# Horn Sentences (1)

- In previous slides, we have seen how to convert any sentence without quantifiers into CNF.
- A literal can be **positive** or **negative** depending on whether it is an atomic sentence or the negation of it (P, $\neg$P).
- A Horn Sentence (names after Alfred Horn) is a sentence in CNF where every disjunction of literals contains at most one positive literal.



# Horn Sentences (2)

- None of the following sentences is a Horn sentence:
  - $\neg$Home(Claire) $\wedge$ (Home(Max) $\vee$ Happy(Carl))
  - (Home(Claire) $\vee$ Home(Max) $\vee$ $\neg$Happy(Claire)) $\wedge$ $\neg$Happy(Carl)
- The first is not a Horn sentence because the second disjunction contains two positive literals.
- The second is not because the first disjunction contains two positive literals too.

# Horn Sentences (3)

- All of the following are Horn sentences:
  - $\neg$Home(Claire) $\wedge$ ($\neg$Home(Max) $\vee$ Happy(Carl))
  - Home(Claire) $\wedge$ Home(Max) $\wedge$ $\neg$Home(Carl))
  - Home(Claire) $\vee$ $\neg$Home(Max) $\vee$ $\neg$Home(Carl))
- All conjunctions have at most one positive literal in the constituent disjunctions.



# Horn Sentences (4)

- We have considered the truth table method to determine whether a sentence is satisfiable or not. The problem is that this technique is completely mechanical, prone to error and consumes a lot of resources (consider the truth table for 50 atomic sentences).
- For Horn sentences we can check for satifiability in one row. This method is known as the **satisfaction algorithm for Horn sentences**.

# Satisfaction Algorithm (1)

- Consider the sentence **S:**
  `Home(Claire)` ∧ ¬`Home(Max)` ∧
  (`Home(Max)` ∨ ¬(`Home(Claire)`))
- Let's abbreviate `Home(Claire)` to **C** and to `Home(Max)` **M**.
- **Write the table**:

| **C** | **M** | **C ∧ ¬M ∧ (M ∨ ¬C)** |
|---|---|---|
| | | |



# Satisfaction Algorithm (2)

- **Find is some atomic sentences are themselves conjuncts in S and write True beneath it**. C is atomic and a conjunct.

| **C** | **M** | **C ∧ ¬M ∧ (M ∨ ¬C)** |
|---|---|---|
| T | | |

- **Fill as much as possible in the RHS**.

| **C** | **M** | **C ∧ ¬M ∧ (M ∨ ¬C)** |
|---|---|---|
| T | | **T** **F** |

# Satisfaction Algorithm (3)

- If we look at the last conjunct we see that if this must resolve to **T** then **M** must be **T**.

| C | M | C ∧ ¬M ∧ (M ∨ ¬C) |
|---|---|---|
| T | T | T T F |

- But this will make the second conjunct becomes false, rendering the whole sentence unsatisfiable.

| C | M | C ∧ ¬M ∧ (M ∨ ¬C) |
|---|---|---|
| T | T | T F T F |



# Satisfaction Algorithm in General

1. **S** is a Horn sentence.
2. Start out as if building a truth table but just write the header.
3. See is any atomic sentences are themselves conjuncts of S. Write **T** beneath them in the reference column.
4. Is some are true, fill in as much as possible of the RHS. For example if you wrote **T** under an atomic sentence **A**, you would write **F** under any ¬**A**.
5. Continue until there is no other way to proceed.
6. If you are forced to assign **F** to one of the conjuncts then the sentence is not satisfiable.