Machine Learning

Version Spaces Learning

Dr. V. Vijayarajan, Associate Professor,
SCOPE

Introduction:

◎ Very many approaches to machine learning:

➔ Neural Net approaches

➔ Symbolic approaches:

- version spaces
- decision trees
- knowledge discovery
- data mining
- speed up learning
- inductive learning
- …

Current best learning algorithm

function Current-Best-Learning(*examples)* **returns** hypothesis *H*

H := hypothesis consistent with first example

for each remaining example *e* in *examples* **do**

if *e* is false positive for *H* **then**

H := **choose** a specialization of *H* consistent with *examples*

else if *e* is false negative for *H* **then**

H := **choose** a generalization of *H* consistent with *examples*

if no consistent generalization/specialization found **then fail**

end

return *H*

Note: choose operations are nondeterministic and indicate backtracking points.

Definitions

- Positive example: an instance of the hypothesis
- Negative example: not an instance of the hypothesis
- False negative example: the hypothesis predicts it should be a negative example but it is in fact positive
- False positive example: should be positive but it is actually negative.

Specialization and generalization

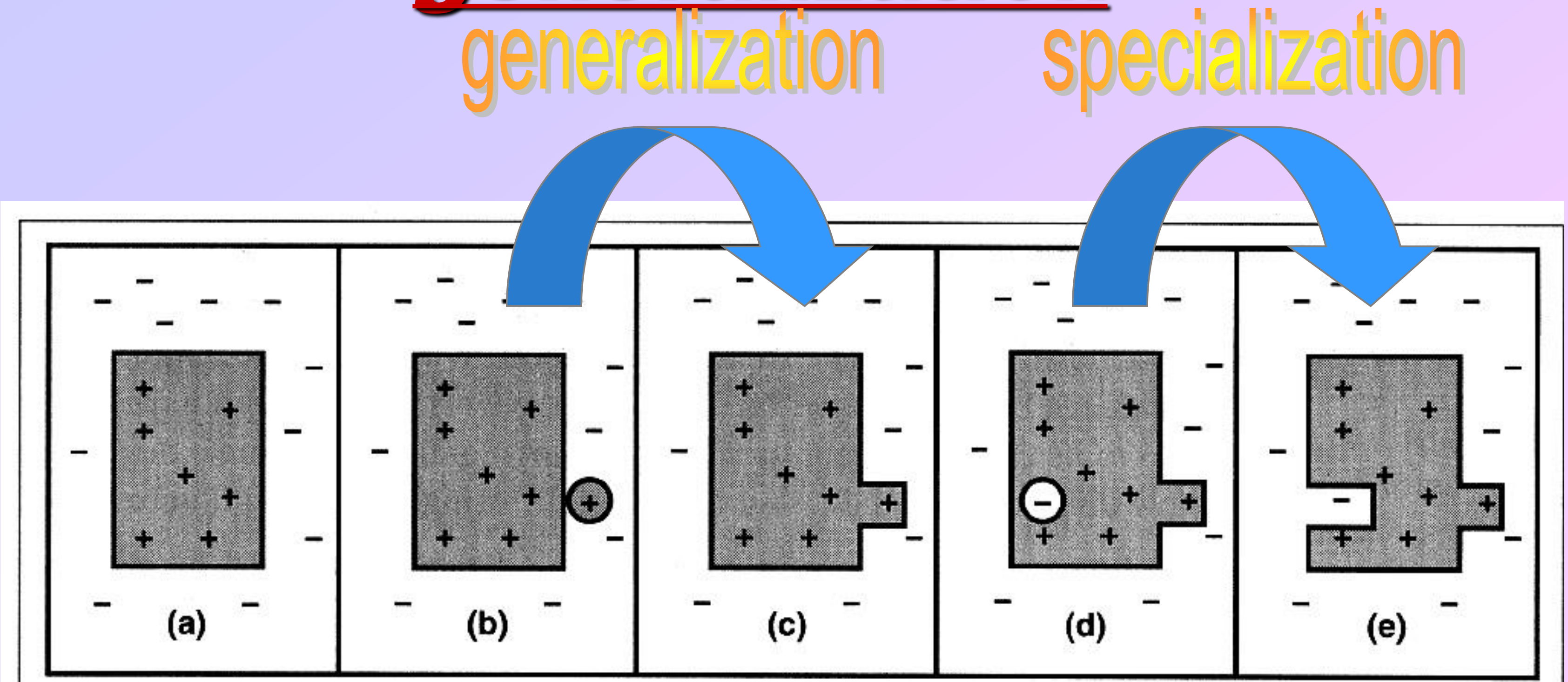


Figure 18.10 (a) A consistent hypothesis. (b) A false negative. (c) The hypothesis is generalized. (d) A false positive. (e) The hypothesis is specialized.

+ indicates positive examples

– indicates negative examples

Circled + and – indicates the example being added

How to Generalize

a) Replacing Constants with Variables:
   Object(Animal,Bird) → Object (X,Bird)

b) Dropping Conjuncts:
   Object(Animal,Bird) & Feature(Animal,Wings)
   → Object(Animal,Bird)

c) Adding Disjuncts:
   Feature(Animal,Feathers) → Feature(Animal,Feathers) v
   Feature(Animal,Fly)

d) Generalizing Terms:
   Feature(Bird,Wings) → Feature(Bird,Primary-Feature)

http://www.pitt.edu/~suthers/infsci1054/8.html

How to Specialize

a) Replacing Variables with Constants:
   Object (X, Bird) → Object(Animal, Bird)

b) Adding Conjuncts:
   Object(Animal,Bird) → Object(Animal,Bird) & Feature(Animal,Wings)

c) Dropping Disjuncts:
   Feature(Animal,Feathers) v Feature(Animal,Fly) → Feature(Animal,Fly)

d) Specializing Terms:
   Feature(Bird,Primary-Feature) → Feature(Bird,Wings)

http://www.pitt.edu/~suthers/infsci1054/8.html

Version Spaces

A concept learning technique based on refining models of the world.

Concept Learning

- **Example:**
  - ➔ a student has the following observations about having an allergic reaction after meals:

Restaurant	Meal	Day	Cost	Reaction	
Alma 3	breakfast	Friday	cheap	Yes	+
De Moete	lunch	Friday	expensive	No	-
Alma 3	lunch	Saturday	cheap	Yes	+
Sedes	breakfast	Sunday	cheap	No	-
Alma 3	breakfast	Sunday	expensive	No	-

➔ Concept to learn: under which circumstances do I get an allergic reaction after meals ??

In general

- There is a set of all possible events:
  - Example:

Restaurant		Meal		Day		Cost		
3	X	3	X	7	X	2	=	126

- There is a boolean function (implicitly) defined on this set.
  - Example:

Reaction: Restaurant X Meal X Day X Cost --> Bool

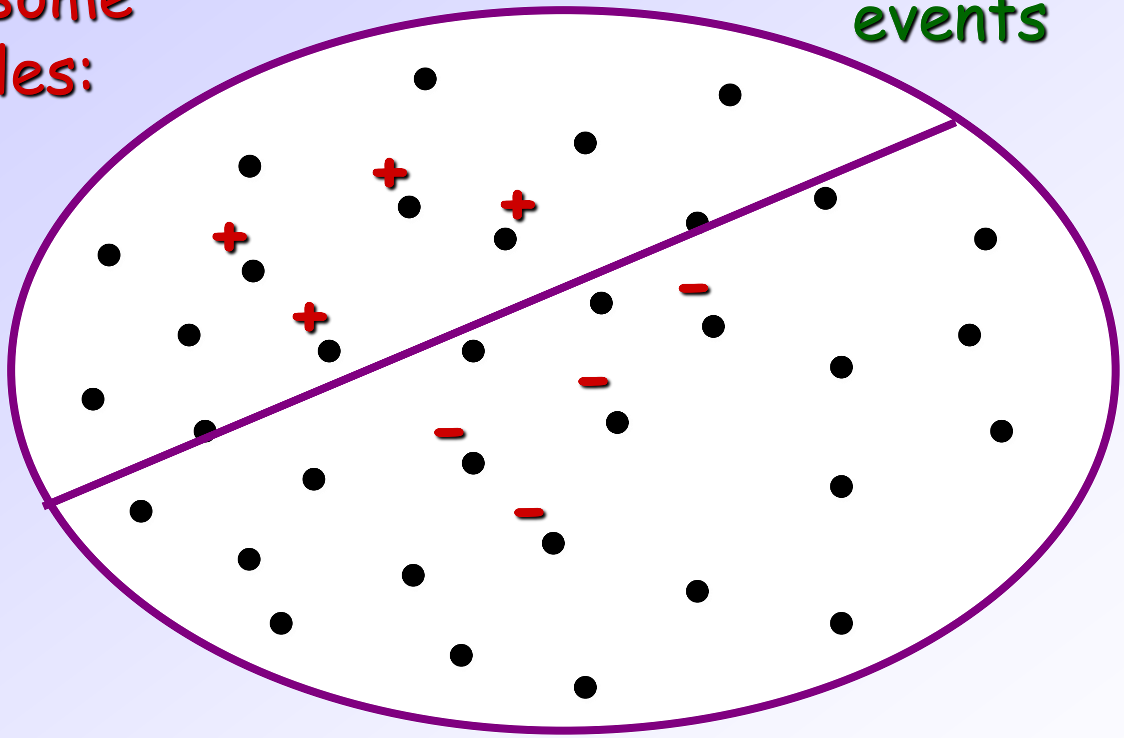
- We have the value of this function for SOME examples only.

Find an inductive 'guess' of the concept, that covers all the examples!

Pictured:

Given some examples:

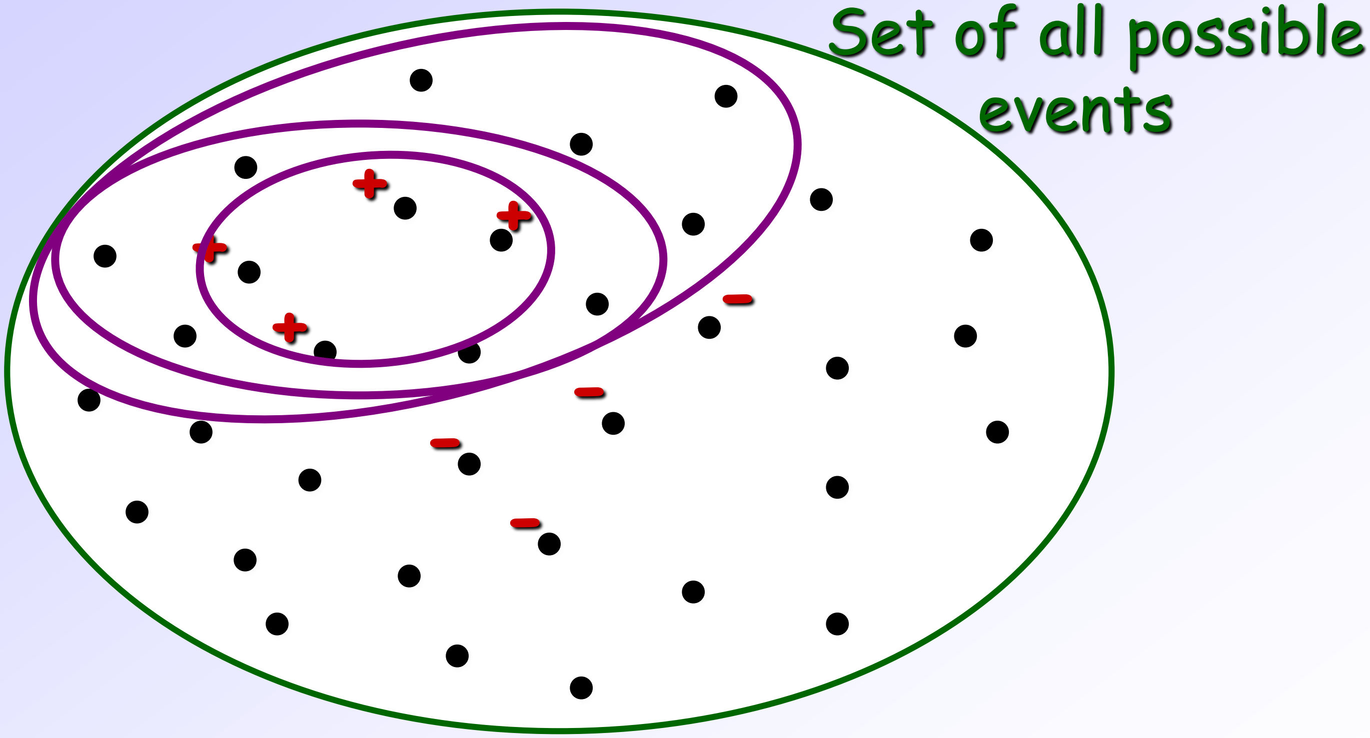
Set of all possible events



Find a concept that covers all the positive examples and none of the negative ones!

Non-determinism

- Many different ways to solve this!



- How to choose ?

An obvious, but bad choice:

Restaurant	Meal	Day	Cost	Reaction	
Alma 3	breakfast	Friday	cheap	Yes	+
De Moete	lunch	Friday	expensive	No	-
Alma 3	lunch	Saturday	cheap	Yes	+
Sedes	breakfast	Sunday	cheap	No	-
Alma 3	breakfast	Sunday	expensive	No	-

- **The concept IS:**

→ Alma 3 and breakfast and Friday and cheap

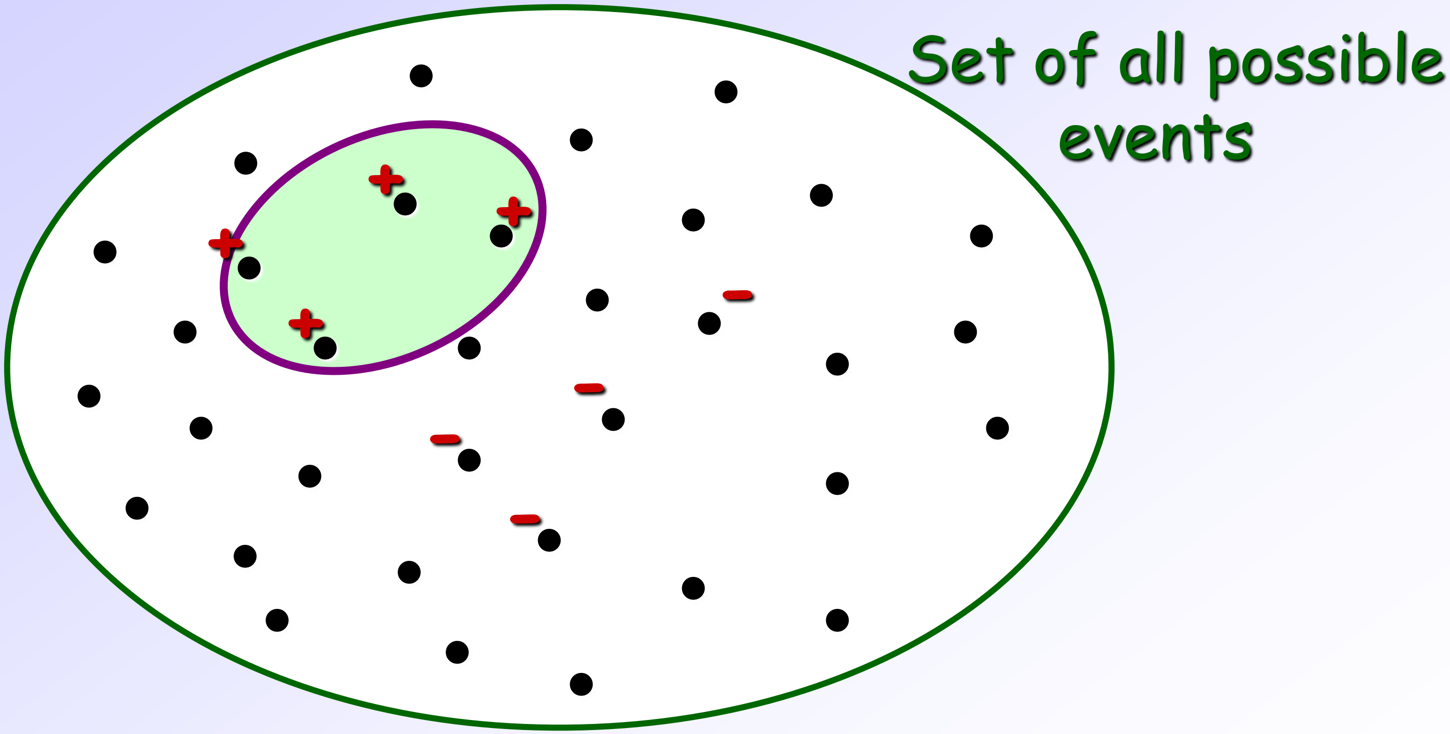
OR

→ Alma 3 and lunch and Saturday and cheap

Does NOT generalize the examples any !!

Pictured:

◎ Only the positive examples are positive !



Equally bad is:

Restaurant	Meal	Day	Cost	Reaction	
Alma 3	breakfast	Friday	cheap	Yes	+
De Moete	lunch	Friday	expensive	No	-
Alma 3	lunch	Saturday	cheap	Yes	+
Sedes	breakfast	Sunday	cheap	No	-
Alma 3	breakfast	Sunday	expensive	No	-

- The concept is anything EXCEPT:

→ De Moete and lunch and Friday and expensive

AND

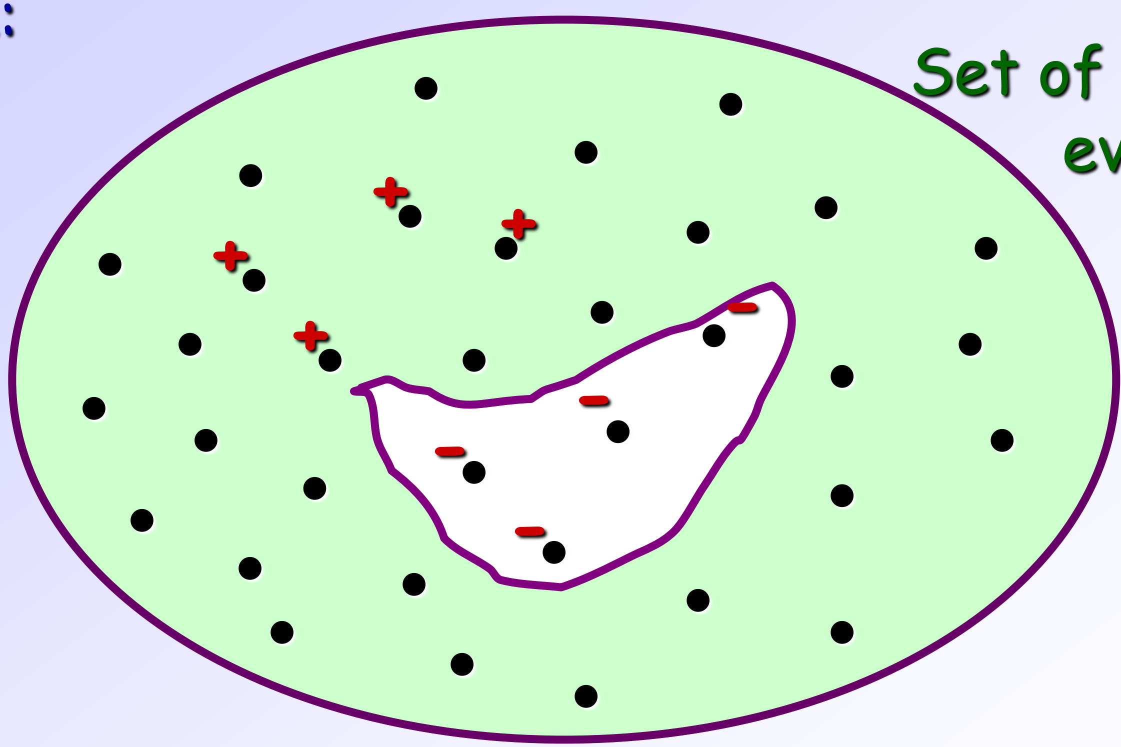
→ Sedes and breakfast and Sunday and cheap

AND

→ Alma 3 and breakfast and Sunday and expensive

Pictured:

- Everything except the negative examples are positive:



Solution: fix a language of hypotheses:

- We introduce a fix language of concept descriptions.

= hypothesis space

- The concept can only be identified as being one of the hypotheses in this language

→ avoids the problem of having 'useless' conclusions
→ forces some generalization/induction to cover more than just the given examples.

Reaction - Example:

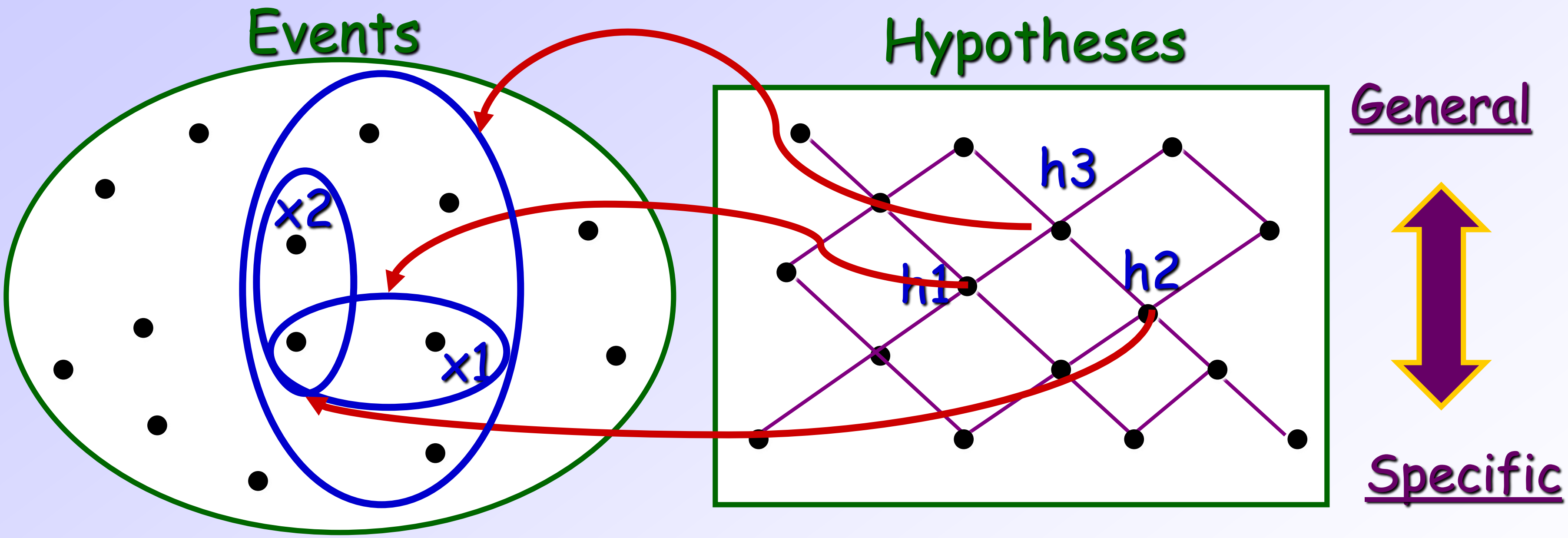
Restaurant	Meal	Day	Cost	Reaction	
Alma 3	breakfast	Friday	cheap	Yes	+
De Moete	lunch	Friday	expensive	No	-
Alma 3	lunch	Saturday	cheap	Yes	+
Sedes	breakfast	Sunday	cheap	No	-
Alma 3	breakfast	Sunday	expensive	No	-

- Every hypothesis is a 4-tuple:

  - ➔ most general hypothesis: [?, ?, ?, ?]
  - ➔ maximally specific: ex.: [Sedes, lunch, Monday, cheap]
  - ➔ combinations of ? and values are allowed: ex.: [De Moete, ?, ?, expensive]
  - ➔ or [?, lunch, ?, ?]

- One more hypothesis: $\perp$ (bottom = denotes no example)

Hypotheses relate to sets of possible events



x1 =
< Alma 3, lunch, Monday, expensive>
x2 =
< Sedes, lunch, Sunday, cheap>

h1 = [?, lunch, Monday, ?]
h2 = [?, lunch, ?, cheap]
h3 = [?, lunch, ?, ?]

Expressive power of this hypothesis language:

- Conjunctions of explicit, individual properties

- Examples:

  - [?, lunch, Monday, ?] : Meal = lunch $\wedge$ Day = Monday
  - [?, lunch, ?, cheap] : Meal = lunch $\wedge$ Cost = cheap
  - [?, lunch, ?, ?] : Meal = lunch

- In addition to the 2 special hypotheses:

  - [?, ?, ?, ?] : anything
  - $\perp$: nothing

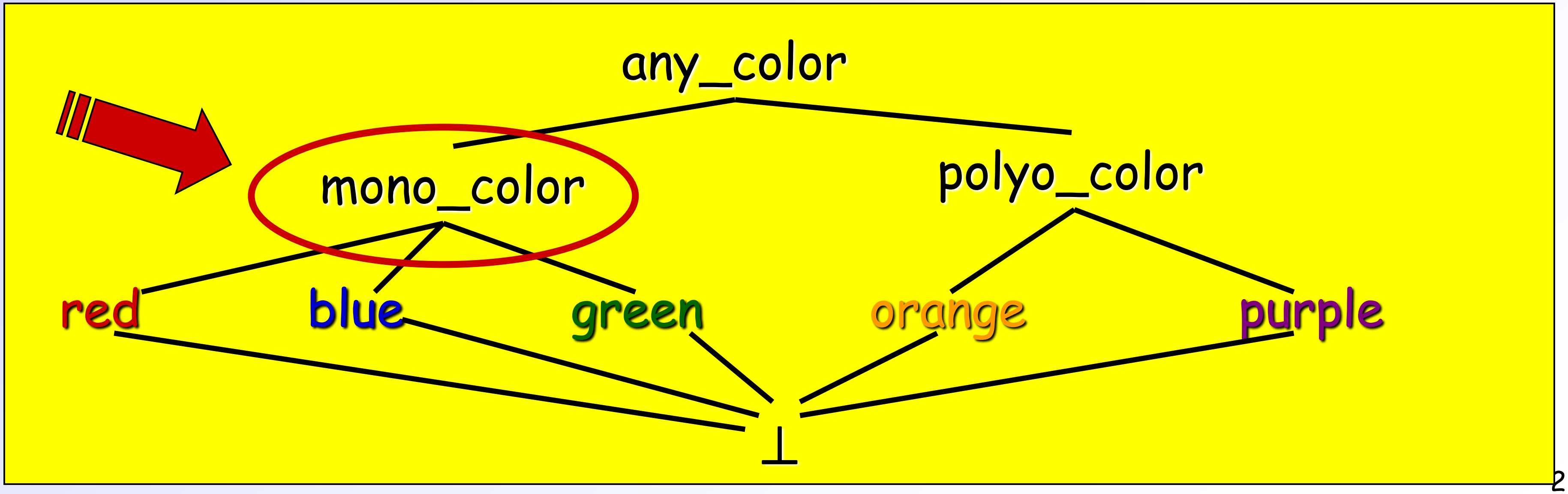
Other languages of hypotheses are allowed

- Example: identify the color of a given sequence of colored objects.

the examples:

red : +
purple : -
blue : +

- A useful language of hypotheses:



Important about hypothesis languages:

- They should have a specific <-> general ordering.

  Corresponding to the set-inclusion of the events they cover.

Defining Concept Learning:

Given:

A set X of possible events :

- Ex.: Eat-events: <Restaurant, Meal, Day, Cost>

An (unknown) target function c: X -> {-, +}

- Ex.: Reaction: Eat-events -> { -, +}

A language of hypotheses H

- Ex.: conjunctions: [?, lunch, Monday, ?]

A set of training examples D, with their value under c

- Ex.: (<Alma 3, breakfast,Friday,cheap>,+) , ...

Find:

A hypothesis h in H such that for all x in D:

x is covered by h $\Leftrightarrow$ c(x) = +

The inductive learning hypothesis:

If a hypothesis approximates the target function well over a sufficiently large number of examples,

then the hypothesis will also approximate the target function well on other unobserved examples.

Find-S: a naïve algorithm

Initialize: $h := \perp$

For each positive training example x in D **Do:**

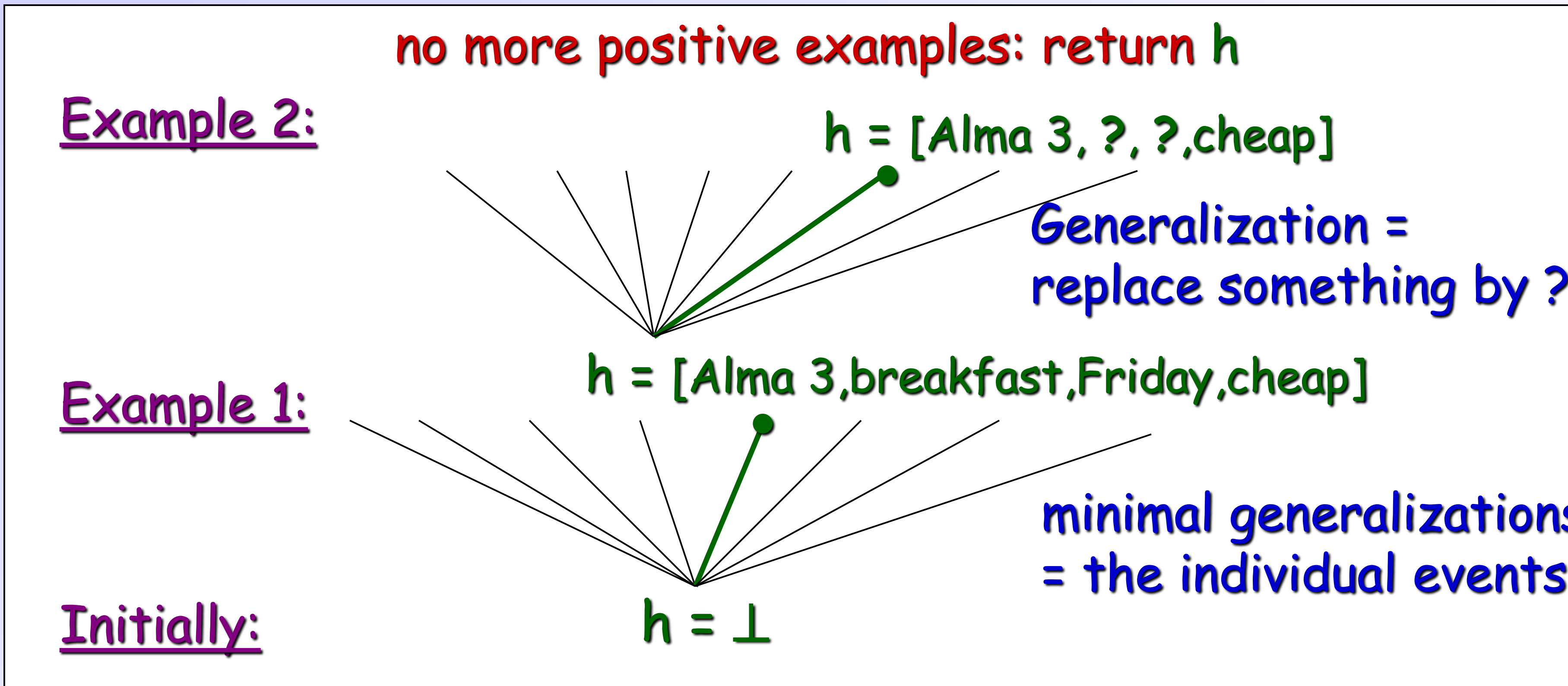
  If h does not cover x:

    Replace h by a minimal generalization of h that covers x

Return h

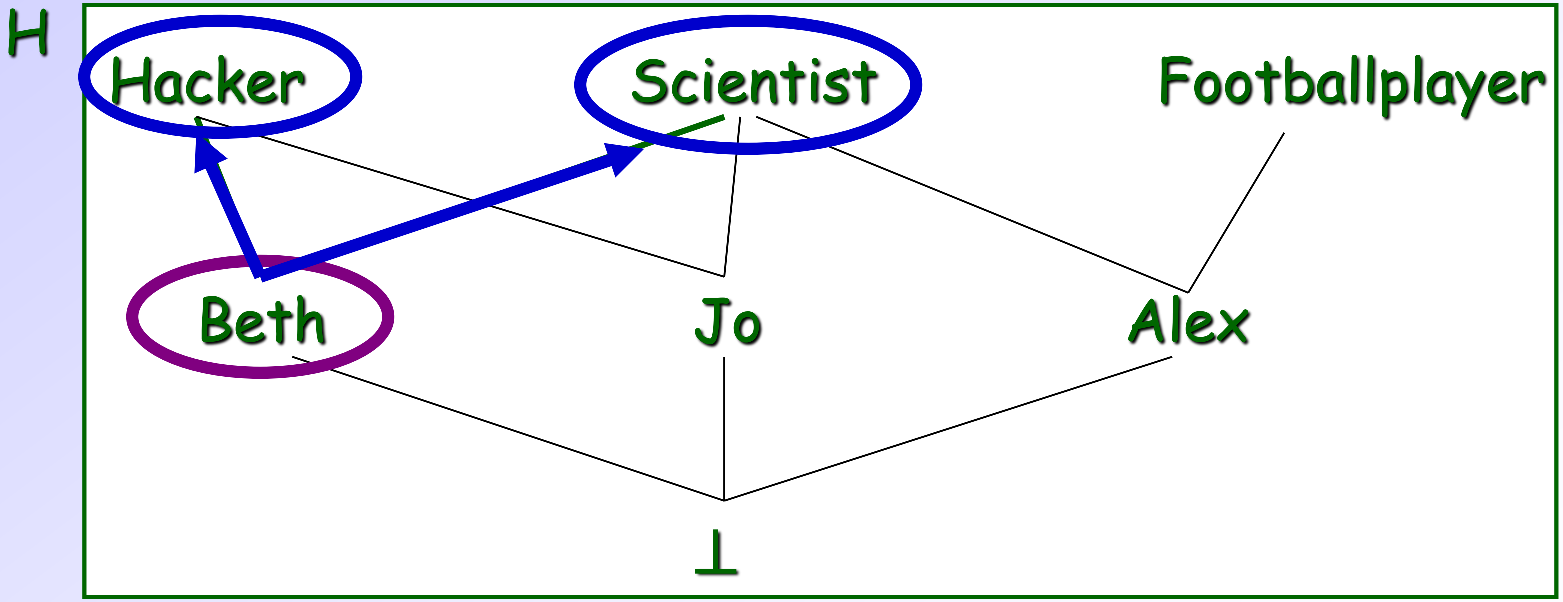
Reaction example:

Restaurant	Meal	Day	Cost	Reaction
Alma 3	breakfast	Friday	cheap	Yes +
De Moete	lunch	Friday	expensive	No −
Alma 3	lunch	Saturday	cheap	Yes +
Sedes	breakfast	Sunday	cheap	No −
Alma 3	breakfast	Sunday	expensive	No −



Properties of Find-S

- Non-deterministic:
  - ➔ Depending on H, there maybe several minimal generalizations:

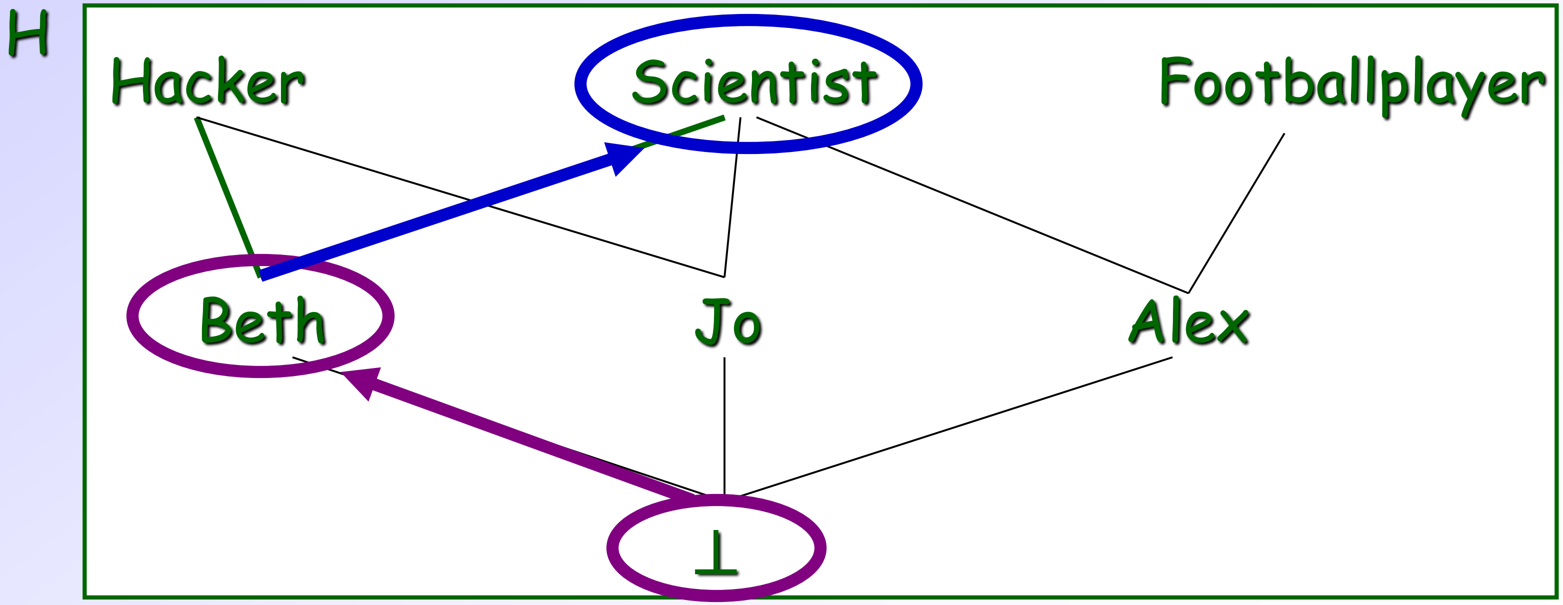


Beth can be minimally generalized in 2 ways to include a new example Jo.

Properties of Find-S (2)

- May pick incorrect hypothesis (w.r.t. the negative examples):

D :	Beth	+
	Alex	-
	Jo	+



Is a wrong solution

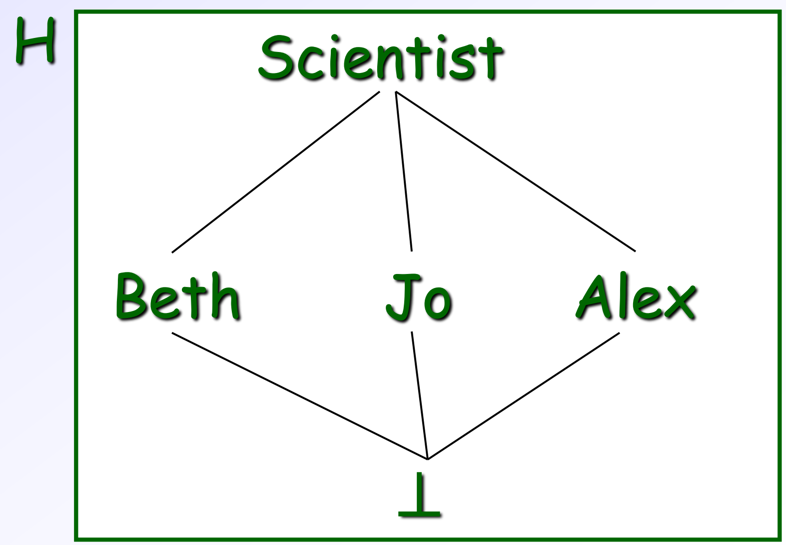
Properties of Find-S (3)

- Cannot detect inconsistency of the training data:

D :	Beth	+
	Jo	+
	Beth	-

- Nor inability of the language H to learn the concept:

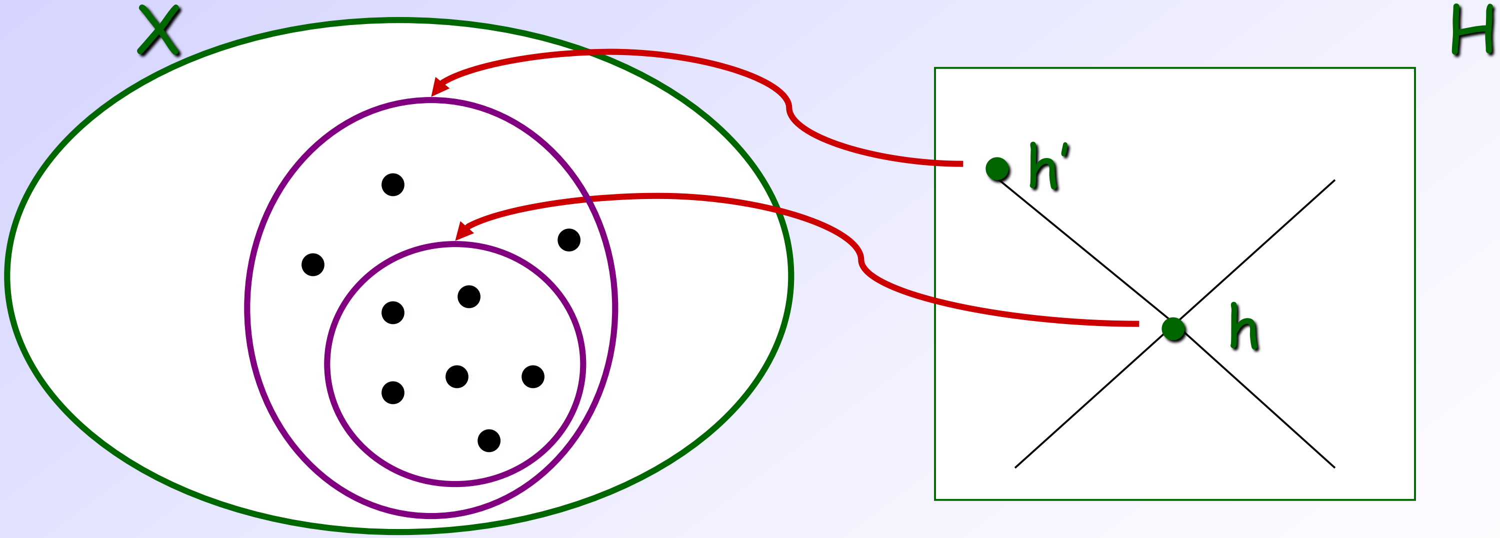
D :	Beth	+
	Alex	-
	Jo	+



Nice about Find-S:

◎ It doesn't have to remember previous examples !

➔If the previous h already covered all previous examples, then a minimal generalization h' will too !



If h already covered the 20 first examples, then h' will as well

Dual Find-S:

Initialize: h := [?, ?, .., ?]

For each negative training example x in D Do:

If h does cover x:

Replace h by a minimal specialization of h that does not cover x

Return h

Reaction example:

Restaurant	Meal	Day	Cost	Reaction
Alma 3	breakfast	Friday	cheap	Yes +
De Moete	lunch	Friday	expensive	No −
Alma 3	lunch	Saturday	cheap	Yes +
Sedes	breakfast	Sunday	cheap	No −
Alma 3	breakfast	Sunday	expensive	No −

Initially: h = [?, ?, ?, ?]

Example 1: h= [?, breakfast, ?, ?]

Example 2: h= [Alma 3, breakfast, ?, ?]

Example 3: h= [Alma 3, breakfast, ?, cheap]

Version Spaces: the idea:

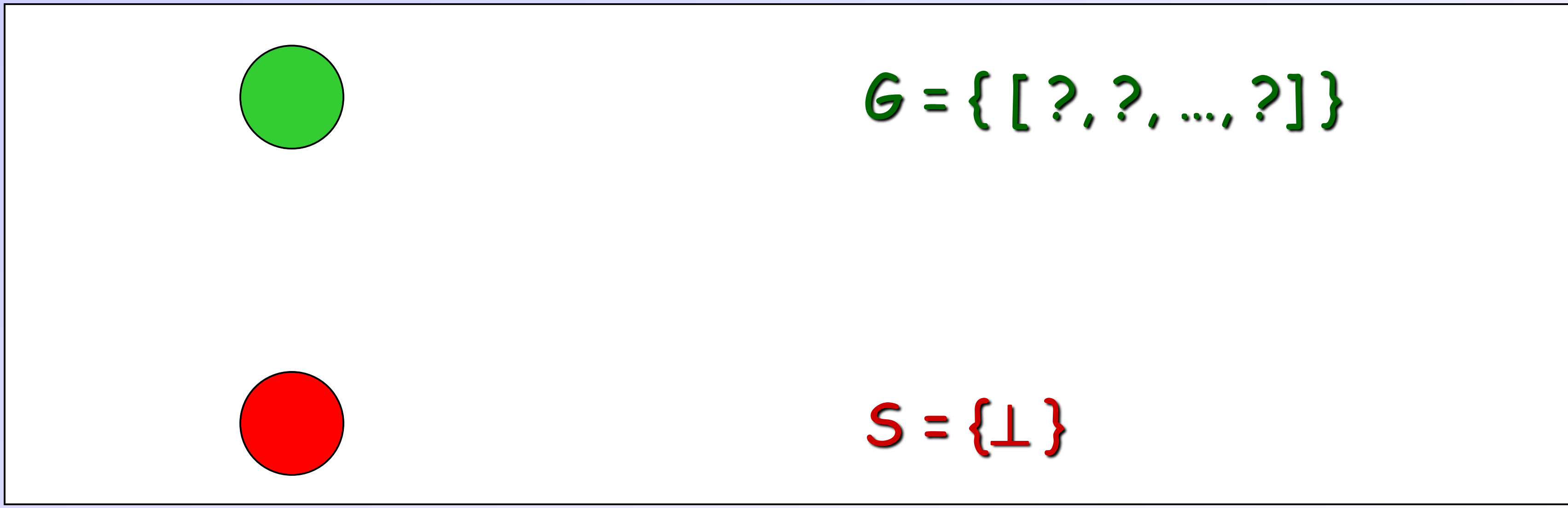
- Perform both Find-S and Dual Find-S:
  - ➔ Find-S deals with positive examples
  - ➔ Dual Find-S deals with negative examples

- BUT:

  do NOT select 1 minimal generalization or specialization at each step,

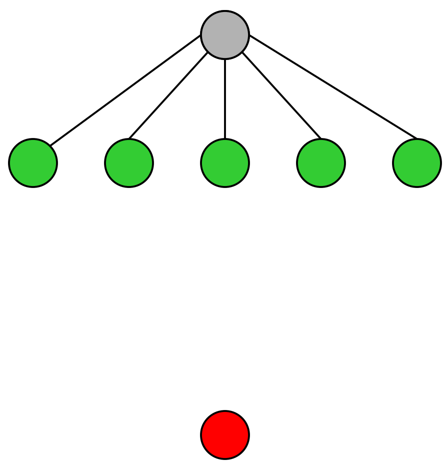
  but keep track of ALL minimal generalizations or specializations

Version spaces: initialization



- The version spaces G and S are initialized to be the smallest and the largest hypotheses only.

Negative examples:



~~G = {[?, ?, …, ?]}~~

G = {h1, h2, …, hn}

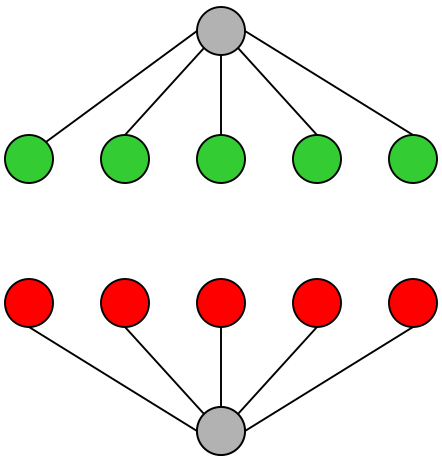
S = {⊥}

- Replace the top hypothesis by ALL minimal specializations that DO NOT cover the negative example.

Invariant: only the hypotheses more specific than the ones of G are still possible:

they don't cover the negative example

Positive examples:



G = {h1, h2, …, hn}

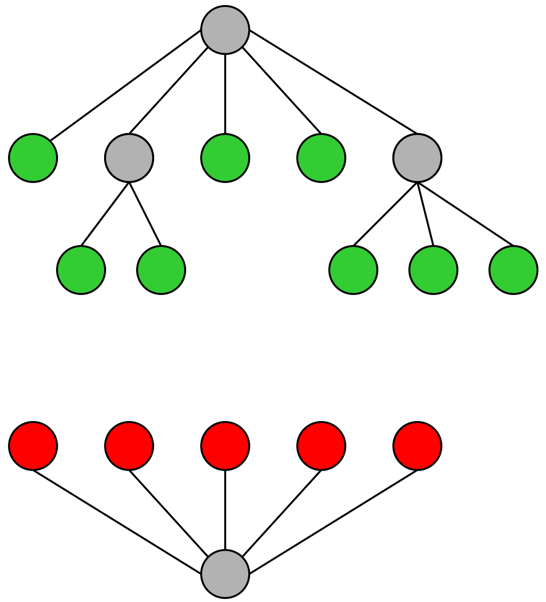
S = {h1’, h2’, …,hm’}

~~S = {⊥}~~

- Replace the bottom hypothesis by ALL minimal generalizations that DO cover the positive example.

Invariant: only the hypotheses more general than the ones of S are still possible:
they do cover the positive example

Later: negative examples



G = ~~{h1, h2, …, hn}~~

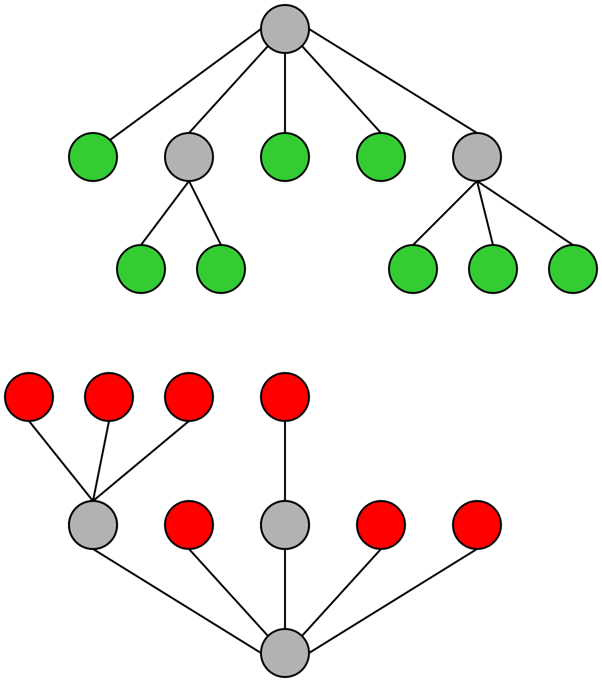
G = {h1, h21, h22, …, hn}

S = {h1', h2', …,hm'}

- Replace the all hypotheses in G that cover a next negative example by ALL minimal specializations that DO NOT cover the negative example.

Invariant: only the hypotheses more specific than the ones of G are still possible:
they don't cover the negative example

Later: positive examples



G = {h1, h21, h22, …, hn}

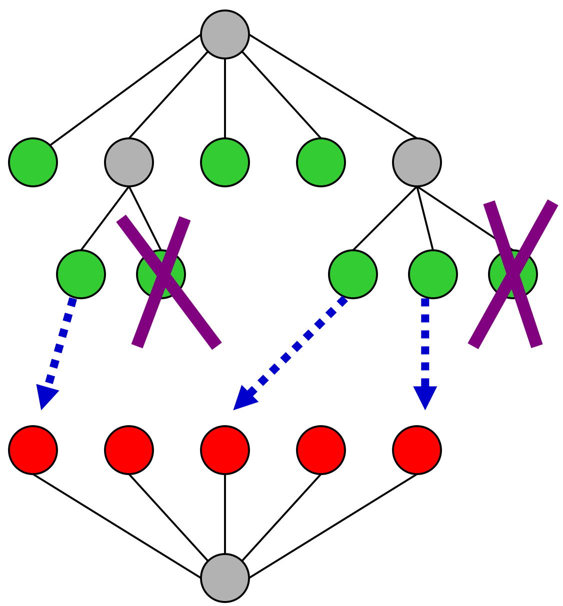
S = {h11’, h12’, h13’, h2’, …,hm’}

S = ~~{h1’, h2’, …,hm’}~~

- Replace the all hypotheses in S that do not cover a next positive example by ALL minimal generalizations that DO cover the example.

Invariant: only the hypotheses more general than the ones of S are still possible:
they do cover the positive example

Optimization: negative:



G = {h1, h21, …, hn}

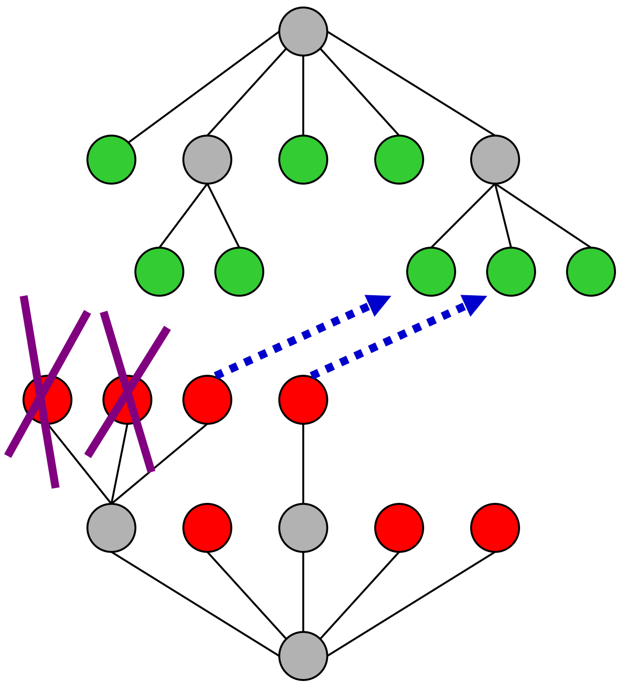
… are more general than …

S = {h1', h2', …,hm' }

- Only consider specializations of elements in G that are still more general than some specific hypothesis (in S)

Invariant: on S used !

Optimization: positive:



G = {h1, h21, h22, …, hn}

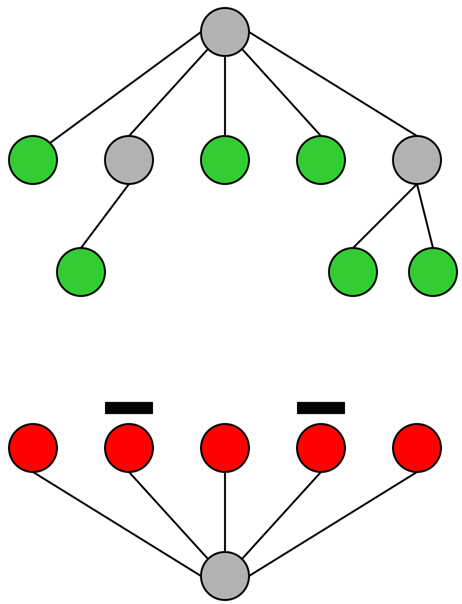
… are more specific than …

S = {h13', h2', …,hm'}

- Only consider generalizations of elements in S that are still more specific than some general hypothesis (in G)

Invariant: on G used !

Pruning: negative examples



$G = \{h1, h21, \ldots, hn\}$

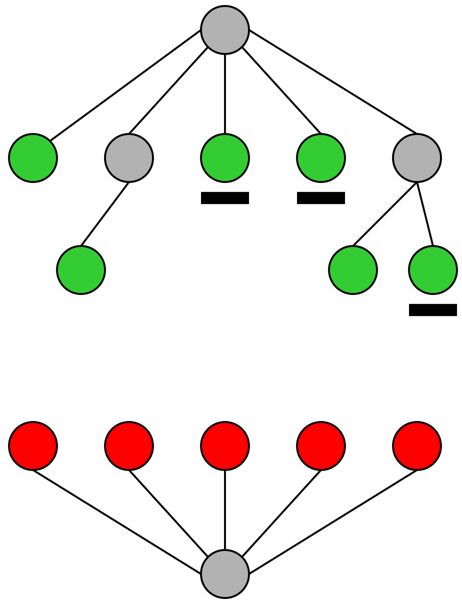
Cover the last negative example!

$S = \{h1', h3', \ldots, hm'\}$

- The new negative example can also be used to prune all the S - hypotheses that cover the negative example.

Invariant only works for the previous examples, not the last one

Pruning: positive examples



G = {h1, h21, …, hn}

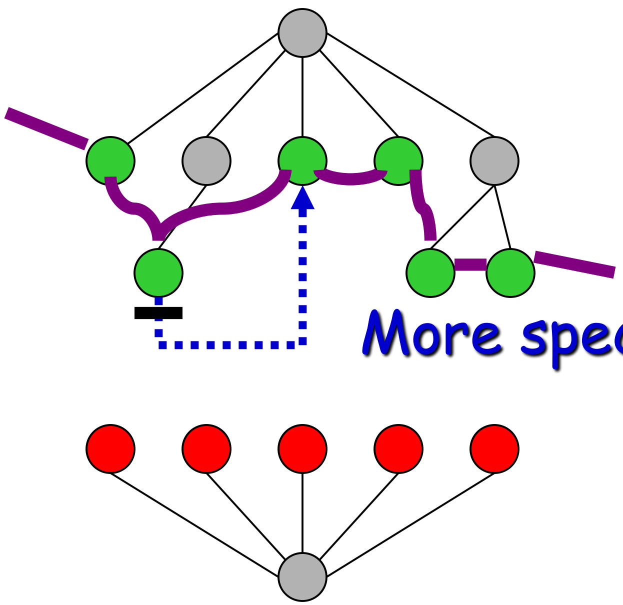
Don't cover the last positive example

S = {h1', h3', …,hm' }

- The new positive example can also be used to prune all the G - hypotheses that do not cover the positive example.

Eliminate redundant hypotheses

Obviously also for S !



$G = \{h1, h21, \ldots, hn\}$

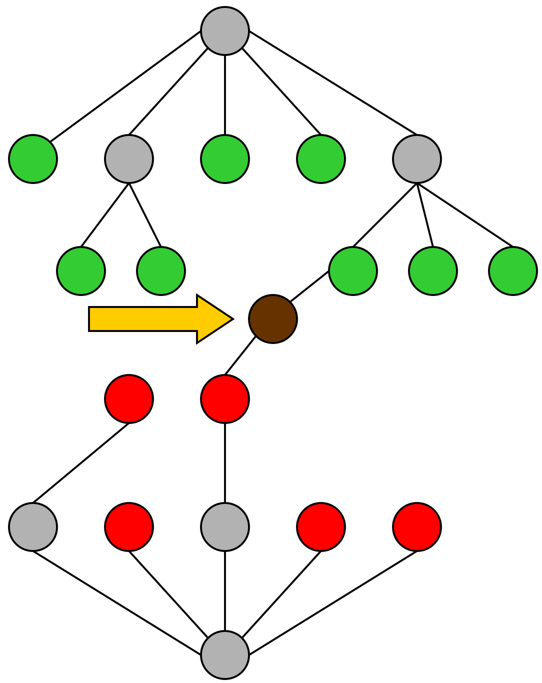
More specific than another general hypothesis

$S = \{h1', h3', \ldots, hm'\}$

- If a hypothesis from G is more specific than another hypothesis from G: eliminate it !

Reason: Invariant acts as a wave front: anything above G is not allowed. The most general elements of G define the real boundary

Convergence:



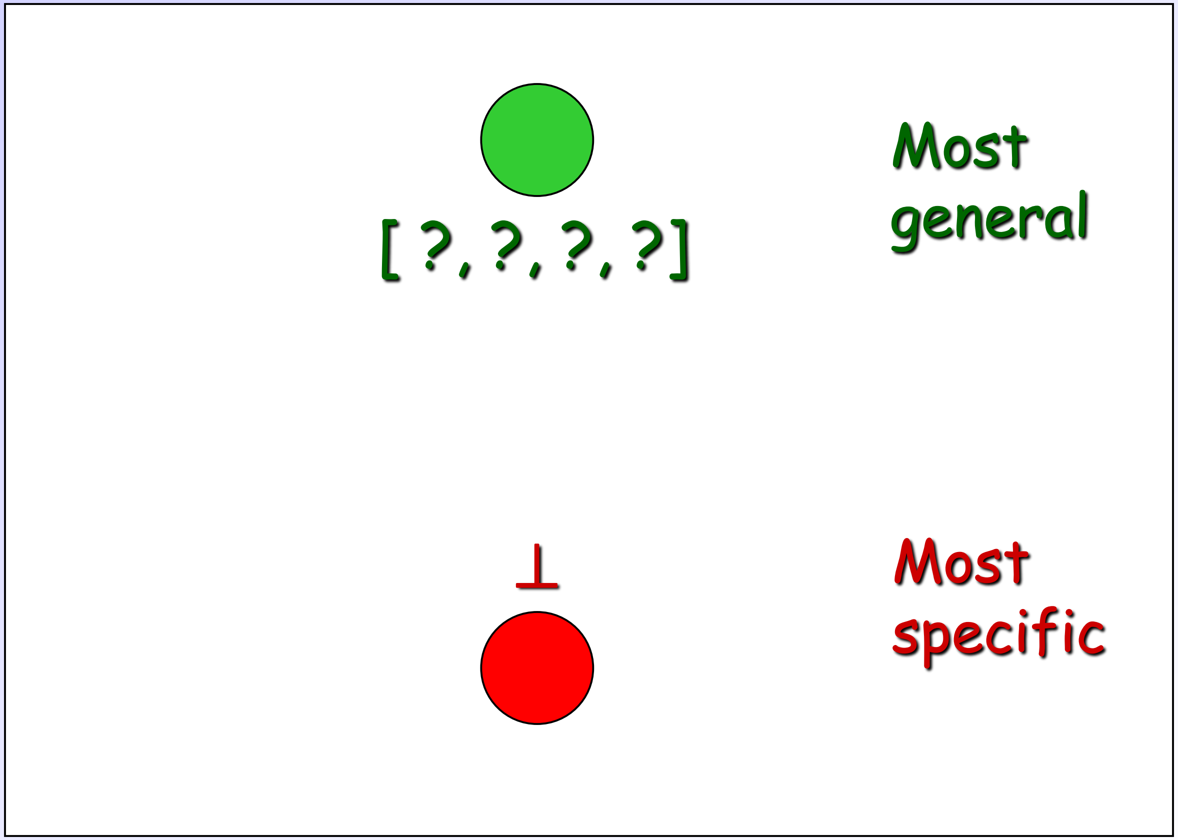
G = {h1, h21, h22, …, hn}

S = {h13', h2', …,hm'}

- Eventually, if G and S MAY get a common element: Version Spaces has converged to a solution.
- Remaining examples need to be verified for the solution.

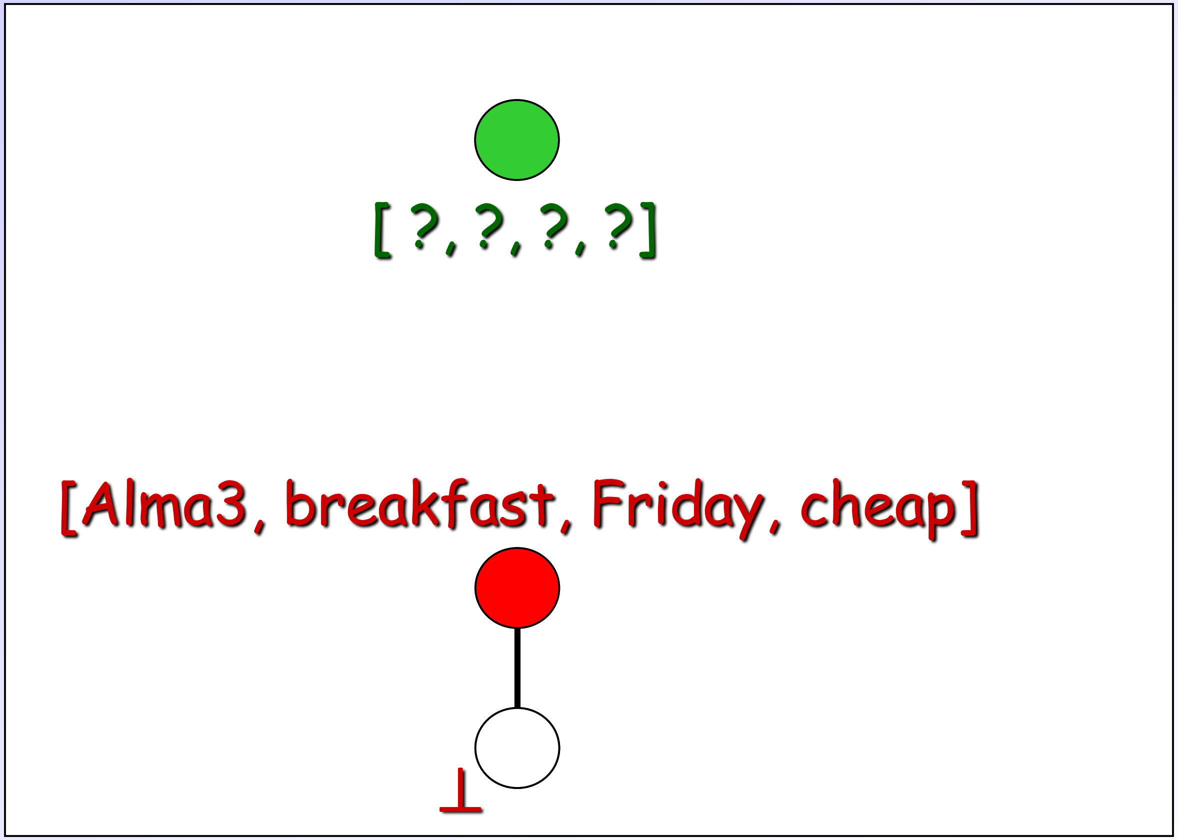
Reaction example

◎ Initialization:



Alma3, breakfast, Friday, cheap: +

- Positive example: minimal generalization of ⊥

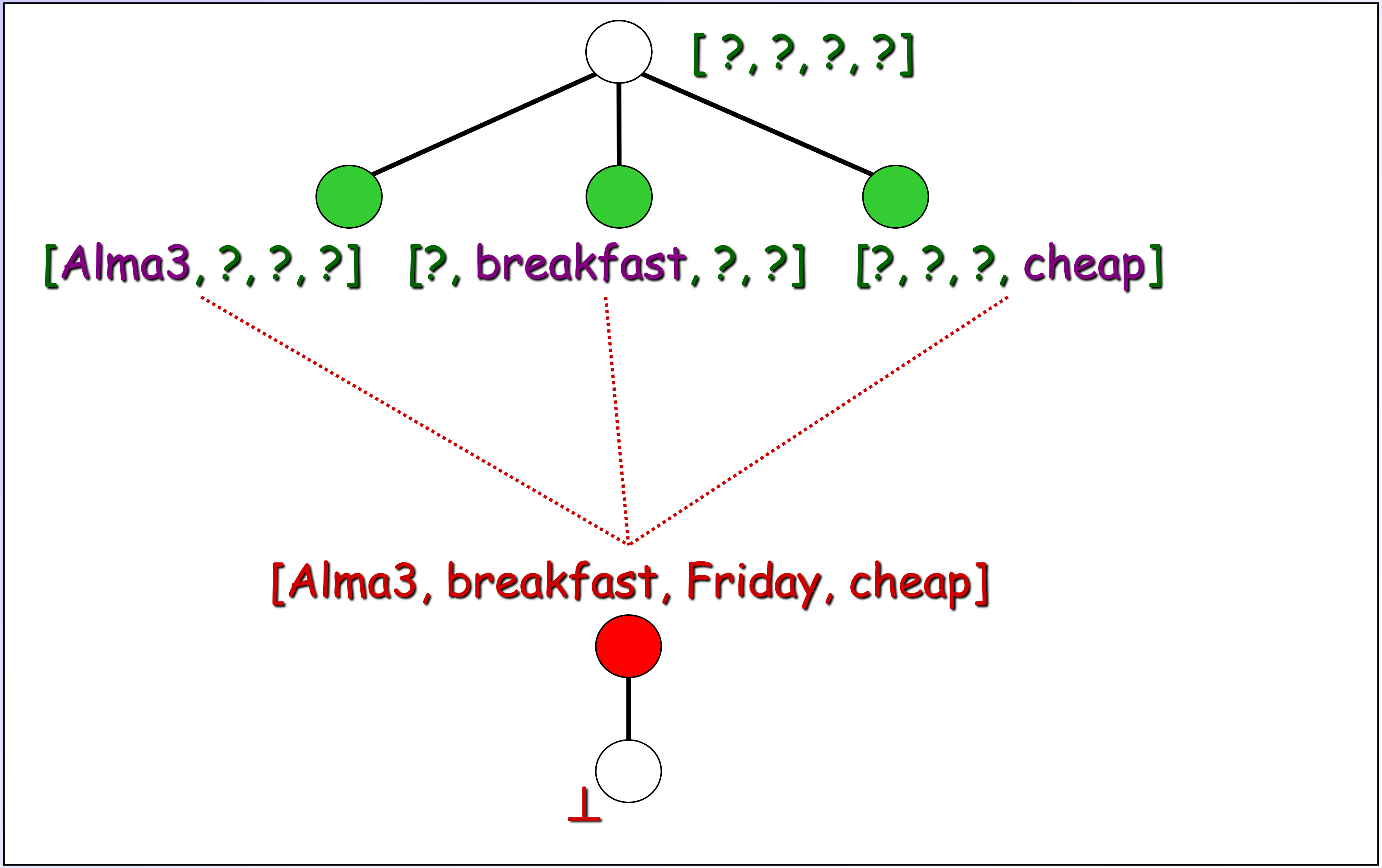


DeMoete, lunch, Friday, expensive: -

- Negative example: minimal specialization of [?, ?, ?, ?]
- 15 possible specializations !!

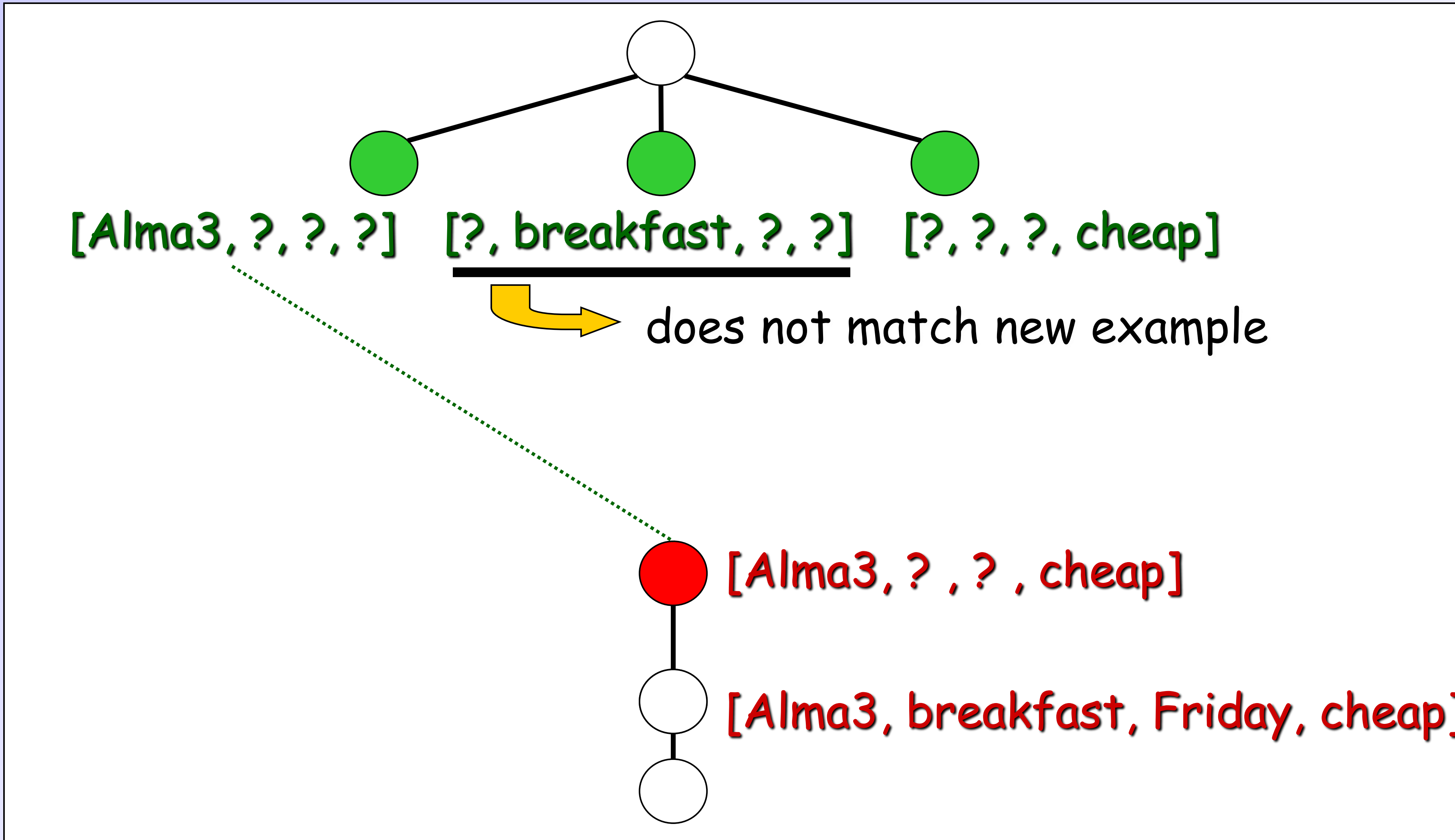
[Alma3, ?, ?, ?]		
[DeMoete, ?, ?, ?]	X	Matches the negative example
[Sedes, ?, ?, ?]	X	Does not generalize the specific model
[?, breakfast, ?, ?]		
[?, lunch, ?, ?]	X	
[?, dinner, ?, ?]	X	
[?, ?, Monday, ?]	X	
[?, ?, Tuesday, ?]	X	Remain !
[?, ?, Wednesday, ?]	X	
[?, ?, Thursday, ?]	X	
[?, ?, Friday, ?]	X	
[?, ?, Saturday, ?]	X	Specific model: [Alma3, breakfast, Friday, cheap]
[?, ?, Sunday, ?]	X	
[?, ?, ?, cheap]		
[?, ?, ?, expensive]	X	

Result after example 2:



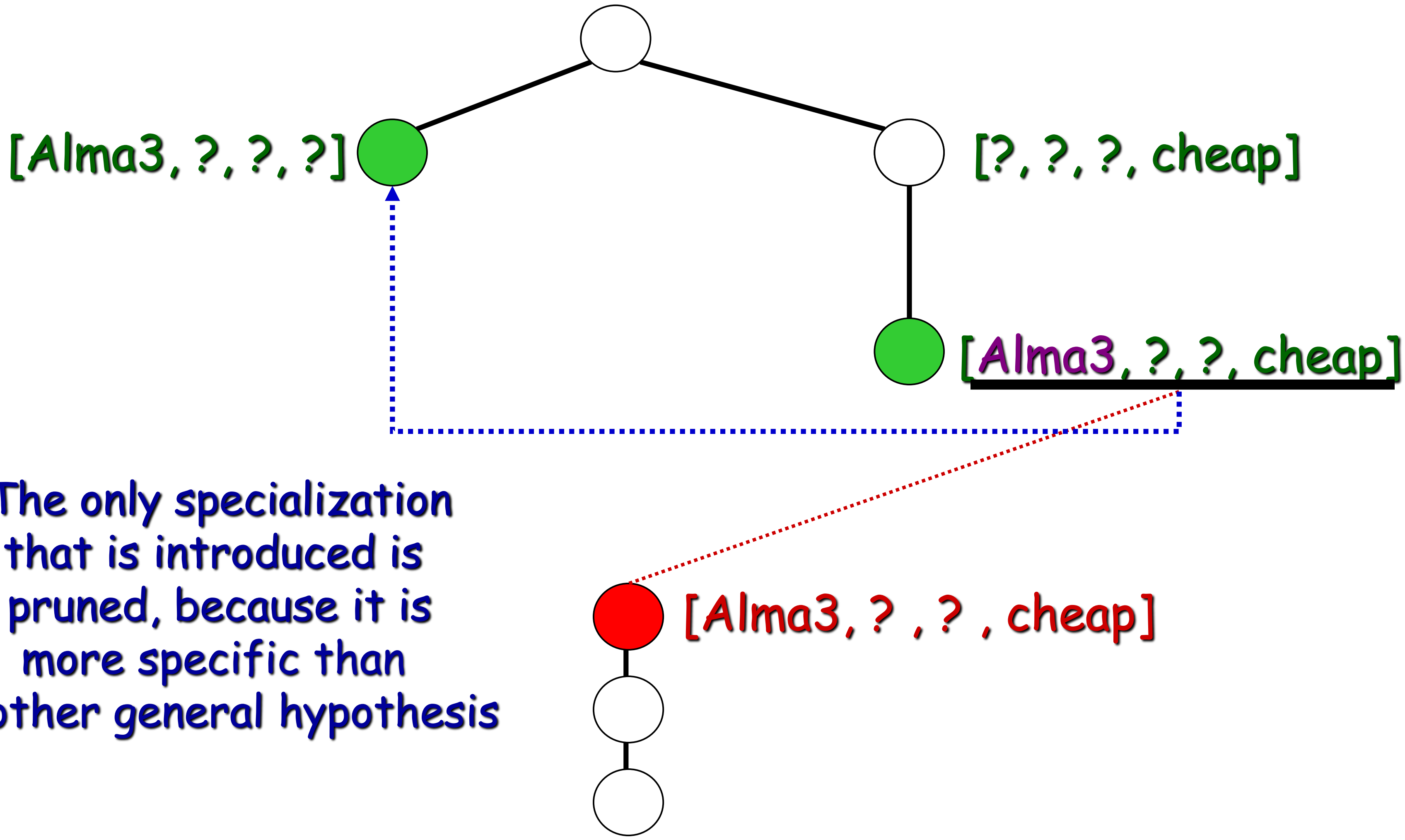
Alma3, lunch, Saturday, cheap: +

- Positive example: minimal generalization of [Alma3, breakfast, Friday, cheap]



Sedes, breakfast, Sunday, cheap: -

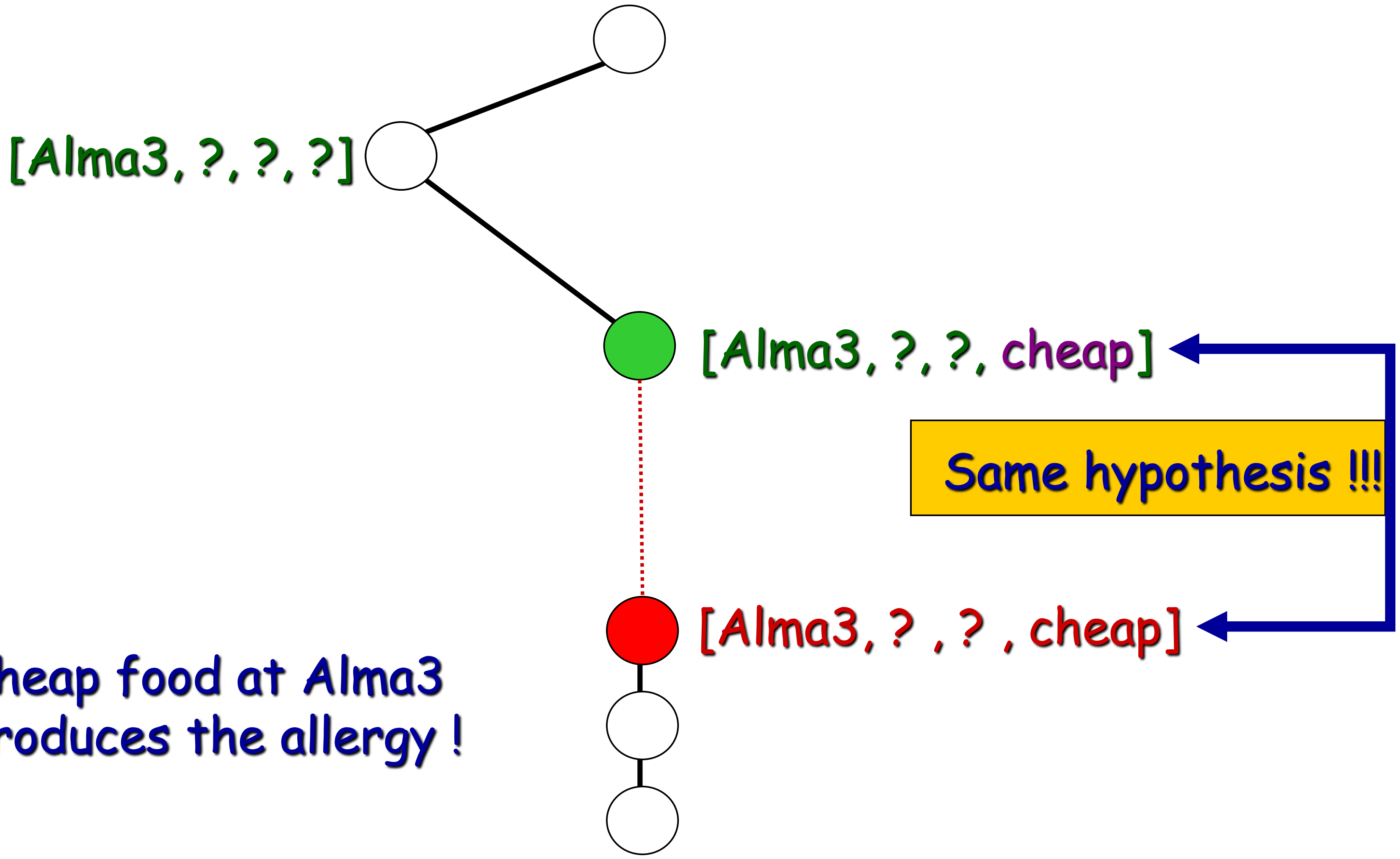
- Negative example: minimal specialization of the general models:



The only specialization that is introduced is pruned, because it is more specific than another general hypothesis

Alma 3, breakfast, Sunday, expensive: -

- Negative example: minimal specialization of [Alma3, ?, ?, ?]



Cheap food at Alma3 produces the allergy !

Version Space Algorithm:

Initially: G := { the hypothesis that covers everything}
S := {⊥}

For each new positive example:

Generalize all hypotheses in S that do not cover the example yet, but ensure the following:

- Only introduce minimal changes on the hypotheses.
- Each new specific hypothesis is a specialization of some general hypothesis.
- No new specific hypothesis is a generalization of some other specific hypothesis.

Prune away all hypotheses in G that do not cover the example.

Version Space Algorithm (2):

⋮

For each new negative example:

Specialize all hypotheses in G that cover the example, but ensure the following:

- Only introduce minimal changes on the hypotheses.
- Each new general hypothesis is a generalization of some specific hypothesis.
- No new general hypothesis is a specialization of some other general hypothesis.

Prune away all hypotheses in S that cover the example.

Until there are no more examples: report S and G

OR S or G become empty: report failure

Properties of VS:

- Symmetry:
  - ➔ positive and negative examples are dealt with in a completely dual way.

- Does not need to remember previous examples.

- Noise:
  - ➔ VS cannot deal with noise !
    - If a positive example is given to be negative then VS eliminates the desired hypothesis from the Version Space G !