**Implementation of the Candidate-Elimination**

To understand the implementation, let us try to implement it to a smaller data set with a bunch of examples to decide if a person wants to go for a walk.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Weather | Temperature | Humidity | Wind | Water | Forecast | EnjoySpot |
| Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| Sunny | Warm | High | Strong | Warm | Same | Yes |
| Rainy | Cold | High | Strong | Warm | Change | No |
| Sunny | Warm | High | Strong | Cool | Chnage | Yes |

We want a boolean function that would be **true** for all the examples for which EnjoySport = **Yes**. Each boolean function defined over the space of the tuples of <Time, Weather, Temperature, Company, Humidity, Wind> is a candidate hypothesis for our target concept “days when Aldo enjoys his favorite water sport”. Since that’s an infinite space to search, we restrict it by choosing an appropriate hypothesis representation.

For example, we may consider only the hypotheses that are conjunctions of the terms A = v, where A of <Time, Weather, Temperature, Company, Humidity, Wind> and v is:

a single value A can take or one of two special symbols:

?, which means that any admissible value of A is acceptable

θ, which means that A should have no value

For instance, the hypothesis (Time = Rainy) ˄ (Weather = Normal) can be represented as follows:

<Rainy, Normal, ?, ? ,? ?>

and would correspond to the boolean function:

There may be multiple hypotheses that fully capture the positive objects in our data. But, we’re interested in those also consistent with the **negative** ones. All the functions consistent with **positive and negative** objects (which means they classify them correctly) constitute the version space of the target concept.

