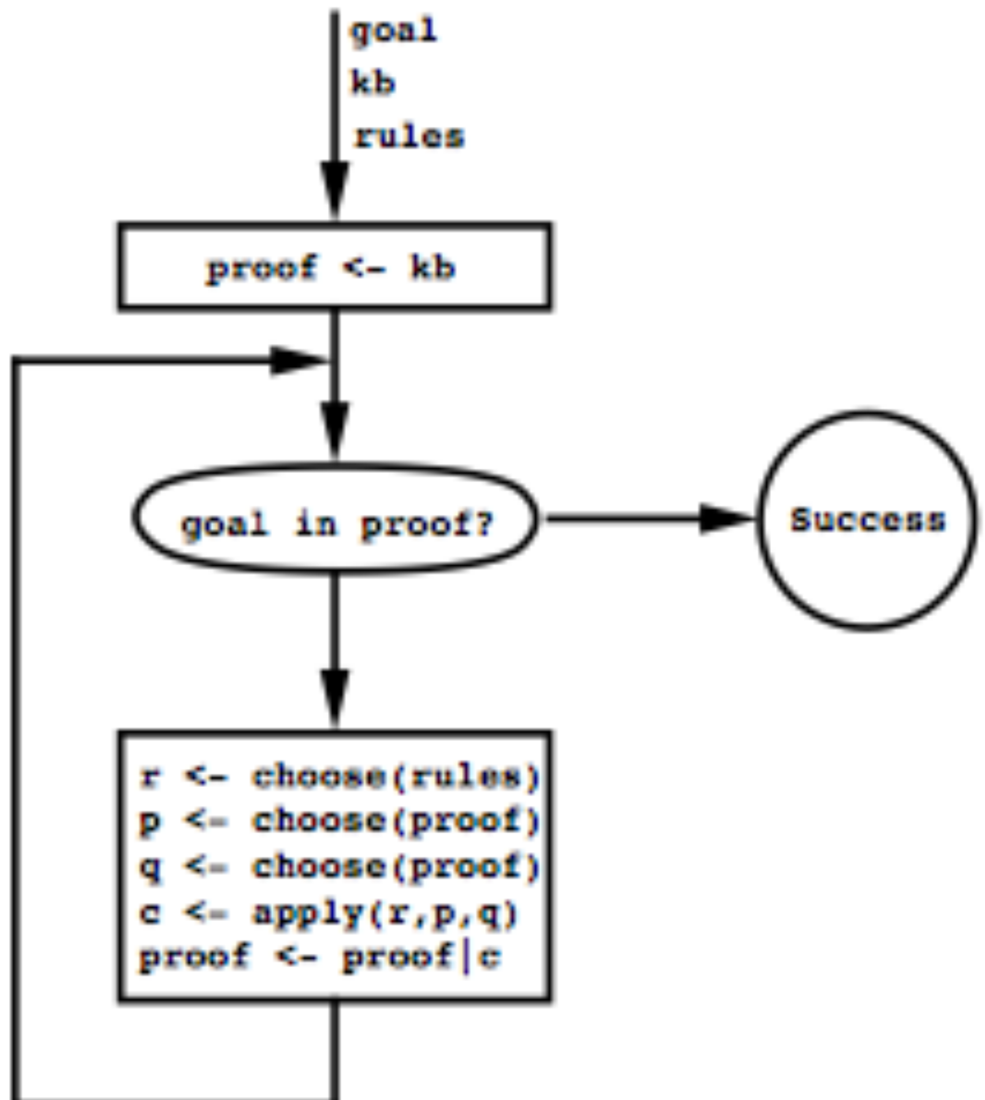
Does not work for FOL in general

Method for consequence finding

Formal proofs?

Hard to mechanize

termination on
finite Con[Σ]?



Method for consequence finding

Resolutions method?

Generate and test

How to systematically generate FOL clauses?

Consequence-finding using resolution

Σ = { p | q, ~q }
1. { p, q }
2. { ~q }
3. { p }   Res 1,2

We write $\Sigma \mid\!-^{*}_{Res}\ \varphi$ to denote that φ can be derived from Σ in a finite number of resolution steps.

And we write Res[Σ] to denote $\{ \varphi : \Sigma \mid\!-^{*}_{Res}\ \varphi \}$.
Resolution closure

Consequence-finding using resolution

So what if we take
$Res[\Sigma]$ as $Con[\Sigma]$?

The resolution rule of inference is sound.

So all derived clauses are sound consequences

But is it complete?

Generative Incompleteness

Resolution is refutation complete:

If a set of sentences Γ is unsatisfiable, $\Gamma \mid\!-^*_{Res} \{\ \}$

Resolution principle is not generatively complete:

If $\Gamma \models \varphi$, it is not necessarily the case that $\Gamma \mid\!-^*_{Res} \varphi$

Completeness of Resolution for CF [Lee 1967]

unrestricted resolution is complete for consequence-finding:
That is, Res[Σ] includes every formulate needed for Con[Σ].

Given any set of clauses Σ,
Res[Σ] satisfies all the requirements to be Con[Σ].

Notice that the resolution principle itself, used for proof finding by refutation, was introduced earlier [Robinson, 1965]

Resolution Strategies?

In proof-finding, unrestricted resolution involves many redundant steps which may be optimized away using various resolution strategies.

Do the same strategies work for consequence-finding?

Completeness of elimination strategies

Identical clause elimination
OK

Tautology elimination
OK

Subsumed clause elimination
OK

Pure literal elimination
NO
{p,~q}
{q,r}

Completeness of restriction strategies

Set of Support
NO

Empty set of support:
1. {p,q} Background
2. {~p}   Background

Missing {q}

Completeness of restriction strategies

Linear resolution (unordered)
OK [Minicozzi and Reiter, 1972]

semi-ordered resolution

NO [Minicozzi and Reiter, 1972]
S = {
<p,q>
<p,~q>
}

Missing: {p}

Skipping

Model elimination is not complete for consequence-finding =(

But variations based on skipping have been shown to be complete for consequence-finding:

1. <*p, q > top clause
2. <p, *q > skip p
3. <p,*p> resolve against <p,~q> on ~q

SOL-resolution [Inoue 1991]
SFK-resolution [del Val 1999]

Conclusions for consequence-finding

Many applications require consequence-finding rather than simply proof-finding.

Some resolution strategies are sound and complete for consequence-finding.

Unfortunately, semi-ordered resolution is not complete for CF.

Some variations complete for CF at the cost of increased search space.

Research questions

Targeted consequences:

Given a class of consequences Q, and set of clauses Σ, find Con [Σ] $\cap$ Q.

Incremental computation:

Given Con[Σ] and a new clause c, find Con[Σ U {c}].

Given Con[Σ] and a clause d, find Con[Σ - {d}].

References

[Inoue 1991] K. Inoue. Consequence-finding based on ordered linear resolution. IJCAI 1991. http://dli.iiit.ac.in/ijcai/IJCAI-91-VOL1/PDF/026.pdf

[del Val 1999] A. del Val. A new method for consequence finding and compilation in restricted languages. AAAI 1999. http://www.aaai.org/Papers/AAAI/1999/AAAI99-038.pdf

[Lee 1967] Lee, C. 1967 *A Completeness Theorem and a Computer Program for Finding Theorems Derivable from Given Axioms*. Doctoral Thesis. UMI Order Number: AAI6810359., University of California, Berkeley.

[Minicozzi and Reiter, 1972] Eliana Minicozzi and Raymond Reiter. A note on linear resolution strategies in consequence-finding. *Artificial Intelligence*, 3: 175-180, 1972. [link]