13

HORN FORMULA.

A horn clause is of the form

$$P_1 \wedge P_2 \ldots P_n \rightarrow P.$$

P_i's and P are variables (positive literals)

Rule based system
↳
$P_1, P_2 \ldots P_n \rightarrow P$
$Q_1, Q_2 \ldots Q_n \rightarrow Q.$

Definition 1.46 A *Horn formula* is a formula ϕ of propositional logic if it can be generated as an instance of H in this grammar:

$$\begin{aligned} P &::= \bot \mid \top \mid p \\ A &::= P \mid P \wedge A \\ C &::= A \rightarrow P \\ H &::= C \mid C \wedge H. \end{aligned} \tag{1.7}$$

We call each instance of C a *Horn clause*.

Horn formulas are conjunctions of Horn clauses. A Horn clause is an implication whose assumption A is a conjunction of propositions of type P and whose conclusion is also of type P. Examples of Horn formulas are

$$(p \wedge q \wedge s \rightarrow p) \wedge (q \wedge r \rightarrow p) \wedge (p \wedge s \rightarrow s)$$
$$(p \wedge q \wedge s \rightarrow \bot) \wedge (q \wedge r \rightarrow p) \wedge (\top \rightarrow s)$$
$$(p_2 \wedge p_3 \wedge p_5 \rightarrow p_{13}) \wedge (\top \rightarrow p_5) \wedge (p_5 \wedge p_{11} \rightarrow \bot).$$

Examples of formulas which are *not* Horn formulas are

$$(p \wedge q \wedge s \rightarrow \neg p) \wedge (q \wedge r \rightarrow p) \wedge (p \wedge s \rightarrow s)$$
$$(p \wedge q \wedge s \rightarrow \bot) \wedge (\neg q \wedge r \rightarrow p) \wedge (\top \rightarrow s)$$
$$(p_2 \wedge p_3 \wedge p_5 \rightarrow p_{13} \wedge p_{27}) \wedge (\top \rightarrow p_5) \wedge (p_5 \wedge p_{11} \rightarrow \bot)$$
$$(p_2 \wedge p_3 \wedge p_5 \rightarrow p_{13} \wedge p_{27}) \wedge (\top \rightarrow p_5) \wedge (p_5 \wedge p_{11} \vee \bot).$$

The first formula is not a Horn formula since $\neg p$, the conclusion of the implication of the first conjunct, is not of type P. The second formula does not qualify since the premise of the implication of the second conjunct, $\neg q \wedge r$, is not a conjunction of atoms, $\bot$, or $\top$. The third formula is not a Horn formula since the conclusion of the implication of the first conjunct, $p_{13} \wedge p_{27}$, is not of type P. The fourth formula clearly is not a Horn formula since it is not a conjunction of implications.

The algorithm we propose for deciding the satisfiability of a Horn formula ϕ maintains a list of all occurrences of type P in ϕ and proceeds like this:

ALGORITHM

function HORN (ϕ):
/* precondition: ϕ is a Horn formula */
/* postcondition: HORN (ϕ) decides the satisfiability for ϕ */
begin function
    mark all occurrences of $\top$ in ϕ;
    while there is a conjunct $P_1 \wedge P_2 \wedge \cdots \wedge P_{k_i} \to P'$ of ϕ
        such that all P_j are marked but P' isn't **do**
        mark P'
    end while
    if $\perp$ is marked **then return** 'unsatisfiable' **else return** 'satisfiable'
end function

Correctness

> A variable or $\{\perp, \top\}$ is marked only if every satisfying assignment of the input needs to be marked $\top$.

> When the algo outputs satisfiable, the set of marked variables, assign true & remaining set of variables assigned false, is a satisfying assignment

function HORN (ϕ):
/* precondition: ϕ is a Horn formula */
/* postcondition: HORN (ϕ) decides the satisfiability for ϕ */
begin function
    mark all occurrences of $\top$ in ϕ;
    while there is a conjunct $P_1 \wedge P_2 \wedge \cdots \wedge P_{k_i} \to P'$ of ϕ
        such that all P_j are marked but P' isn't **do**
        mark P'
    end while
    if $\perp$ is marked **then return** 'unsatisfiable' **else return** 'satisfiable'
end function

We need to make sure that this algorithm terminates on all Horn formulas ϕ as input and that its output (= its decision) is always correct.

Theorem 1.47 *The algorithm* HORN *is correct for the satisfiability decision problem of Horn formulas and has no more than $n + 1$ cycles in its while-statement if n is the number of atoms in ϕ. In particular,* HORN *always terminates on correct input.*

PROOF: Let us first consider the question of program termination. Notice that entering the body of the while-statement has the effect of marking an unmarked P which is not $\top$. Since this marking applies to all occurrences of P in ϕ, the while-statement can have at most one more cycle than there are atoms in ϕ.

Since we guaranteed termination, it suffices to show that the answers given by the algorithm HORN are always correct. To that end, it helps to reveal the functional role of those markings. Essentially, marking a P means that that P has got to be true if the formula ϕ is ever going to be satisfiable. We use mathematical induction to show that

'All marked P are true for all valuations in which ϕ evaluates to T.' (1.8)

holds after any number of executions of the body of the while-statement above. The base case, zero executions, is when the while-statement has not yet been entered but we already and only marked all occurrences of $\top$. Since $\top$ must be true in all valuations, (1.8) follows.

In the inductive step, we assume that (1.8) holds after k cycles of the while-statement. Then we need to show that same assertion for all marked P after $k + 1$ cycles. If we enter the $(k + 1)$th cycle, the condition of the while-statement is certainly true. Thus, there exists a conjunct $P_1 \wedge P_2 \wedge \cdots \wedge P_{k_i} \to P'$ of ϕ such that all P_j are marked. Let v be any valuation in which ϕ is true. By our induction hypothesis, we know that all P_j and therefore $P_1 \wedge P_2 \wedge \cdots \wedge P_{k_i}$ have to be true in v as well. The conjunct $P_1 \wedge P_2 \wedge \cdots \wedge P_{k_i} \to P'$ of ϕ has be to true in v, too, from which we infer that P' has to be true in v.

By mathematical induction, we therefore secured that (1.8) holds no matter how many cycles that while-statement went through.

Finally, we need to make sure that the if-statement above always renders correct replies. First, if $\perp$ is marked, then there has to be some conjunct $P_1 \wedge P_2 \wedge \cdots \wedge P_{k_i} \to \perp$ of ϕ such that all P_i are marked as well. By (1.8) that conjunct of ϕ evaluates to $\mathtt{T} \to \mathtt{F} = \mathtt{F}$ whenever ϕ is true. As this is impossible the reply 'unsatisfiable' is correct. Second, if $\perp$ is not marked, we simply assign T to all marked atoms and F to all unmarked atoms and use proof by contradiction to show that ϕ has to be true with respect to that valuation.

If ϕ is *not* true under that valuation, it must make one of its principal conjuncts $P_1 \wedge P_2 \wedge \cdots \wedge P_{k_i} \to P'$ false. By the semantics of implication this can only mean that all P_j are true and P' is false. By the definition of our valuation, we then infer that all P_j are marked, so $P_1 \wedge P_2 \wedge \cdots \wedge P_{k_i} \to P'$ is a conjunct of ϕ that would have been dealt with in one of the cycles of the while-statement and so P' is marked, too. Since $\perp$ is not marked, P' has to be $\top$ or some atom q. In any event, the conjunct is then true by (1.8), a contradiction $\square$

Note that the proof by contradiction employed in the last proof was not really needed. It just made the argument seem more natural to us. The literature is full of such examples where one uses proof by contradiction more out of psychological than proof-theoretical necessity.

1.6 SAT solvers

The marking algorithm for Horn formulas computes marks as constraints on all valuations that can make a formule true. By (1.8), all marked atoms have to be true for any such valuation. We can extend this idea to general formulas ϕ by computing constraints saying which subformulas of ϕ require a certain truth value for all valuations that make ϕ true:

'All marked subformulas evaluate to their mark value
for all valuations in which ϕ evaluates to T.' (1.9)

In that way, marking atomic formulas generalizes to marking subformulas; and 'true' marks generalize into 'true' and 'false' marks. At the same