Decision Procedures in Verification

First-Order Logic (4)

3.12.2018

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Until now:

Syntax (one-sorted signatures vs. many-sorted signatures)

Semantics

Structures (also many-sorted)

Models, Validity, and Satisfiability

Entailment and Equivalence

Theories (Syntactic vs. Semantics view)

Herbrand models $\mapsto$ The Bernays-Schönfinkel class

Algorithmic Problems

Decidability/Undecidability

Methods: Resolution

2.7 General Resolution

Propositional resolution:

refutationally complete,

clearly inferior to the DPLL procedure
(even with various improvements).

But: in contrast to the DPLL procedure, resolution can be easily extended to non-ground clauses.

Propositional resolution: reminder

Resolution inference rule:

$$\frac{C \vee A \qquad \neg A \vee D}{C \vee D}$$

Terminology: $C \vee D$: resolvent; A: resolved atom

(Positive) factorisation inference rule:

$$\frac{C \vee A \vee A}{C \vee A}$$

Resolution for ground clauses

- Exactly the same as for propositional clauses

  Ground atoms $\mapsto$ propositional variables

Theorem

Res is sound and refutationally complete (for all sets of ground clauses)

Resolution for ground clauses

- Refinements with orderings and selection functions:

  Need: - well-founded ordering on ground atomic formulae/literals

  - selection function (for negative literals)

  $$S : C \quad \mapsto \quad \text{set of occurrences of } \textit{negative} \text{ literals in } C$$

  Example of selection with selected literals indicated as $\boxed{X}$:

  $\boxed{\neg A} \vee \neg A \vee B$

  $\boxed{\neg B_0} \vee \boxed{\neg B_1} \vee A$

Resolution Calculus $Res_S^{\succ}$

Ordered resolution with selection

$$\frac{C \vee A \qquad D \vee \neg A}{C \vee D}$$

if

1. $A \succ C$;
2. nothing is selected in C by S;
3. $\neg A$ is selected in $D \vee \neg A$,
   or else nothing is selected in $D \vee \neg A$ and $\neg A \succeq \max(D)$.

Note: For positive literals, $A \succ C$ is the same as $A \succ \max(C)$.

Ordered factoring

$$\frac{C \vee A \vee A}{(C \vee A)}$$

if A is maximal in C and nothing is selected in C.

Resolution for ground clauses

Let $\succ$ be a total and well-founded ordering on ground atoms, and S a selection function.

Theorem. $\text{Res}_S^\succ$ is sound and refutationally complete for all sets of ground clauses.

Soundness: sufficient to show that

(1) $C \vee A, D \vee \neg A \models C \vee D$

(2) $C \vee A \vee A \models C \vee A$

Completeness: Let $\succ$ be a clause ordering, let N be saturated wrt. $Res_S^\succ$, and suppose that $\bot \not\in N$. Then $I_N^\succ \models N$, where $I_N^\succ$ is incrementally constructed as follows:

General Resolution through Instantiation

Idea: instantiate clauses appropriately:

Problems:

More than one instance of a clause can participate in a proof.

Even worse: There are infinitely many possible instances.

Observation:

Instantiation must produce complementary literals
(so that inferences become possible).

Idea:

Do not instantiate more than necessary to get complementary literals.

Lifting Principle

Problem: Make saturation of infinite sets of clauses as they arise from taking the (ground) instances of finitely many general clauses (with variables) effective and efficient.

Idea (Robinson 65):

- Resolution for general clauses:
- *Equality* of ground atoms is generalized to *unifiability* of general atoms;
- Only compute *most general* (minimal) unifiers.

Resolution for General Clauses

General binary resolution *Res*:

$$\frac{C \vee A \qquad D \vee \neg B}{(C \vee D)\sigma} \quad \text{if } \sigma = \mathrm{mgu}(A, B) \qquad \text{[resolution]}$$

$$\frac{C \vee A \vee B}{(C \vee A)\sigma} \quad \text{if } \sigma = \mathrm{mgu}(A, B) \quad \text{[factorization]}$$

For inferences with more than one premise, we assume that the variables in the premises are (bijectively) renamed such that they become different to any variable in the other premises.
We do not formalize this. Which names one uses for variables is otherwise irrelevant.

Unification

Let $E = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ (s_i, t_i terms or atoms) a multi-set of equality problems. A substitution σ is called a unifier of E if $s_i\sigma = t_i\sigma$ for all $1 \leq i \leq n$.

If a unifier of E exists, then E is called unifiable.

Unification after Martelli/Montanari

$$
\begin{array}{llcl}
(1) & t \doteq t, E & \Rightarrow_{MM} & E \\
(2) & f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n), E & \Rightarrow_{MM} & s_1 \doteq t_1, \ldots, s_n \doteq t_n, E \\
(3) & f(\ldots) \doteq g(\ldots), E & \Rightarrow_{MM} & \bot \\
(4) & x \doteq t, E & \Rightarrow_{MM} & x \doteq t, E[t/x] \\
 & & & \quad \text{if } x \in var(E), x \notin var(t) \\
(5) & x \doteq t, E & \Rightarrow_{MM} & \bot \\
 & & & \quad \text{if } x \neq t, x \in var(t) \\
(6) & t \doteq x, E & \Rightarrow_{MM} & x \doteq t, E \\
 & & & \quad \text{if } t \notin X
\end{array}
$$

Examples

Example 1:

$$\{x \doteq f(a), g(x,x) \doteq g(x,y)\} \quad \Rightarrow_4$$

$$\{x \doteq f(a), g(f(a),f(a)) \doteq g(f(a),y)\} \quad \Rightarrow_2$$

$$\{x \doteq f(a), f(a) \doteq f(a), f(a) \doteq y\} \quad \Rightarrow_1$$

$$\{x \doteq f(a), f(a) \doteq y\} \quad \Rightarrow_6$$

$$\{x \doteq f(a), y \doteq f(a)\}$$

Example 2:

$$\{x \doteq f(a), g(x,x) \doteq h(x,y)\} \quad \Rightarrow_3 \perp$$

Example 3:

$$\{f(x,x) \doteq f(y,g(y))\} \quad \Rightarrow_2$$

$$\{x \doteq y, x \doteq g(y)\} \quad \Rightarrow_4$$

$$\{x \doteq y, y \doteq g(y)\} \quad \Rightarrow_5 \perp$$

MM: Main Properties

If $E = x_1 \doteq u_1, \ldots, x_k \doteq u_k$, with x_i pairwise distinct, $x_i \notin var(u_j)$, then E is called an (equational problem in) solved form representing the solution $\sigma_E = [u_1/x_1, \ldots, u_k/x_k]$.

Proposition 2.28:
If E is a solved form then σ_E is am mgu of E.

Theorem 2.29:

1. If $E \Rightarrow_{MM} E'$ then σ is a unifier of E iff σ is a unifier of E'
2. If $E \Rightarrow^*_{MM} \bot$ then E is not unifiable.
3. If $E \Rightarrow^*_{MM} E'$ with E' in solved form, then $\sigma_{E'}$ is an mgu of E.

Main Unification Theorem

Theorem 2.30:

E is unifiable if and only if there is a most general unifier σ of E, such that σ is idempotent and $dom(\sigma) \cup codom(\sigma) \subseteq var(E)$.

Proof: See e.g. Baader & Nipkow: Term rewriting and all that.

Problem: *exponential growth* of terms possible

Example:

$E = \{x_1 \approx f(x_0, x_0), x_2 \approx f(x_1, x_1), \ldots, x_n \approx f(x_{n-1}, x_{n-1})\}$

m.g.u. $[x_1 \mapsto f(x_0, x_0), x_2 \mapsto f(f(x_0, x_0), f(x_0, x_0)), ...]$

$x_i \mapsto$ complete binart tree of heigth i

Solution: Use acyclic term graphs; union/find algorithms

Lifting Lemma

Lemma 2.31

Let C and D be variable-disjoint clauses. If

$$\dfrac{\begin{array}{c} C \\ \sigma \downarrow \\ C\sigma \end{array} \qquad\qquad \begin{array}{c} D \\ \rho \downarrow \\ D\rho \end{array}}{C'}$$

[propositional resolution]

then there exists a substitution τ such that

$$\dfrac{C \qquad\qquad D}{C''}$$

[general resolution]

$$\begin{array}{c} C'' \\ \rho \downarrow \\ C' = C''\tau \end{array}$$

Lifting Lemma

An analogous lifting lemma holds for factorization.

Saturation of Sets of General Clauses

Corollary 2.32:

Let N be a set of general clauses saturated under Res, i.e., $Res(N) \subseteq N$. Then also $G_\Sigma(N)$ is saturated, that is,

$$Res(G_\Sigma(N)) \subseteq G_\Sigma(N).$$

Saturation of Sets of General Clauses

Corollary 2.32:

Let N be a set of general clauses saturated under Res, i.e., $\mathit{Res}(N) \subseteq N$. Then also $G_\Sigma(N)$ is saturated, that is,

$$\mathit{Res}(G_\Sigma(N)) \subseteq G_\Sigma(N).$$

Proof:

W.l.o.g. we may assume that clauses in N are pairwise variable-disjoint. (Otherwise make them disjoint, and this renaming process changes neither $\mathit{Res}(N)$ nor $G_\Sigma(N)$.)

Let $C' \in \mathit{Res}(G_\Sigma(N))$, meaning (i) there exist resolvable ground instances $C\sigma$ and $D\rho$ of N with resolvent C', or else (ii) C' is a factor of a ground instance $C\sigma$ of C.

Case (i): By the Lifting Lemma, C and D are resolvable with a resolvent C'' with $C''\tau = C'$, for a suitable substitution τ. As $C'' \in N$ by assumption, we obtain that $C' \in G_\Sigma(N)$.

Case (ii): Similar.

Herbrand's Theorem

Lemma 2.33:

Let N be a set of Σ-clauses, let $\mathcal{A}$ be an interpretation.
Then $\mathcal{A} \models N$ implies $\mathcal{A} \models G_\Sigma(N)$.

Lemma 2.34:

Let N be a set of Σ-clauses, let $\mathcal{A}$ be a *Herbrand* interpretation.
Then $\mathcal{A} \models G_\Sigma(N)$ implies $\mathcal{A} \models N$.

Herbrand's Theorem

Theorem 2.35 (Herbrand):
A set N of Σ-clauses is satisfiable if and only if it has a Herbrand model over Σ.

Proof:
The "$\Leftarrow$" part is trivial. For the "$\Rightarrow$" part let $N \not\models \bot$.

$$
\begin{aligned}
N \not\models \bot &\Rightarrow \bot \notin \mathit{Res}^*(N) && \text{(resolution is sound)}\\
&\Rightarrow \bot \notin G_\Sigma(\mathit{Res}^*(N))\\
&\Rightarrow I_{G_\Sigma(\mathit{Res}^*(N))} \models G_\Sigma(\mathit{Res}^*(N)) && \text{(Thm. 2.23; Cor. 2.32)}\\
&\Rightarrow I_{G_\Sigma(\mathit{Res}^*(N))} \models \mathit{Res}^*(N) && \text{(Lemma 2.34)}\\
&\Rightarrow I_{G_\Sigma(\mathit{Res}^*(N))} \models N && (N \subseteq \mathit{Res}^*(N))
\end{aligned}
$$

The Theorem of Löwenheim-Skolem

Theorem 2.36 (Löwenheim–Skolem):
Let Σ be a countable signature and let S be a set of closed Σ-formulas. Then S is satisfiable iff S has a model over a countable universe.

Proof:
If both X and Σ are countable, then S can be at most countably infinite. Now generate, maintaining satisfiability, a set N of clauses from S. This extends Σ by at most countably many new Skolem functions to Σ'. As Σ' is countable, so is $T_{\Sigma'}$, the universe of Herbrand-interpretations over Σ'. Now apply Theorem 2.35.

Refutational Completeness of General Resolution

Theorem 2.37:

Let N be a set of general clauses where $Res(N) \subseteq N$. Then

$$N \models \bot \Leftrightarrow \bot \in N.$$

Proof:

Let $Res(N) \subseteq N$. By Corollary 2.32: $Res(G_\Sigma(N)) \subseteq G_\Sigma(N)$

$N \models \bot \Leftrightarrow G_\Sigma(N) \models \bot$ (Lemma 2.33/2.34; Theorem 2.35)

$\Leftrightarrow \bot \in G_\Sigma(N)$ (propositional resolution sound and complete)

$\Leftrightarrow \bot \in N$

Compactness of Predicate Logic

Theorem 2.38 (Compactness Theorem for First-Order Logic):
Let Φ be a set of first-order formulas.
Φ is unsatisfiable $\Leftrightarrow$ some finite subset $\Psi \subseteq \Phi$ is unsatisfiable.

Proof:
The "$\Leftarrow$" part is trivial. For the "$\Rightarrow$" part let Φ be unsatisfiable and let N be the set of clauses obtained by Skolemization and CNF transformation of the formulas in Φ. Clearly $\mathit{Res}^*(N)$ is unsatisfiable. By Theorem 2.37, $\bot \in \mathit{Res}^*(N)$, and therefore $\bot \in \mathit{Res}^n(N)$ for some $n \in \mathbb{N}$. Consequently, $\bot$ has a finite resolution proof B of depth $\leq n$. Choose Ψ as the subset of formulas in Φ such that the corresponding clauses contain the assumptions (leaves) of B.

2.12 Ordered Resolution with Selection

Motivation: Search space for *Res* *very* large.

Ideas for improvement:

1. In the completeness proof (Model Existence Theorem 2.23) one only needs to resolve and factor maximal atoms
   $\Rightarrow$ if the calculus is restricted to inferences involving maximal atoms, the proof remains correct
   $\Rightarrow$ *order restrictions*

2. In the proof, it does not really matter with which negative literal an inference is performed
   $\Rightarrow$ choose a negative literal don't-care-nondeterministically
   $\Rightarrow$ *selection*

Selection Functions

A selection function is a mapping

$$S : C \quad \mapsto \quad \text{set of occurrences of } \textit{negative} \text{ literals in } C$$

Example of selection with selected literals indicated as $\boxed{X}$:

$$\boxed{\neg A} \vee \neg A \vee B$$

$$\boxed{\neg B_0} \vee \boxed{\neg B_1} \vee A$$

Resolution Calculus $Res_S^{\succ}$

In the completeness proof, we talk about (strictly) maximal literals of *ground* clauses.

In the non-ground calculus, we have to consider those literals that correspond to (strictly) maximal literals of ground instances:

Let $\succ$ be a total and well-founded ordering on ground atoms.
A literal L is called [strictly] maximal in a clause C if and only if there exists a ground substitution σ such that for all L' in C: $L\sigma \succeq L'\sigma$ $[L\sigma \succ L'\sigma]$.

Resolution Calculus $Res_S^{\succ}$

Let $\succ$ be an atom ordering and S a selection function.

$$\frac{C \vee A \qquad \neg B \vee D}{(C \vee D)\sigma} \qquad \text{[ordered resolution with selection]}$$

if $\sigma = \text{mgu}(A, B)$ and

(i) $A\sigma$ strictly maximal wrt. $C\sigma$;

(ii) nothing is selected in C by S;

(iii) either $\neg B$ is selected,
or else nothing is selected in $\neg B \vee D$ and $\neg B\sigma$ is maximal in $D\sigma$.

Resolution Calculus $Res_S^{\succ}$

$$\frac{C \vee A \vee B}{(C \vee A)\sigma} \qquad\qquad [\text{ordered factoring}]$$

if $\sigma = \mathrm{mgu}(A, B)$ and $A\sigma$ is maximal in $C\sigma$ and nothing is selected in C.

Soundness and Refutational Completeness

Theorem 2.39:
Let $\succ$ be an atom ordering and S a selection function such that $Res_S^{\succ}(N) \subseteq N$. Then

$$N \models \bot \Leftrightarrow \bot \in N$$

Proof:
The "$\Leftarrow$" part is trivial. For the "$\Rightarrow$" part consider first the propositional level: Construct a candidate model I_N as for unrestricted resolution, except that clauses C in N that have selected literals are not productive, even when they are false in I_C and when their maximal atom occurs only once and positively.
The result for general clauses follows using the lifting lemma.

Craig Interpolation

Theorem: $Res_S^{\succ}$ is sound and refutationally complete.

A theoretical application of ordered resolution is Craig- Interpolation:

Theorem (Craig 57)

Let F and G be two propositional formulas such that $F \models G$.

Then there exists a formula H (called the interpolant for $F \models G$), such that H contains only propostional variables occurring both in F and in G, and such that $F \models H$ and $H \models G$.

Craig Interpolation

Proof:

Translate F and $\neg G$ into CNF.

Let N and M, resp., denote the resulting clause set.

Choose an atom ordering $\succ$ for which the propositional variables that occur in F but not in G are maximal.

Saturate N into N^* wrt. $\mathrm{Res}_S^\succ$ with an empty selection function S.

Then saturate $N^* \cup M$ wrt. $\mathrm{Res}_S^\succ$ to derive $\bot$.

As N^* is already saturated, due to the ordering restrictions only inferences need to be considered where premises, if they are from N^*, only contain symbols that also occur in G.

The conjunction of these premises is an interpolant H.

The theorem also holds for first-order formulas. For universal formulas the above proof can be easily extended. In the general case, a proof based on resolution technology is more complicated because of Skolemization.

Applications of Craig Interpolation

Modular databases

Given: Two databases (different but possibly overlapping languages)

Task: Is the union of the two databases consistent? If not: locate error

Applications of Craig Interpolation

Modular databases

Given: Two databases (different but possibly overlapping languages)

Logical modeling: $F_1 \wedge F_2$

Task: Is the union of the two databases consistent? If not: locate error

$F_1 \wedge F_2 \models \perp$

Applications of Craig Interpolation

Modular databases

Given: Two databases (different but possibly overlapping languages)

Logical modeling: $F_1 \wedge F_2$

Task: Is the union of the two databases consistent? If not: locate error

$F_1 \wedge F_2 \models \perp$

$F_1 \models \neg F_2$ (assume we are in prop. logic)

Applications of Craig Interpolation

Modular databases

Given: Two databases (different but possibly overlapping languages)

Logical modeling: $F_1 \wedge F_2$

Task: Is the union of the two databases consistent? If not: locate error

$F_1 \wedge F_2 \models \perp$

$F_1 \models \neg F_2$ (assume we are in prop. logic)

Craig Interpolation (propositional case)

There exists I containing only propositional variables occurring in F_1 and F_2 such that:

$F_1 \models I$ and $I \models \neg F_2$

Applications of Craig Interpolation

Reasoning in combinations of theories

Given: Two theories (different but possibly overlapping languages)
s.t. decision procedures for component theories for certain fragments exist

Task: Reason in the combination of the two theories

Question: Which information needs to be exchanged between provers?

Answer: Craig Interpolation

The case of two disjoint theories will be discussed later in this lecture

Applications of Craig Interpolation

Verification (programs or hardware)

Model programs as transition systems.

- Sets of states expressed as formulae
- Transitions expressed as formulae T

Question:
Can a state in a certain set of states E (error)
be reached from some state in a set I (initial) in k steps?

$\phi_I \wedge T_1 \wedge T_2 \wedge \cdots \wedge T_k \wedge \phi_E$

Applications of Craig Interpolation

Verification (programs or hardware)

Model programs as transition systems.

- Sets of states expressed as formulae
- Transitions expressed as formulae T

Question:
Can a state in a certain set of states E (error)
be reached from some state in a set I (initial) in k steps?

$$\underbrace{(\phi_I \wedge T_1)}_{F_1} \wedge \underbrace{(T_2 \wedge \cdots \wedge T_k \wedge \phi_E)}_{F_2}$$

Not reachable: $F_1 \wedge F_2 \models \perp$

Applications of Craig Interpolation

Verification (programs or hardware)

Model programs as transition systems.

- Sets of states expressed as formulae
- Transitions expressed as formulae T

Question:

Can a state in a certain set of states E (error)
be reached from some state in a set I (initial) in k steps?

$\underbrace{(\phi_I \wedge T_1)}_{F_1} \wedge \underbrace{(T_2 \wedge \cdots \wedge T_k \wedge \phi_E)}_{F_2}$ Not reachable: $F_1 \wedge F_2 \models \perp$

Interpolant: I overapproximates the set of successors of ϕ_I.

Goal

Goal: Make resolution efficient

Identify clauses which are not needed and can be discarded

Redundancy

So far: local restrictions of the resolution inference rules using orderings and selection functions.

Is it also possible to delete clauses altogether?
Under which circumstances are clauses unnecessary?
(Conjecture: e. g., if they are tautologies or if they are subsumed by other clauses.)

Intuition: If a clause is guaranteed to be neither a minimal counterexample nor productive, then we do not need it.

Recall

Construction of I for the extended clause set:

	clauses C	I_C	Δ_C	Remarks
1	$\neg P_0$	$\emptyset$	$\emptyset$	
2	$P_0 \vee P_1$	$\emptyset$	$\{P_1\}$	
3	$P_1 \vee P_2$	$\{P_1\}$	$\emptyset$	
4	$\neg P_1 \vee P_2$	$\{P_1\}$	$\{P_2\}$	
9	$\neg P_1 \vee \neg P_1 \vee P_3 \vee P_0$	$\{P_1, P_2\}$	$\{P_3\}$	
8	$\neg P_1 \vee \neg P_1 \vee P_3 \vee P_3 \vee P_0$	$\{P_1, P_2, P_3\}$	$\emptyset$	true in $\mathcal{A}_C$
5	$\neg P_1 \vee P_4 \vee P_3 \vee P_0$	$\{P_1, P_2, P_3\}$	$\emptyset$	
6	$\neg P_1 \vee \neg P_4 \vee P_3$	$\{P_1, P_2, P_3\}$	$\emptyset$	true in $\mathcal{A}_C$
7	$\neg P_3 \vee P_5$	$\{P_1, P_2, P_3\}$	$\{P_5\}$	

The resulting $I = \{P_1, P_2, P_3, P_5\}$ is a model of the clause set.

A Formal Notion of Redundancy

Let N be a set of ground clauses and C a ground clause (not necessarily in N). C is called redundant w. r. t. N, if there exist $C_1, \dots, C_n \in N$, $n \geq 0$, such that $C_i \prec C$ and $C_1, \dots, C_n \models C$.

Redundancy for general clauses:
C is called redundant w. r. t. N, if all ground instances $C\sigma$ of C are redundant w. r. t. $G_\Sigma(N)$.

Intuition: Redundant clauses are neither minimal counterexamples nor productive.

Note: The same ordering $\succ$ is used for ordering restrictions and for redundancy (and for the completeness proof).

Examples of Redundancy

Proposition 2.40:

- C tautology (i.e., $\models C$) $\;\Rightarrow\;$ C redundant w. r. t. any set N.
- $C\sigma \subset D \;\Rightarrow\; D$ redundant w. r. t. $N \cup \{C\}$
- $C\sigma \subseteq D \;\Rightarrow\; D \vee \overline{L}\sigma$ redundant w. r. t. $N \cup \{C \vee L,\; D\}$

(Under certain conditions one may also use non-strict subsumption, but this requires a slightly more complicated definition of redundancy.)

Saturation up to Redundancy

N is called saturated up to redundancy (wrt. $\mathit{Res}_S^{\succ}$)

$$:\Leftrightarrow \mathit{Res}_S^{\succ}(N \setminus \mathit{Red}(N)) \subseteq N \cup \mathit{Red}(N)$$

Theorem 2.41:

Let N be saturated up to redundancy. Then

$$N \models \bot \Leftrightarrow \bot \in N$$

Saturation up to Redundancy

Proof (Sketch):

(i) Ground case:

- consider the construction of the candidate model $I_N^{\succ}$ for $Res_S^{\succ}$
- redundant clauses are not productive
- redundant clauses in N are not minimal counterexamples for $I_N^{\succ}$

The premises of "essential" inferences are either minimal counterexamples or productive.

(ii) Lifting: no additional problems over the proof of Theorem 2.39.

Monotonicity Properties of Redundancy

Theorem 2.42:

(i) $N \subseteq M \Rightarrow Red(N) \subseteq Red(M)$

(ii) $M \subseteq Red(N) \Rightarrow Red(N) \subseteq Red(N \setminus M)$

Proof:

(i) Let $C \in Red(N)$. Then there exist $C_1, \ldots, C_n \in N$, $n \geq 0$ such that $C_i \prec C$ for all $i = 1, \ldots, n$ and $C_1, \ldots, C_n \models C$.

We assumed that $N \subseteq M$, so we know that $C_1, \ldots, C_n \in M$. Thus: there exist $C_1, \ldots, C_n \in M$, $n \geq 0$ such that $C_i \prec C$ for all $i = 1, \ldots, n$ and $C_1, \ldots, C_n \models C$. Therefore, $C \in Red(M)$.

Monotonicity Properties of Redundancy

Theorem 2.42:

(i) $N \subseteq M \Rightarrow Red(N) \subseteq Red(M)$

(ii) $M \subseteq Red(N) \Rightarrow Red(N) \subseteq Red(N \setminus M)$

Proof (Idea):

(ii) Let $C \in Red(N)$. Then there exist $C_1, \ldots, C_n \in N$, $n \geq 0$ such that $C_i \prec C$ for all $i = 1, \ldots, n$ and $C_1, \ldots, C_n \models C$.

Case 1: For all i, $C_i \notin M$. Then $C \in Red(N \setminus M)$.

Case 2: For some i, $C_i \in M \subseteq Red(N)$. Then for every such index i there exist $C_1^i, \ldots, C_{n_i}^i \in N$ such that $C_j^i \prec C_i$ and $C_1^i, \ldots, C_{n_i}^i \models C_i$. We can replace C_i above with $C_1^i, \ldots, C_{n_i}^i$. We can iterate the procedure until none of the C_i's are in M (termination guaranteed by the fact that $\succ$ is well-founded).

Some theorem provers for first-order logic

- SPASS `http://www.spass-prover.org/`
- E `http://www4.informatik.tu-muenchen.de/~schulz/E/E.html`
- Vampire `http://www.vprover.org/`

Decidable subclasses of first-order logic

Applications

Use ordered resolution with selection to give a decision procedure for the Ackermann class.

The Ackermann class

$\Sigma = (\Omega, \Pi)$, Ω is a finite set of constants

The Ackermann class consists of all sentences of the form

$$\exists x_1 \ldots \exists x_n \forall x \exists y_1 \ldots \exists y_m F(x_1, \ldots, x_n, x, y_1, \ldots, y_m)$$

Idea: CNF translation:

$$\begin{aligned}
&\exists x_1 \ldots \exists x_n \forall x \exists y_1 \ldots \exists y_m F(x_1, \ldots, x_n, x, y_1, \ldots, y_m) \\
&\qquad \Rightarrow_S \forall x F(\overline{c}_1, \ldots, \overline{c}_n, x, f_1(x), \ldots, f_m(x)) \\
&\qquad \Rightarrow_K \forall x \bigwedge \bigvee L_i(c_1, \ldots, c_n, x, f_1(x), \ldots, f_m(x))
\end{aligned}$$

$c_1, \ldots, c_n$ are Skolem constants

$f_1, \ldots, f_m$ are unary Skolem functions

The Ackermann class

$\Sigma = (\Omega, \Pi)$, Ω is a finite set of constants

The Ackermann class consists of all sentences of the form

$$\exists x_1 \ldots \exists x_n \forall x \exists y_1 \ldots \exists y_m F(x_1, \ldots, x_n, x, y_1, \ldots, y_m)$$

Idea: CNF translation:

$$\begin{aligned} &\exists x_1 \ldots \exists x_n \forall x \exists y_1 \ldots \exists y_m F(x_1, \ldots, x_n, x, y_1, \ldots, y_m) \\ &\qquad \Rightarrow^* \forall x \bigwedge \bigvee L_i(c_1, \ldots, c_n, x, f_1(x), \ldots, f_m(x)) \end{aligned}$$

The clauses are in the following classes:

$G = G(c_1, \ldots, c_n)$ ground clauses without function symbols
$V = V(x, c_1, \ldots, c_n)$ clauses with one variable and without function symbols
$G_f = G(c_1, \ldots, c_n, f_1, \ldots, f_n)$ ground clauses with function symbols
$V_f = V(x, c_1, \ldots, c_n, f_1(x), \ldots, f_n(x))$ clauses with a variable & function symbols

The Ackermann class

$G = G(c_1, \ldots, c_n)$ ground clauses without function symbols
$V = V(x, c_1, \ldots, c_n)$ clauses with one variable and without function symbols
$G_f = G(c_1, \ldots, c_n, f_1, \ldots, f_n)$ ground clauses with function symbols
$V_f = V(x, c_1, \ldots, c_n, f_1(x), \ldots, f_n(x))$ clauses with a variable & function symbols

Term ordering

$f(t) \succ t$; terms containing function symbols larger than those who do not.

$B \succ A$ iff exists argument u of B such that every argument t of A: $u \succ t$

Ordered resolution: $G \cup V \cup G_f \cup V_f$ is closed under ordered resolution.

$G, G \mapsto G$; $G, V \mapsto G$; $G, G_f \mapsto$ nothing; $G, V_f \mapsto$ nothing

$V, V \mapsto V \cup G$; $V, G_f \mapsto G \cup G_f$; $V, V_f \mapsto G \cup V \cup G_f \cup V_f$

$G_f, G_f \mapsto G_f$; $G_f, V_f \mapsto G_f \cup G$; $V_f, V_f \mapsto G \cup V \cup V_f \cup G_f$

Observation 1: $G \cup V \cup G_f \cup V_f$ finite set of clauses (up to renaming of variables).

The Ackermann class

$G = G(c_1, \dots, c_n)$ ground clauses without function symbols
$V = V(x, c_1, \dots, c_n)$ clauses with one variable and without function symbols
$G_f = G(c_1, \dots, c_n, f_i)$ ground clauses with function symbols
$V_f = V(x, c_1, \dots, c_n, f_1(x), \dots, f_n(x))$ clauses with a variable & function symbols

Term ordering

$f(t) \succ t$; terms containing function symbols larger than those who do not.

$B \succ A$ iff exists argument u of B such that every argument t of A: $u \succ t$

Ordered resolution: $G \cup V \cup G_f \cup V_f$ is closed under ordered resolution.

$G, G \mapsto G$; $\quad G, V \mapsto G$; $\quad G, G_f \mapsto$ nothing; $\quad G, V_f \mapsto$ nothing

$V, V \mapsto V \cup G$; $\quad V, G_f \mapsto G \cup G_f$; $\quad V, V_f \mapsto G \cup V \cup G_f \cup V_f$

$G_f, G_f \mapsto G_f$; $\quad G_f, V_f \mapsto G_f \cup G$; $\quad V_f, V_f \mapsto G \cup V \cup V_f \cup G_f$

Observation 2: No clauses with nested function symbols can be generated.

The Ackermann Class

Conclusion:

Resolution (with implicit factorization) will always terminate if the input clauses are in the class defined before.

Resolution can be used as a decision procedure to check the satisfiability of formulae in the Ackermann class.

The Monadic Class

Monadic first-order logic (MFO) is FOL (without equality) over purely relational signatures $\Sigma = (\Omega, \Pi)$, where $\Omega = \emptyset$, and every $p \in \Pi$ has arity 1.

Abstract syntax:

$$\Phi := \top \mid P(x) \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \Phi_1 \vee \Phi_2 \mid \forall x\Phi \mid \exists x\Phi$$

Idea. Let Φ be a MFO formula with k predicate symbols.

Let $\mathcal{A} = (U_{\mathcal{A}}, \{p_{\mathcal{A}}\}_{p\in\Pi})$ be a Σ-algebra. The only way to distinguish the elements of $U_{\mathcal{A}}$ is by the atomic formulae $p(x)$, $p \in \Pi$.

- the elements which $a \in U_{\mathcal{A}}$ which belong to the same $p_{\mathcal{A}}$'s, $p \in \Pi$ can be collapsed into one single element.
- if $\Pi = \{p^1, \dots, p^k\}$ then what remains is a *finite structure* with at most 2^k elements.
- the truth value of a formula: computed by evaluating all subformulae.