

Decision Procedures in Verification

First-Order Logic (4)

28.11.2013

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)

Algorithmic Problems

Decidability/Undecidability

Methods: Resolution

Normal Forms and Skolemization

Herbrand interpretations

The Bernays-Schönfinkel Class

(definition; decidability; tractable fragment: Horn clauses)

Until now:

General Resolution

Soundness, refutational completeness

Refinements: Ordered resolution with selection

Consequences:

Herbrand's theorem

The Theorem of Löwenheim-Skolem

Compactness of first-order logic

Consequences

Theorem 2.35 (Herbrand's theorem):

A set N of Σ -clauses is satisfiable if and only if it has a Herbrand model over Σ .

Theorem 2.36 (The theorem of 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.

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.

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 propositional 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. Res_{\succ}^S with an empty selection function S .

Then saturate $N^* \cup M$ wrt. Res_{\succ}^S to derive \perp .

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

$$F_1 \wedge F_2 \models \perp$$

$$F_1 \models \neg F_2 \quad (\text{assume we are in prop. logic})$$

Craig Interpolation (propositional case)

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

$$F_1 \models I \text{ and } I \models \neg F_2$$

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?

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

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. $N \cup \{C\}$
- $C\sigma \subseteq D \Rightarrow D \vee \bar{L}\sigma$ 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. $Res_S^>$)

$$:\Leftrightarrow Res_S^>(N \setminus Red(N)) \subseteq N \cup Red(N)$$

Theorem 2.41:

Let N be saturated up to redundancy. Then

$$N \models \perp \Leftrightarrow \perp \in N$$

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:

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

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

Proof:

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

We assumed that $N \subseteq M$, so we know that $C_1, \dots, C_n \in M$.

Thus: there exist $C_1, \dots, C_n \in M, n \geq 0$ such that $C_i \prec C$ for all $i = 1, \dots, n$ and $C_1, \dots, C_n \Vdash C$. Therefore, $C \in Red(M)$.

Monotonicity Properties of Redundancy

Theorem 2.42:

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

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

Proof (Idea):

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

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 $C_1^i, \dots, C_{n_i}^i \in N$ such that $C_j^i \prec C_i$ and $C_1^i, \dots, C_{n_i}^i \models C_i$. We can replace C_i above with $C_1^i, \dots, C_{n_i}^i$. 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)

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

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

Idea: CNF translation:

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

c_1, \dots, c_n are Skolem constants

f_1, \dots, 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

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

Idea: CNF translation:

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

The clauses are in the following classes:

$G = G(c_1, \dots, c_n)$ ground clauses without function symbols

$V = V(x, c_1, \dots, c_n)$ clauses with one variable and without function symbols

$G_f = G(c_1, \dots, c_n, f_1, \dots, f_n)$ 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

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

$G_f = G(c_1, \dots, c_n, f_1, \dots, f_n)$ 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.

$G, G \mapsto G$; $G, V \mapsto G$; $G, G_f \mapsto$ 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.

$G, G \mapsto G$; $G, V \mapsto G$; $G, G_f \mapsto$ 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.

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:

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

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.

The Monadic Class

MFO Abstract syntax: $\Phi := \top \mid P(x) \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \forall x\Phi$

Theorem (Finite model theorem for MFO). If Φ is a satisfiable MFO formula with k predicate symbols then Φ has a model where the domain is a subset of $\{0, 1\}^k$.

Proof: Let $\mathcal{B} = (\{0, 1\}^k, \{p_{\mathcal{B}}^1, \dots, p_{\mathcal{B}}^k\})$, where $p_{\mathcal{B}}^i = \{(b_1, \dots, b_k) \mid b_i = 1\}$.

Let $\mathcal{A} = (U_{\mathcal{A}}, \{p_{\mathcal{A}}^1, \dots, p_{\mathcal{A}}^k\})$, $\beta : X \rightarrow U_{\mathcal{A}}$ be such that $(\mathcal{A}, \beta) \models \Phi$.

We construct a model for Φ with cardinality at most 2^k as follows:

- Let $h : \mathcal{A} \rightarrow \mathcal{B}$ be defined for all $a \in U_{\mathcal{A}}$ by:

$$h(a) = (b_1, \dots, b_k) \text{ where } b_i = 1 \text{ if } a \in p_{\mathcal{A}}^i \text{ and } 0 \text{ otherwise.}$$

Then $a \in p_{\mathcal{A}}^i$ iff $h(a) \in p_{\mathcal{B}}^i$ for all $a \in U_{\mathcal{A}}$ and all $i = 1, \dots, k$.

- Let $\mathcal{B}' = (\{0, 1\}^k \cap h(U_{\mathcal{A}}), \{p_{\mathcal{B}}^1 \cap h(U_{\mathcal{A}}), \dots, p_{\mathcal{B}}^k \cap h(U_{\mathcal{A}})\})$.
- We show that $(\mathcal{B}', \beta \circ h) \models \Phi$.

The Monadic Class

Let $\mathcal{B} = (\{0, 1\}^k, \{p_{\mathcal{B}}^1, \dots, p_{\mathcal{B}}^k\})$, where $p_{\mathcal{B}}^i = \{(b_1, \dots, b_k) \mid b_i = 1\}$.

Let $\mathcal{A} = (U_{\mathcal{A}}, \{p_{\mathcal{A}}^1, \dots, p_{\mathcal{A}}^k\})$, $\beta : X \rightarrow U_{\mathcal{A}}$ be such that $(\mathcal{A}, \beta) \models \Phi$.

We construct a model for Φ with cardinality at most 2^k as follows:

- Let $h : \mathcal{A} \rightarrow \mathcal{B}$ be defined for all $a \in U_{\mathcal{A}}$ by:

$$h(a) = (b_1, \dots, b_k) \text{ where } b_i = 1 \text{ if } a \in p_{\mathcal{A}}^i \text{ and } 0 \text{ otherwise.}$$

Then $a \in p_{\mathcal{A}}^i$ iff $h(a) \in p_{\mathcal{B}}^i$ for all $a \in U_{\mathcal{A}}$ and all $i = 1, \dots, k$.

- Let $\mathcal{B}' = (\{0, 1\}^k \cap h(U_{\mathcal{A}}), \{p_{\mathcal{B}}^1 \cap h(U_{\mathcal{A}}), \dots, p_{\mathcal{B}}^k \cap h(U_{\mathcal{A}})\})$.
- We show that $(\mathcal{B}', \beta \circ h) \models \Phi$.

Induction on the structure of Φ

- $\Phi = \top$ OK
- $\Phi = p^i(x)$. Then $(\mathcal{A}, \beta) \models \Phi$ iff $\beta(x) \in p_{\mathcal{A}}^i$ iff $h(\beta(x)) \in p_{\mathcal{B}}^i$ iff $(\mathcal{B}', \beta \circ h) \models \Phi$.

The Monadic Class

Let $\mathcal{B} = (\{0, 1\}^k, \{p_{\mathcal{B}}^1, \dots, p_{\mathcal{B}}^k\})$, where $p_{\mathcal{B}}^i = \{(b_1, \dots, b_k) \mid b_i = 1\}$.

Let $\mathcal{A} = (U_{\mathcal{A}}, \{p_{\mathcal{A}}^1, \dots, p_{\mathcal{A}}^k\})$, $\beta : X \rightarrow U_{\mathcal{A}}$ be such that $(\mathcal{A}, \beta) \models \Phi$.

We construct a model for Φ with cardinality at most 2^k as follows:

- Let $h : \mathcal{A} \rightarrow \mathcal{B}$ be defined for all $a \in U_{\mathcal{A}}$ by:

$$h(a) = (b_1, \dots, b_k) \text{ where } b_i = 1 \text{ if } a \in p_{\mathcal{A}}^i \text{ and } 0 \text{ otherwise.}$$

Then $a \in p_{\mathcal{A}}^i$ iff $h(a) \in p_{\mathcal{B}}^i$ for all $a \in U_{\mathcal{A}}$ and all $i = 1, \dots, k$.

- Let $\mathcal{B}' = (\{0, 1\}^k \cap h(U_{\mathcal{A}}), \{p_{\mathcal{B}}^1 \cap h(U_{\mathcal{A}}), \dots, p_{\mathcal{B}}^k \cap h(U_{\mathcal{A}})\})$.
- We show that $(\mathcal{B}', \beta \circ h) \models \Phi$.

Induction on the structure of Φ

- $\Phi = \Phi_1 \wedge \Phi_2$: standard
- $\Phi = \neg \Phi_1$: standard

The Monadic Class

Let $\mathcal{B} = (\{0, 1\}^k, \{p_{\mathcal{B}}^1, \dots, p_{\mathcal{B}}^k\})$, where $p_{\mathcal{B}}^i = \{(b_1, \dots, b_k) \mid b_i = 1\}$.

Let $\mathcal{A} = (U_{\mathcal{A}}, \{p_{\mathcal{A}}^1, \dots, p_{\mathcal{A}}^k\})$, $\beta : X \rightarrow U_{\mathcal{A}}$ be such that $(\mathcal{A}, \beta) \models \Phi$.

We construct a model for Φ with cardinality at most 2^k as follows:

- Let $h : \mathcal{A} \rightarrow \mathcal{B}$ be defined for all $a \in U_{\mathcal{A}}$ by:

$$h(a) = (b_1, \dots, b_k) \text{ where } b_i = 1 \text{ if } a \in p_{\mathcal{A}}^i \text{ and } 0 \text{ otherwise.}$$

Then $a \in p_{\mathcal{A}}^i$ iff $h(a) \in p_{\mathcal{B}}^i$ for all $a \in U_{\mathcal{A}}$ and all $i = 1, \dots, k$.

- Let $\mathcal{B}' = (\{0, 1\}^k \cap h(U_{\mathcal{A}}), \{p_{\mathcal{B}}^1 \cap h(U_{\mathcal{A}}), \dots, p_{\mathcal{B}}^k \cap h(U_{\mathcal{A}})\})$.
- We show that $(\mathcal{B}', \beta \circ h) \models \Phi$.

Induction on the structure of Φ

- $\Phi = \forall x \Phi_1(x)$. Then the following are equivalent:
 - $(\mathcal{A}, \beta) \models \Phi$ (i.e. $(\mathcal{A}, \beta[x \mapsto a]) \models \Phi_1$ for all $a \in U_{\mathcal{A}}$)
 - $(\mathcal{B}', \beta[x \mapsto a] \circ h) \models \Phi_1$ for all $a \in U_{\mathcal{A}}$ (ind. hyp)
 - $(\mathcal{B}', \beta \circ h[x \mapsto b]) \models \Phi_1$ for all $b \in \{0, 1\}^k \cap h(U_{\mathcal{A}})$ (i.e. $(\mathcal{B}', \beta \circ h) \models \Phi$)

The Monadic Class

Resolution-based decision procedure for the Monadic Class (and for several other classes):

William H. Joyner Jr.

Resolution Strategies as Decision Procedures.

J. ACM 23(3): 398-417 (1976)

Idea:

- Use orderings to restrict the possible inferences
- Identify a class of clauses (with terms of bounded depth) which contains the type of clauses generated from the respective fragment and is closed under ordered resolution (+ red. elim. criteria)
- Show that a saturation of the clauses can be obtained in finite time

The Monadic Class

Resolution-based decision procedure for the Monadic Class:

$$\Phi : \quad \forall \bar{x}_1 \exists \bar{y}_1 \dots \forall \bar{x}_k \exists \bar{y}_k (\dots p^s(x_i) \dots p^l(y_i) \dots)$$

$$\mapsto \quad \forall \bar{x}_1 \dots \forall \bar{x}_k (\dots p^s(x_i) \dots p^l(f_{sk}(\bar{x}_1, \dots, \bar{x}_i) \dots))$$

Consider the class MON of clauses with the following properties:

- no literal of height greater than 2 appears
- each variable-disjoint partition has at most $n = \sum_{i=1} |\bar{x}_i|$ variables (can order the variables as x_1, \dots, x_n)
- the variables of each non-ground block can occur either in atoms $p(x_i)$ or in atoms $P(f_{sk}(x_1, \dots, x_t))$, $0 \leq t \leq n$

It can be shown that this class contains all CNF's of formulae in the monadic class and is closed under ordered resolution.