# Decision Procedures for Verification

First-Order Logic (4)

9.12.2014

Viorica Sofronie-Stokkermans

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)**

**Algorithmic Problems:** Check satisfiability

# Until now:

**Normal Forms**

**Herbrand Models**

decidability of the Bernays-Schönkinkel class

**Resolution**

# 2.8 Ordered Resolution with Selection

Motivation: Search space for *Res very* large.

Ideas for improvement:

1. In the completeness proof (Model Existence Theorem) 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 $\sigma$ 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.

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

if $\sigma = \mathrm{mgu}(A, B)$ 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 $Res_S^{\succ}$

$$\frac{C \vee A \vee B}{(C \vee A)\sigma}$$ [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 2.27:**

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.

# 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 $C$ | $I_C$ | $\Delta_C$ | 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 $C_1, \ldots, C_n \in N$, $n \geq 0$, such that $C_i \prec C$ and $C_1, \ldots, C_n \models C$.

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 2.28:**

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

(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. $Res_S^{\succ}$)

$$:\Leftrightarrow Res_S^{\succ}(N \setminus Red(N)) \subseteq N \cup Red(N)$$

**Theorem 2.29:**

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 2.27.

# Monotonicity Properties of Redundancy

**Theoyrem 2.30:**

(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 2.30:**

(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/`

Example (SPASS)

# Consequences

**Corollary 2.22:**

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,

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

**Lemma 2.23:**

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

**Lemma 2.24:**

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

# Consequences

**Theorem (Herbrand's theorem):**
A set $N$ of $\Sigma$-clauses is satisfiable if and only if it has a Herbrand model over $\Sigma$.

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

$$
\begin{aligned}
N \not\models \bot &\Rightarrow \bot \notin Res^*(N) && \text{(resolution is sound)} \\
&\Rightarrow \bot \notin G_\Sigma(Res^*(N)) \\
&\Rightarrow I_{G_\Sigma(Res^*(N))} \models G_\Sigma(Res^*(N)) && \text{(Lemma. 2.23; Cor. 2.22)} \\
&\Rightarrow I_{G_\Sigma(Res^*(N))} \models Res^*(N) && \text{(Lemma 2.24)} \\
&\Rightarrow I_{G_\Sigma(Res^*(N))} \models N && (N \subseteq Res^*(N))
\end{aligned}
$$

# The Theorem of Löwenheim-Skolem

**Theorem 2.31 (Löwenheim–Skolem):**
Let $\Sigma$ be a countable signature and let $S$ be a set of closed $\Sigma$-formulas. Then $S$ is satisfiable iff $S$ has a model over a countable universe.

Proof:
If both $X$ and $\Sigma$ are countable, then $S$ can be at most countably infinite. Now generate, maintaining satisfiability, a set $N$ of clauses from $S$. This extends $\Sigma$ by at most countably many new Skolem functions to $\Sigma'$. As $\Sigma'$ is countable, so is $T_{\Sigma'}$, the universe of Herbrand-interpretations over $\Sigma'$. Now apply Theorem 2.25 (Herbrand's theorem).

# Compactness of Predicate Logic

**Theorem 2.32** (Compactness Theorem for First-Order Logic):
Let $\Phi$ be a set of first-order formulas.
$\Phi$ is unsatisfiable $\Leftrightarrow$ some finite subset $\Psi \subseteq \Phi$ is unsatisfiable.

Proof:
The "$\Leftarrow$" part is trivial. For the "$\Rightarrow$" part let $\Phi$ be unsatisfiable and let $N$ be the set of clauses obtained by Skolemization and CNF transformation of the formulas in $\Phi$. Clearly $Res^*(N)$ is unsatisfiable.

Since resolution is complete, $\bot \in Res^*(N)$, $\bot \in Res^n(N)$ for some $n \in \mathbb{N}$. Consequently, $\bot$ has a finite resolution proof $B$ of depth $\leq n$. Choose $\Psi$ as the subset of formulas in $\Phi$ such that the corresponding clauses contain the assumptions (leaves) of $B$.

# Craig Interpolation

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. $\mathrm{Res}^{\succ}_{S}$ with an empty selection function $S$.

Then saturate $N^* \cup M$ wrt. $\mathrm{Res}_{\succ S}$ 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

$F_1 \wedge F_2 \models \bot$

# 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

$F_1 \wedge F_2 \models \bot$

$F_1 \models \neg F_2$ (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

$F_1 \wedge F_2 \models \bot$

$F_1 \models \neg F_2$ (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: $F_1 \wedge F_2 \models \bot$

# 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: $F_1 \wedge F_2 \models \bot$

Interpolant: $I$ overapproximates the set of successors of $\phi_I$.

# Decidable subclasses of first-order logic

# Applications

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

# The Ackermann class

$\Sigma = (\Omega, \Pi)$, $\Omega$ 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:

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

$c_1, \ldots, c_n$ are Skolem constants

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

# The Ackermann class

$\Sigma = (\Omega, \Pi)$, $\Omega$ 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:

$$\exists x_1 \ldots \exists x_n \forall x \exists y_1 \ldots \exists y_m F(x_1, \ldots, x_n, x, y_1, \ldots, y_m)$$
$$\Rightarrow^* \forall x \bigwedge \bigvee L_i(c_1, \ldots, c_n, x, f_1(x), \ldots, f_m(x))$$

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

$G = G(c_1, \ldots, c_n)$ ground clauses without function symbols
$V = V(x, c_1, \ldots, c_n)$ clauses with one variable and without function symbols
$G_f = G(c_1, \ldots, c_n, f_i)$ 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 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.