# Computational Logic

## Resolution Strategies

Damiano Zanardini

UPM European Master in Computational Logic (EMCL)
School of Computer Science
Technical University of Madrid
damiano@fi.upm.es

Academic Year 2008/2009

# Introduction

## The problem

- the method of *saturation* from a set $\mathcal{C}$ generates, if not limited, a big number of clauses which are redundant or irrelevant
- it is necessary to use systematic *selection rules* which make the process *simpler* and *computationally efficient*
- two kinds of criteria
  - *simplification strategies*: reducing the number of clauses
  - *refinement strategies*: limiting the generation of clauses

## Terminology

- $\mathcal{C}$ is the *initial* set of clauses
- $\mathcal{C}'$ is the *current* set of clauses (at some point during the deduction process where we want to apply the rules)

# Simplification Strategies

1. Elimination of identical clauses
   - obviously, $\mathcal{C} \vdash_{MGU} \Box$ iff $\Box$ can be derived by eliminating identical clauses (apart from one copy, of course)

## How to do it

- if a clause is generated which already appears in $\mathcal{C}'$, then it is not included

# Simplification Strategies

## 2. Elimination of clauses with pure literals

- a literal $L$ is *pure* iff there does not exist in the set a literal $\neg L'$ where $L$ and $L'$ are unifiable
- $\mathcal{C} \vdash_{MGU} \Box$ iff $\Box$ can be derived after removing from $\mathcal{C}$ clauses with pure literals
  - a clause with pure literals is useless for refutation since it will never be eliminated by resolution

## How to do it

- clauses with pure literals are removed from the set
- it is enough to apply this strategy *once*, since no new clauses with pure literals will be generated

# Simplification Strategies

3. Elimination of tautological clauses

- $\mathcal{C} \vdash_{MGU} \Box$ iff $\Box$ can be derived from $\mathcal{C}$ after removing tautologies

## How to do it

- if a clause is generated which is a tautology, then it is not included in $\mathcal{C}'$

# Simplification Strategies

**Example:** $\mathcal{C} = \{p \vee q,\ \neg p \vee q,\ p \vee \neg q,\ \neg p \vee \neg q\}$

- by applying all the simplification rules, the derivation comes to be

| | | |
|---|---|---|
| (1) | $p \vee q$ | |
| (2) | $\neg p \vee q$ | |
| (3) | $p \vee \neg q$ | |
| (4) | $\neg p \vee \neg q$ | |
| (5) | $q$ | (1,2) |
| (6) | $p$ | (1,3) |
| (7) | $\neg p$ | (2,4) |
| (8) | $\neg q$ | (3,4) |
| (9) | $\square$ | (5,8) |

- it must be noted that no other *clever* strategy has been used

# Simplification Strategies

4. Elimination of subsumed clauses
   - a clauses $C$ *subsumes* another clause $D$ if there exists a substitution $\alpha$ such that $C\alpha$ is a subformula of $D$: $D = C\alpha \vee D'$

## Example

$D = p(f(a), x) \vee q(g(y), y) \vee r(b)$ is subsumed by $C = r(z) \vee p(f(u), v)$ under $\alpha = \{u/a, v/x, z/b\}$

# Simplification Strategies

## Lemma (subsumed clauses)

*The set* $\{C_1, .., C_n, C, C\alpha \vee D\}$ *is unsatisfiable iff* $\{C_1, .., C_n, C\}$ *is*

## Proof ($\rightarrow$).

1. $UNSAT(\{C_1, .., C_n, C, C\alpha \vee D\})$
2. suppose $SAT(\{C_1, .., C_n, C\})$: there exists a Herbrand interpretation $I_H$ which makes all $C_i$ and $C$ true
3. $I_H$ makes $C\alpha$ true (since universal quantification is implicit), so that it also makes $C\alpha \vee D$ true
4. $I_H$ satisfies $\{C_1, .., C_n, C, C\alpha \vee D\}$: contradiction with ❶
5. $UNSAT(\{C_1, .., C_n, C\})$

# Simplification Strategies

## Lemma (subsumed clauses)

*The set* $\{C_1, .., C_n, C, C\alpha \vee D\}$ *is unsatisfiable iff* $\{C_1, .., C_n, C\}$ *is*

## Proof ($\leftarrow$).

1. $UNSAT(\{C_1, .., C_n, C\})$
2. there is no interpretation which makes $C_i$ and $C$ true
3. there is no interpretation which makes $C_i$, $C$ and $C\alpha \vee D$ true
4. $UNSAT(\{C_1, .., C_n, C, C\alpha \vee D\})$

# Simplification Strategies

## Procedure for deciding subsumption: is $C_1$ subsumed by $C_2$?

Procedure $\textsc{is\_subsumed\_by}\,(C_1, C_2)$:

---

**if** ($C_2$ is empty) **then return** YES: $C_1$ is subsumed by $C_2$ **else**
  **if** (($p(\bar{t}) \in C_2$ and there is no $p(\bar{t}') \in C_1$) $\vee$
    ($\neg p(\bar{t}) \in C_2$ and there is no $\neg p(\bar{t}') \in C_1$))
  **then return** NO: $C_1$ is not subsumed by $C_2$
  $L_2 = q(\bar{t})$ is the first literal in $C_2$
  $CL_1 = \{q(\bar{t'}) \in C_1 \mid \bar{t'}$ are terms$\}$
  **for each** ($L \in CL_1$)
    $\mu_L = MGU(L_2, L)$ such that $Domain\,(\mu_L) \cap Vars(L) = \emptyset$
    **if** (such $\mu_L$ exists)
      $C_2'$ is $C_2$ where $L_2$ has been removed
      $C_2'' = C_2'\mu_L$
      **if** ($\textsc{is\_subsumed\_by}\,(C_1, C_2'') =$ YES) **then**
        **return** YES: $C_1$ is subsumed by $C_2$
  **return** NO: $C_1$ is not subsumed by $C_2$

# Simplification Strategies

Example: $C_1 = p(a,s) \vee p(b,z) \vee \neg q(f(z),b)$,
$C_2 = p(x,y) \vee \neg q(w,x)$

- $L_2 = p(x,y)$
- $CL_1 = \{p(a,s), p(b,z)\}$
- $\mu_{p(a,s)} = \{x/a, y/s\}$
- $\mu_{p(b,z)} = \{x/b, y/z\}$
- $\mu_{p(a,s)} \rightsquigarrow C_2'' = \neg q(w,a)$
- $q(w,a)$ and $q(f(z),b)$ are not unifiable
- $\mu_{p(b,z)} \rightsquigarrow C_2'' = \neg q(w,b)$
- and $MGU(q(w,b), q(f(z),b)) = \{w/f(z)\}$
- therefore, $C_1$ is subsumed by $C_2$

# Search Trees

## Derivations

A *derivation* of $C$ from $\{C_1, .., C_n\}$ is a sequence $\langle C_1, .., C_n, R_1, .., R_m \rangle$ such that

- every $R_i$ is the resolvent of two previous clauses
- no resolution step is done more than once
- $R_m = C$

## Refutations

A *refutation* of $\{C_1, .., C_n\}$ is a derivation of $\Box$ from $\{C_1, .., C_n\}$

## Facts

- a derivation is a correct deduction (by correctness of *MGU* resolution)
- if $UNSAT(\mathcal{C})$, then there exists a refutation for $\mathcal{C}$ (by completeness of *MGU* resolution)

# Search Trees

## Search tree $T$ of $\{C_1, .., C_n\}$

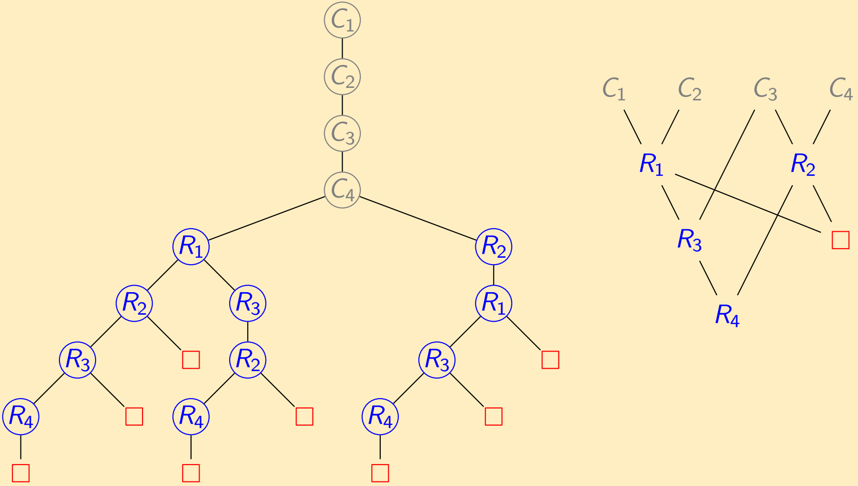
- $C_1$ is the root of $T$
- $C_{i+1}$ is a node of $T$, where $C_i$ is its (direct) predecessor $(1 \leq i \leq n)$
- let $N_p$ be the set of predecessors of the node $N$, plus $N$ itself
- every node $N$ of level $i \geq n$ has, as successors, *all* clauses $R$ such that
  - $R$ is a resolvent of two clauses belonging to $N_p$
  - $R \notin N_p$

## Properties

- every path from $C_1$ to a node $N$ is a derivation of $N$
- every possible derivation is represented by a path in the search tree
- the tree for $\mathcal{C}$ contains all the resolvents for $\mathcal{C}$
- if $\square$ is a resolvent, then there is at least a node labeled with $\square$

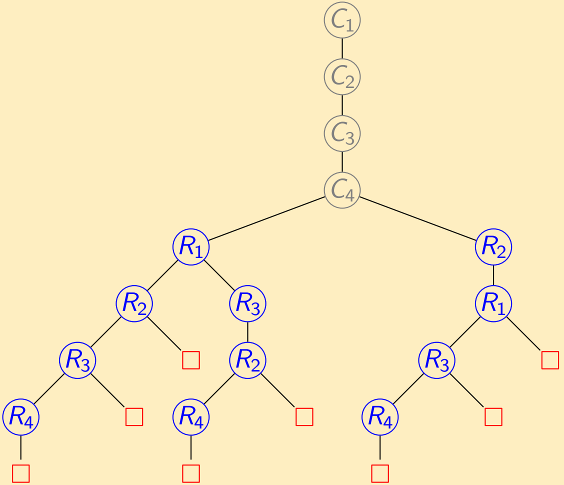
# Search Trees

## Derivations and search trees



# Search Trees

## Derivations and search trees



$C_1\, C_2\, C_3\, C_4\, R_1\, R_2\, R_3\, R_4 \square$

$C_1\, C_2\, C_3\, C_4\, R_1\, R_2\, R_3 \square$

$C_1\, C_2\, C_3\, C_4\, R_1\, R_2 \square$

$C_1\, C_2\, C_3\, C_4\, R_1\, R_3\, R_2\, R_4 \square$

$C_1\, C_2\, C_3\, C_4\, R_1\, R_3\, R_2 \square$

$C_1\, C_2\, C_3\, C_4\, R_2\, R_1\, R_3\, R_4 \square$

$C_1\, C_2\, C_3\, C_4\, R_2\, R_1\, R_3 \square$

$C_1\, C_2\, C_3\, C_4\, R_2\, R_1 \square$

# Search Trees

## Restricted search trees

- refinement strategies make the search simpler by only considering derivations which satisfy a given property
  - i.e., trees with a given shape
- a search tree can be reduced by imposing conditions on the successors of a node $N$, by restricting the clauses $D_i$ and $D_j$ which can produce resolvents starting from $N$

# Linear Resolution

## Linear Derivations

A linear derivation of $C_m$ from $\{C_1, .., C_n\}$ is a sequence $C_1, .., C_n, C_{n+1}, .., C_m$ such that

- $C_{n+1}$ is the resolvent of two clauses of $\{C_1, .., C_n\}$ (*header clauses*)
- for every $i > n + 1$, $C_i$ is the resolvent of $C_{i-1}$ and another clause $C_j$, with $j < i - 1$

## Properties

Linear resolution is *complete*: $UNSAT(\mathcal{C})$ iff there exists a linear refutation of $\mathcal{C}$

- derivations can be restricted to linear derivations
- search trees can be restricted to linear search trees

In a derivation of $C$ from $\mathcal{C}$, it is not necessary to prove all the clauses in $\mathcal{C}$ as a starting point for the refutation (of $\neg C$)

- if a set $\mathcal{C}$ is satisfiable and $\mathcal{C} \cup \neg C$ is not, then there exists a linear refutation starting from $\neg C$

# Input Resolution

## Input derivations

An input derivation of $C_m$ from $\{C_1, .., C_n\}$ is a sequence $C_1, .., C_n, C_{n+1}, .., C_m$ such that

- for every $i > n$, $C_i$ is a resolvent of $C_k \in \{C_1, .., C_n\}$ and another clause $C_j$, $j < i$

## Example

$$C_1 = \neg p(x) \vee q(x) \qquad C_2 = \neg r(x) \vee \neg q(x) \qquad C_3 = r(a)$$
$$C_4 = s(a), \qquad C_5 = \neg s(x) \vee p(x)$$

- input refutation from $C_1$: $\begin{array}{ll} R_1 = \neg p(x) \vee \neg r(x) & (C_1, C_2) \\ R_2 = \neg s(x) \vee \neg r(x) & (R_1, C_5) \\ R_3 = \neg s(a) & (R_2, C_3) \\ R_4 = \Box & (R_3, C_4) \end{array}$

# Input Resolution

## Input derivations

An input derivation of $C_m$ from $\{C_1, .., C_n\}$ is a sequence $C_1, .., C_n, C_{n+1}, .., C_m$ such that

- for every $i > n$, $C_i$ is a resolvent of $C_k \in \{C_1, .., C_n\}$ and another clause $C_j$, $j < i$

## Example

$$C_1 = \neg p(x) \vee q(x) \qquad C_2 = \neg r(x) \vee \neg q(x) \qquad C_3 = r(a)$$
$$C_4 = s(a), \qquad C_5 = \neg s(x) \vee p(x)$$

- input refutation from $C_5$: $\begin{array}{ll} R_1 = p(a) & (C_4, C_5) \\ R_2 = q(a) & (R_1, C_1) \\ R_3 = \neg r(a) & (R_2, C_2) \\ R_4 = \square & (R_3, C_3) \end{array}$

# Input Resolution

**Example:** $C_1 = p \vee q$, $C_2 = \neg q$, $C_3 = r \vee q$, $C_4 = \neg r$

- *input non-linear* refutation from $C_1$:
  $R_1 = p \quad (C_1, C_2)$
  $R_2 = r \quad (C_2, C_3)$
  $R_3 = \Box \quad (R_2, C_4)$
- since $R_1$ is not involved in the rest of the derivation, we can build an *input linear* refutation from the first one:
  $R_1 = r \quad (C_2, C_3)$
  $R_2 = \Box \quad (R_1, C_4)$

# Input Resolution

## Lemma

*Given an input non-linear derivation of $R_m$, it is possible to construct an input linear derivation of $R_m$*

## Proof.

Let $C_1, .., C_n, R_1, .., R_m$ an input non-linear derivation of $R_m$

1. let $R_{k+1}$ $(n+1 \leq k \leq m)$ the first resolvent which is non-linearly derivated
2. $R_{k+1}$ is the resolvent of $C \in \{C_1, .., C_n\}$ and $R_j$ $(1 \leq j < k)$
3. for input resolution, $R_{k+1}$ and $R_k$ cannot resolve with each other
4. for ❸, it is possible to generate two independent derivations
   - $C_1, .., C_n, R_1, .., R_k, ..$ (linear until $R_k$)
   - $C_1, .., C_n, R_1, .., R_j, R_{k+1}, ..$ (linear until $R_{k+1}$)
5. one of these derivations will terminate in $R_m$

# Input Resolution

**Example:** $C_1 = p \vee q$, $C_2 = \neg p \vee q$, $C_3 = r \vee \neg q$, $C_4 = \neg r \vee \neg q$

- *non-input non-linear*:
  $R_1 = q \vee q \quad (C_1,C_2)$
  $R_2 = \neg q \vee \neg q \quad (C_3,C_4)$
  $R_3 = \Box \quad (R_1,R_2)$
- for every non-linear derivation there exists a linear equivalent one:
  $R_1 = q \vee q \quad (C_1,C_2)$
  $R_2 = r \quad (R_1,C_3)$
  $R_3 = \neg q \quad (R_2,C_4)$
  $R_4 = \Box \quad (R_3,R_1)$
- is it possible to find an input derivation for every non-input derivation?

**Input resolution is not complete**

It is *not* possible to say that, for every unsatisfiable set of clauses, there exists an input refutation

# Directed Resolution (Wos-Robinson-Carson, 1965)

## Directed derivations

A directed derivation of $C_m$ from $\{C_1, .., C_n\}$, with a *support set* $S \subset \mathcal{C}$, is a sequence $C_1, .., C_n, C_{n+1}, .., C_m$ such that

- for every $i > n$, $C_i$ is a resolvent of two previous clauses in the sequence, such that at least one of them does *not* belong to $S$
- clauses in $S$ are *support clauses*, while clauses in $\mathcal{C} \setminus S$ are *goal clauses*
- this technique is motivated by the fact that:
  - suppose we want to prove $B$ from $A_1 \wedge .. \wedge A_k$
  - i.e., that $A_1 \wedge .. \wedge A_k \wedge \neg B$ is unsatisfiable
  - in this case, $A_1 \wedge .. \wedge A_k$ is usually satisfiable in itself
  - therefore, it might be wise to avoid resolving two clauses of such set
  - the support set identifies the subset of $\mathcal{C}$ which is supposed to be satisfiable (the result to be proven is not in the support set)

# Directed Resolution (Wos-Robinson-Carson, 1965)

## Example

$$\mathcal{C} = \{C_1 = s \vee t,\ \ C_2 = \neg s \vee p,\ \ C_3 = \neg q \vee r,\ \ C_4 = q \vee \neg p,\ \ C_5 = u \vee \neg r,\ \ C_6 = \neg u,\ \ C_7 = \neg t\}$$

$$S = \{C_1,\ C_2,\ C_3,\ C_4,\ C_5\}$$

| directed | |
|---|---|
| $R_1 = s$ | $(C_1, C_7)$ |
| $R_2 = p$ | $(R_1, C_2)$ |
| $R_3 = q$ | $(R_2, C_4)$ |
| $R_4 = r$ | $(R_3, C_3)$ |
| $R_5 = u$ | $(R_4, C_5)$ |
| $R_6 = \square$ | $(R_5, C_6)$ |

| non-directed | |
|---|---|
| $R_1 = t \vee p$ | $(C_1, C_2)$ |
| $R_2 = p$ | $(R_1, C_7)$ |
| $R_3 = q$ | $(R_2, C_4)$ |
| $R_4 = r$ | $(R_3, C_3)$ |
| $R_5 = u$ | $(R_4, C_5)$ |
| $R_6 = \square$ | $(R_5, C_6)$ |

# Directed Resolution (Wos-Robinson-Carson, 1965)

## Properties

Directed resolution is *complete*: if $UNSAT(\mathcal{C})$ and $S \subset \mathcal{C}$ is satisfiable, then there exists a directed refutation of $\mathcal{C}$ with support set $S$

- this is not so useful if no way to find a satisfiable $S$ is given

## Heuristic for finding $S$

In practice, when trying a refutation of a conclusion from a set of premises, it is reasonable to consider the premises satisfiable

- premises: $S$
- negation of the conclusion (clause form): $\mathcal{C} \setminus S$
- if the premises are inconsistent, then every result can be derived
- yet, otherwise, $\square$ can be derived from negating the conclusion

# Ordered Resolution

## Ordered derivations

An ordered derivation of $C_m$ from $\{C_1, .., C_n\}$ is a sequence $C_1, .., C_n, C_{n+1}, .., C_m$ such that

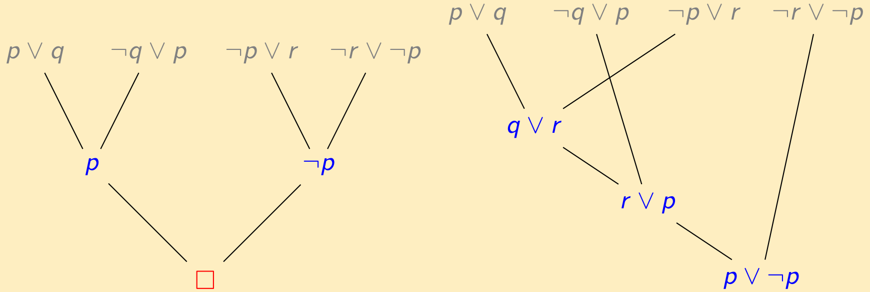
- for every $i > n$, $C_i$ is the resolvent of two previous clauses $A_1 \vee L_{11} \vee .. \vee L_{1p}$ and $\neg A_2 \vee L_{21} \vee .. \vee L_{2q}$, where $A_1$ and $A_2$ are unifiable with $MGU\ \sigma$
- the literals of $C_i$ are ordered as: $(L_{11} \vee .. \vee L_{1p} \vee L_{21} \vee .. \vee L_{2q})\sigma$

# Ordered Resolution

## (Non-)Properties

Ordered resolution is *not* complete

counterexample: $\{p \vee q, \quad \neg q \vee p, \quad \neg p \vee r, \quad \neg r \vee \neg p\}$



# Summary

## Correctness and completeness

- Correctness: $\Box$ can be derived only if $UNSAT(\mathcal{C})$
- Completeness: if $UNSAT(\mathcal{C})$, then $\Box$ can be derived

| | correct | complete |
|---|---|---|
| linear | ✓ | ✓ |
| input | ✓ | no |
| directed | ✓ | ✓(if $SAT(S)$) |
| ordered | ✓ | no |