Herbrand's Theorem

Summary

- Herbrand's Theorem [Chang-Lee 4.5]
- Implementation of the Herbrand Theorem [Chang-Lee Ch. 4.6]

Herbrand's Theorem: Intro

Basic Concepts

- Very important for symbolic logic
- Foundation for many automatic proof procedures
- Closely Related to H-Satisfiability:
  - A set S of clauses is unsatisfiable iff S is false under all the H-Interpretations

H-Satisfiability: Main benefit

Benefit

- To check whether S is inconsistent we do not need to check all possible interpretations
- We can focus on H-Interpretations
- We need to consider only the Herbrand Universe (not all possible domains)

H-Satisfiability not good enough

Problem

- To check whether S is inconsistent we do need to check all H-Intepretations
- H-Interpretations can be infinitely many
- We need a finite procedure!
- Semantic Tree can help to systematically organise all possible H-Interpretations

Herbrand's Theorem: Version I

Theorem (Herbrand's Theorem: Version I)

A set S of clauses is unsatisfiable iff corresponding to every complete semantic tree of S, there is a finite closed semantic tree.

Example

Example (Closed semantic tree for unsatisfiable clauses)

Consider the formula $S = \{\neg P(x) \vee Q(x), P(f(a)), \neg Q(z)\}$
$H = \{a, f(a), f(f(a)), \cdots\}$ $A = \{P(a), Q(a), P(f(a)), Q(f(a)), \cdots\}$

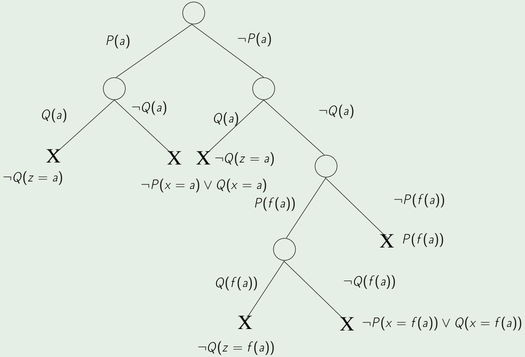


Figure: Closed Semantic tree

Herbrand's Theorem: Version I, proof $\Rightarrow$

Herbrand's Theorem: Version I, $\Rightarrow$.

S unsat. $\Rightarrow$ finite closed semantic tree for every complete tree
Suppose: S is unsat., T complete semantic tree for S.

- Consider each branch B of T: for each branch B there is a complete interpretation I_B.
- S is unsatisfiable: I_B must falsify a ground instance C' of some clause C of S.
- Since C' is finite, then there must be a failure node N_B which is a finite number of links away from the root.
- Since for every branch B we can find a failure node N_B there is a corresponding closed semantic tree T'.
- Since only a finite number of links are connected to each node then T' is finite (using Konig's lemma).

Herbrand's Theorem: Version I, proof $\Leftarrow$

Herbrand's Theorem: Version I, $\Leftarrow$.

Finite closed semantic tree for every complete tree $\Rightarrow$ S unsat.
Suppose: T' finite closed semantic tree corresponding to every complete semantic tree T of S.

- For every branch of T there must be an interpretation that falsify S.
- Every possible interpretation falsifies S, thus S is unsatisfiable.

$\square$

Herbrand's Theorem: Version II

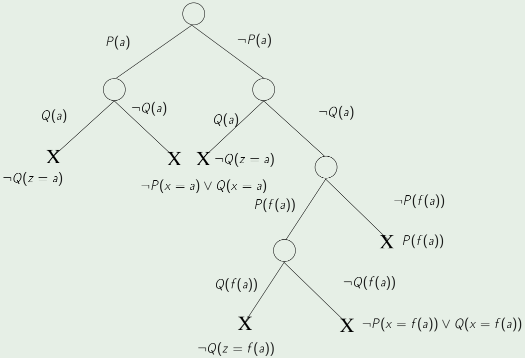
Theorem (Herbrand's Theorem: Version II)

A set S of clauses is unsatisfiable iff there is a finite unsatisfiable set S' of ground instances of clauses S

Example: Building S'

Example (Building S')

Consider the formula $S = \{\neg P(x) \vee Q(x), P(f(a)), \neg Q(z)\}$
$S' =$
$\{\neg Q(f(a)), \neg P(f(a)) \vee Q(f(a)), P(f(a)), \neg Q(a), \neg P(a) \vee Q(a)\}$



Herbrand's Theorem: Version II, proof $\Rightarrow$

Herbrand's Theorem: Version II, $\Rightarrow$.

S unsat. $\Rightarrow$ finite unsatisfiable set S' of ground instances.

Suppose: S is unsat., T a complete semantic tree for S

- Previous version of Herbrand's Theorem $\Rightarrow$ there is a finite closed semantic tree T' corresponding to T.
- Build S' as the set of all ground instances of clauses that are falsified at all the failure nodes of T'
- S' is finite because there are a finite number of failure nodes in T'
- S' is unsatisfiable because it is falsified by every interpretation of S.

$\square$

Herbrand's Theorem: Version II, proof $\Leftarrow$

Herbrand's Theorem: Version II, $\Leftarrow$.

Finite unsat. set S' of ground instances $\Rightarrow$ S unsat.
Suppose: S' finite unsatisfiable set of ground instances of clauses in S.

- Every I interpretation for S must contain one I' for S'.
- If $I' \not\models S'$ then $I \not\models S'$
- Since S' is unsatisfiable then for every $I' \not\models S'$
- Then S' is falsified by every interpretation I of S.
- Therefore S is falsified by every possible interpretation I, and thus S is unsatisfiable.

$\square$

Example

Example

Consider the formula $S = \{P(x), \neg P(a)\}$

- S is unsatisfiable (We can check all H-Interpretations).
- Then by Herbrand's theorem there is a finite set S' of ground instances of clauses in S, which are unsatisfiable.
- $S' = \{P(a), \neg P(a)\}$ is one of these sets.

Note

Given a set of clauses S, unsatisfiable, the set S' of unsatisfiable ground instances of clauses in S in general is not unique. For example $S' = \{P(f(a), \neg P(f(a)) \vee Q(f(a)), \neg Q(f(a))\}$ is a different (smaller) set of unsatisfiable ground clauses for $S = \{P(f(a)), \neg P(x) \vee Q(x), \neg Q(z)\}$

Exercise: building S'

Exercise

1. Consider the unsatisfiable formula $S = \{P(x), Q(x, f(x)) \vee \neg P(x), \neg Q(g(y), z)\}$ Find one of the finite unsatisfiable set S'. [Chang-Lee, Ex. 14 pag.69]
2. Consider the unsatisfiable formula $S = \{P(x, a, g(x, b)), \neg P(f(y), z, g(f(a), b))\}$ Find one of the finite unsatisfiable set S'. [Chang-Lee, Ex. 13 pag.69]

Herbrand's Theorem and semi-decidability of FOL

Herbrand's theorem and semi-decidability of FOL

- Herbrand's Theorem implies semi-decidability of FOL
- Semi-decidable: if S is unsatisfiable the decision procedure will halt proving it, if not, decision procedure may loop forever.

Refutation procedure based on Herbrand's Theorem

Refutation based on ground clauses

Given Herbrand's results to develop a mechanical procedure for testing unsatisfiability of S we need to:

- successively generate sets $S'_0, S'_1, S'_2, \cdots, S'_n, \cdots$ of ground instances of clauses in S
- successively test S'_i for unsatisfiability
- if S is unsatisfiable this procedure will terminate after N iteration finding an unsatisfiable S'_N with N finite.

Gilmore's method

Gilmore 1960

Gilmore built such a procedure:

- S_i' is the set of ground clauses obtained by replacing the variables in S with the constants in the ith level constant set of the Herbrand Universe H_i.
- Each S_i' is a set of ground clauses
- We can use any method for propositional logic to prove unsatisfiability
- Gilmore used the Multiplication method.

Example: Gilmore's Method

Example (Generating S_i')

Consider the formula $S = \{P(x), \neg P(a)\}$

- $H_0 = \{a\}$
- $S_0' = P(a) \wedge \neg P(a)$

Multiplication method

Multiplication Method

For each S_i'

- Reduce S_i' into a DNF
  - $S_i' = D_1 \vee D_2 \vee \cdots \vee D_m$ where each $D_i = L_1 \wedge L_2 \wedge \cdots \wedge L_k$.
  - it is possible to do this by using the distributive property of $\vee$ and $\wedge$.
- Any D_i that contains a complementary pair is removed from S_i'
- If S_i' is empty then it is unsatisfiable and the proof is found.

Example: Gilmore's Method

Example (Prooving unsatisfiability of S_i')

Consider the formula $S = \{P(x), \neg P(a)\}$

- $H_0 = \{a\}$
- $S_0' = P(a) \wedge \neg P(a)$
- S_0' contains a complementary pair
- $\Rightarrow S_0'$ is unsatisfiable.
- $\Rightarrow S$ is unsatisfiable.

Example: Gilmore's Method

Example

Consider the formula $S = \{P(a), \neg P(x) \vee Q(f(x)), \neg Q(f(a))\}$
$H = \{a, f(a), f(f(a)), \cdots\}$

- $H_0 = \{a\}$
- $S_0' = P(a) \wedge (\neg P(a) \vee Q(f(a))) \wedge \neg Q(f(a))$
- DNF for S_0':
  - $((P(a) \wedge \neg P(a)) \vee (P(a) \wedge Q(f(a)))) \wedge (\neg Q(f(a)))$
  - $((P(a) \wedge \neg P(a) \wedge \neg Q(f(a))) \vee (P(a) \wedge Q(f(a)) \wedge \neg Q(f(a)))$
- $\Rightarrow S_0'$ is unsatisfiable
- $\Rightarrow S$ is unsatisfiable.

Example: not termination

Example (Gilmore's might not terminate)

Consider the formula $S = \{P(x, f(x))\}$
$H = \{a, f(a), f(f(a)), \cdots\}$

- $H_0 = \{a\}$
- $P(a, f(a))$ not unsatisfiable
- $H_1 = \{a, f(a)\}$
- $P(f(a), f(f(a)))$ not unsatisfiable
- $H_2 = \{a, f(a), f(f(a))\}$
- $P(f(f(a)), f(f(f(a))))$ not unsatisfiable
- $\cdots$

Computational issues

Gilmore's method inefficiency

- multiplication method to test satisfiability is highly inefficient
- Number of conjunctions explode (for ten two-literals clauses we have 2^{10} conjunctions)
- Many formulas could not be proved unsatisfiable in a reasonable amount of time
- Davis Putnam addressed this computational inefficiency