3.3 Ground Resolution

Freitag, 24. April 2015 08:30

Drawbacks of Gilmore's Algorithm:

- unclear with which ground terms one should instantiate variables
- to check whether a ground formula is satisfiable:
  try out all possible assignments of $\underbrace{\text{atomic ground formulas}}_{\text{correspond to propositional variables}}$ to {TRUE, FALSE}

Why is it not enough to check unsatisfiability of
ψ_1 or ψ_2 or ψ_3 or ... ?

Ex: p(0).
p(s(X)) :- p(X).

$\left(\begin{array}{c} 0 \hat{=} 0 \\ s(0) \hat{=} 1 \\ s(s(0)) \hat{=} 2 \\ \vdots \end{array} \right)$

?- p(s(s(0))).

φ_1: $p(0)$.
φ_2: $\forall X \; p(X) \rightarrow p(s(X))$
φ : $p(s(s(0)))$

We have to check unsatisfiability of

ψ: $\forall X \; p(0) \wedge (p(X) \rightarrow p(s(X))) \wedge \neg p(s(s(0)))$

$[X/0]$: ψ_1: $p(0) \wedge (p(0) \rightarrow p(s(0))) \wedge \neg p(s(s(0)))$

Satisfiable: $p(0)$: TRUE, $p(s(0))$: TRUE, $p(s(s(0)))$: FALSE

$[X/s(0)]$: ψ_2 : $p(0) \wedge (p(s(0)) \rightarrow p(s(s(0)))) \wedge \neg p(s(s(0)))$

satisfiable: $p(0)$: TRUE, $p(s(0))$: FALSE, $p(s(s(0)))$: FALSE

$[X/s(s(0))]$: ψ_3: ...

also satisfiable

$\Rightarrow$ all ψ_i on their own are satisfiable

but $\psi_1 \wedge \psi_2$ is unsatisfiable

Reason: the same rule has to be applied several times with different instantiations.

<u>Goal</u>: Improve the 2nd drawback of Gilmore's algorithm (i.e., check unsatisfiability of ground formulas)

<u>Solution</u>: Resolution (today: ground resolution)

3.3. Ground Resolution

Input: Formula $\forall X_1, \dots, X_n \; \psi$ in Skolem NF

Goal: check unsatisfiability

First step: transform quantifier-free formula ψ to <u>Conjunctive normal form</u> (CNF)

<u>Def 3.3.1</u> (CNF)

A formula ψ is in <u>CNF</u> iff it is quantifier-free and it has the following form:

$$(L_{1,1} \vee \dots \vee L_{1,n_1}) \wedge \dots \wedge (L_{m,1} \vee \dots \vee L_{\dots})$$

$$(L_{1,1} \vee \ldots \vee L_{1,n_1}) \wedge \ldots \wedge (L_{m,1} \vee \ldots \vee L_{m,n_m})$$

Here, $L_{i,j}$ are <u>literals</u>, i.e., they are atomic or negated atomic formulas (i.e., they have the form $p(t_1, \ldots, t_n)$ or $\neg p(t_1, \ldots, t_n)$).

For every literal L we define its <u>negation</u> $\overline{L}$ as follows:

$$\overline{L} = \begin{cases} \neg A, & \text{if } L = A \in At(\Sigma, \Delta, \mathcal{V}) \\ A, & \text{if } L = \neg A \text{ for } A \in At(\Sigma, \Delta, \mathcal{V}) \end{cases}$$

A set of literals is called a <u>clause</u>.

Every formula φ in CNF corresponds to the following <u>clause set</u>:

$$\mathcal{K}(\varphi) = \{ \underbrace{\{L_{1,1}, \ldots, L_{1,n_1}\}}_{\text{clause}}, \ldots, \underbrace{\{L_{m,1}, \ldots, L_{m,n_m}\}}_{\text{clause}} \}$$

So a clause stands for the universally quantified disjunction of its literals and a clause set corresponds to the conjunction of its clauses.

The <u>empty clause</u> is denoted $\square$ and it is unsatisfiable by definition.

<u>Thm 332</u> (Transformation to CNF)

For every quantifier-free formula φ one can automatically construct an equivalent formula φ' in CNF.

<u>Proof</u>: First, replace sub-formulas $\varphi_1 \leftrightarrow \varphi_2$ by

$(\psi_1 \to \psi_2) \wedge (\psi_2 \to \psi_1)$.

Then replace sub-formulas $\psi_1 \to \psi_2$ by $\neg \psi_1 \vee \psi_2$.

Then apply the following algorithm $CNF(\psi)$:

- if ψ is atomic, then return ψ
- if $\psi = \psi_1 \wedge \psi_2$, then $CNF(\psi_1) \wedge CNF(\psi_2)$
- if $\psi = \psi_1 \vee \psi_2$, then compute

$$CNF(\psi_1) = \bigwedge_{i \in \{1,\dots,m_1\}} \psi_i'$$

$$CNF(\psi_2) = \bigwedge_{j \in \{1,\dots,m_2\}} \psi_j''$$

Then return

$$\bigwedge_{\substack{i \in \{1,\dots,m_1\} \\ j \in \{1,\dots,m_2\}}} (\psi_i' \vee \psi_j'')$$

$(\psi_1' \wedge \psi_2') \vee$

$(\psi_1'' \wedge \psi_2'')$

is equivalent to

$(\psi_1' \vee \psi_1'') \wedge$

$(\psi_1' \vee \psi_2'') \wedge$

$(\psi_2' \vee \psi_1'') \wedge$

$(\psi_2' \vee \psi_2'')$

Distribution Law

- if $\psi = \neg \psi_1$, then compute

$$CNF(\psi_1) = \bigwedge_{i \in \{1,\dots,m\}} \left(\bigvee_{j \in \{1,\dots,n_i\}} L_{i,j} \right)$$

Applying De Morgan Laws results in

$$\bigvee_{i \in \{1,\dots,m\}} \left(\bigwedge_{j \in \{1,\dots,n_i\}} \overline{L_{i,j}} \right)$$

Applying the distribution law yields the following formula that is returned:

$$\bigwedge_{j_1 \in \{1,\dots,n_1\}} \left(\overline{L_{1,j_1}} \vee \dots \vee \overline{L_{m,j_m}} \right)$$

$j_1 \in \{1,\dots,n_1\}$

$\vdots$

$j_m \in \{1,\dots,n_m\}$

Ex. 3.3.3 φ is the following formula with $p, q, r \in \Delta_0$:

$$\neg(\neg p \wedge (\neg q \vee r))$$

De Morgan laws yield:

$$p \vee (q \wedge \neg r)$$

Distribution law results in:

$$(p \vee q) \wedge (p \vee \neg r) \quad \leftarrow \text{in CNF}$$

Remaining goal: Check unsatisfiability of a ground formula in CNF, i.e., of a set of ground clauses.

Def 3.3.4 (Propositional Resolution)

Let K_1, K_2 be ground clauses. Then the clause R is a resolvent of K_1 and K_2 iff there is a literal $L \in K_1$ with $\overline{L} \in K_2$ and $R = (K_1 \setminus \{L\}) \cup (K_2 \setminus \{\overline{L}\})$.

For a clause set $\mathcal{K}$ we define

$\mathrm{Res}(\mathcal{K}) = \mathcal{K} \cup \{R \mid R$ is resolvent of two clauses from $\mathcal{K}\}$.

Moreover, let

$$Res^0(\mathcal{K}) = \mathcal{K}$$

$$Res^{n+1}(\mathcal{K}) = Res(Res^n(\mathcal{K})) \text{ for all } n \geq 0.$$

So the set of all clauses that can be deduced by resolution is

$$Res^*(\mathcal{K}) = \bigcup_{n \geq 0} Res^n(\mathcal{K})$$

Obviously, we have $\Box \in Res^*(\mathcal{K})$ iff there is a sequence of clauses $K_1, \dots, K_m$ such that the following holds for all $1 \leq i \leq m$:

- $K_i \in \mathcal{K}$ or
- K_i is a resolvent of K_j and K_k for $j, k < i$.

To display resolution proofs, we often use diagrams:

$$\begin{tikzcd} K_1 \arrow[dr, no head] & & K_2 \arrow[dl, no head] \\ & R & \end{tikzcd}$$

means that R is resolvent of K_1 and K_2

Ex 335 $\Box$ can be derived

We now have to show that

$$\underbrace{\Box \in Res^*(\mathcal{K})}_{\text{syntax, can be checked}} \quad \text{iff} \quad \underbrace{\mathcal{K} \text{ is unsatisfiable}}_{\text{semantics}}$$

$\Rightarrow$: soundness

$\Leftarrow$: completeness

of resolution

automatically

To prove soundness of ground resolution, we show that adding resolvents preserves equivalence.

<u>Lemma 336</u> (Propositional Resolution Lemma)

Let $\mathcal{K}$ be a set of ground clauses. If $K_1, K_2 \in \mathcal{K}$ and R is resolvent of K_1 and K_2, then $\mathcal{K}$ and $\mathcal{K} \cup \{R\}$ are equivalent.

<u>Proof</u>: "$\Leftarrow$": If there is a structure S with $S \models \mathcal{K} \cup \{R\}$, then also $S \models \mathcal{K}$.

"$\Rightarrow$": Let $S \models \mathcal{K}$.

There is a literal $L \in K_1$, $\overline{L} \in K_2$, $R = (K_1 \setminus \{L\}) \cup (K_2 \setminus \{\overline{L}\})$.

Assume that $S \not\models \mathcal{K} \cup \{R\}$, i.e., $S \not\models R$

If $S \models L$, then $S \models \mathcal{K}$ implies $S \models K_2$ which in turn implies $S \models K_2 \setminus \{\overline{L}\}$. Thus, $S \models R$ ↯.

If $S \not\models L$, then in a similar way one can show $S \models K_1 \setminus \{L\}$. Thus, $S \models R$ ↯.

$\boxed{\text{q.e.d.}}$

<u>Thm 337</u> (Soundness and Completeness of propositional resolution)

Let $\mathcal{K}$ be a (possibly infinite) set of ground clauses.

Then: $\square \in Res^*(\mathcal{K})$ iff $\mathcal{K}$ is unsatisfiable

Then: $\square \in Res^*(\mathcal{K})$ iff $\mathcal{K}$ is unsatisfiable.

Proof: "$\Rightarrow$" (Soundness)

Resolution Lemma 336 states that $\mathcal{K}$ and $Res(\mathcal{K})$ are equivalent.

By induction, one can show that $\mathcal{K}$ is equivalent to $Res^n(\mathcal{K})$ for all $n \in \mathbb{N}$.

$\square \in Res^*(\mathcal{K})$

$\curvearrowright$ there is an $n \in \mathbb{N}$ such that $\square \in Res^n(\mathcal{K})$

$\curvearrowright$ $Res^n(\mathcal{K})$ is unsatisfiable

$\curvearrowright$ $\mathcal{K}$ is unsatisfiable.

"$\Leftarrow$": (Completeness)

$\mathcal{K}$ is unsatisfiable

$\curvearrowright$ there is a finite subset $\mathcal{K}' \subseteq \mathcal{K}$ that is unsatisfiable

We prove $\square \in Res^*(\mathcal{K}')$ by induction on the number n of different atomic formulas in $\mathcal{K}'$!

Ind Base: $n = 0$

There are only 2 clause sets without atomic formulas:

$\mathcal{K}' = \emptyset$ is valid (holds in every structure)

or $\mathcal{K}' = \{\square\}$ is unsatisfiable

Then $\square \in Res^0(\mathcal{K}') \subseteq Res^*(\mathcal{K}')$.

Ind Step: $n > 0$

Let A be an atomic formula occurring in $\mathcal{K}'$!

Let $\mathcal{K}^+$ result from $\mathcal{K}'$ by omitting all clauses that contain A. Moreover, $\neg A$ is removed from all remaining clauses: $\mathcal{K}^+ = \{ K \setminus \{\neg A\} \mid K \in \mathcal{K}',\ A \notin K \}$

$$\mathcal{K}^- = \{ K \setminus \{A\} \mid K \in \mathcal{K}',\ \neg A \notin K \}$$

Clearly, A does not occur anymore in $\mathcal{K}^+$ and $\mathcal{K}^-$.
Thus: $\mathcal{K}^+$, $\mathcal{K}^-$ contain at most $n-1$ atomic formulas.

$\mathcal{K}^+$ is unsatisfiable:
If $S \models \mathcal{K}^+$, then S could be extended to a structure S' with $S' \models A$. Then: $S' \models \mathcal{K}'$. ↯ to the unsatisfiability of $\mathcal{K}'$.

Induction Hypothesis:

$$\square \in Res^*(\mathcal{K}^+), \quad \square \in Res^*(\mathcal{K}^-)$$

This means that there is a sequence of clauses $K_1, \dots, K_m$ with $K_m = \square$ and for all $1 \le i \le m$:

- $K_i \in \mathcal{K}^+$ or
- K_i is a resolvent of K_j and K_k for $j, k < i$

If those clauses $K_i \in \mathcal{K}^+$ that were used in the resolution proof are also contained in $\mathcal{K}'$, then this is already a resolution proof from $\mathcal{K}'$, i.e., $\square \in Res^*(\mathcal{K}')$.
Otherwise: re-insert $\neg A$ into those clauses K_i where it had been removed. This yields again a resolution proof from $\mathcal{K}'$ ending in $\{\neg A\}$.

from $\mathcal{K}'$ ending in $\{\neg A\}$.

(Reason:
$$
\begin{tikzcd}[column sep=small]
K_j \arrow[dr, dash] & & K_k \arrow[dl, dash] \\
& K_i &
\end{tikzcd}
\quad \rightsquigarrow \quad
\begin{tikzcd}[column sep=small]
K_j \cup \{\neg A\} \arrow[dr, dash] & & K_k \arrow[dl, dash] \\
& K_i \cup \{\neg A\} &
\end{tikzcd}
$$
)

$$\Rightarrow \{\neg A\} \in \mathrm{Res}^*(\mathcal{K}').$$

Similarly, there is a resolution proof of $\square$ from $\mathcal{K}^-$.
If this proof used only clauses from $\mathcal{K}'$, then we directly have $\square \in \mathrm{Res}^*(\mathcal{K}')$.
Otherwise: re-insert A into the clauses from $\mathcal{K}^-$

$$\Rightarrow \{A\} \in \mathrm{Res}^*(\mathcal{K}').$$

One last resolution step yields $\square \in \mathrm{Res}^*(\mathcal{K}')$:

$$
\begin{tikzcd}[column sep=small]
\{A\} \arrow[dr, dash] & & \{\neg A\} \arrow[dl, dash] \\
& \square &
\end{tikzcd}
$$

$\blacksquare$

Now we can improve the algorithm of Gilmore to the

Ground Resolution Algorithm.

Advantage over Gilmore's Alg: better check for unsatisfiability

Same disadvantage as Gilmore: step from predicate to propositional logic is done via Herbrand-expansion (instantiate variables by all possible ground terms)

Ground Res. Alg. is sound and complete:

- if $\{\varphi_1, \dots, \varphi_n\} \models \varphi$, then alg. terminates and returns "true"
- if $\{\varphi_1, \dots, \varphi_n\} \not\models \varphi$, then alg. does not return "true" (but it doesn't terminate in general)

Now: Improve the step from pred. to prop. logic (avoid a blind guess by which ground terms one has to instantiate variables)