UNIVERSITY of Bari ALDO MORO

Department of Information Technology

Master’s Degree in Computer Science

Fundamentals of

Artificial Intelligence

Case study

**GreMaES**

a Prolog-based Greenhouse Management Expert System

Francesco Peragine

f.peragine@studenti.uniba.it

m. 737873

<https://github.com/francescoperagine/GreMaES>

Sommario

[Introduction 3](#_Toc130828784)

[User mode 3](#_Toc130828785)

[Knowledge base mode 3](#_Toc130828786)

[Monitor mode 3](#_Toc130828787)

[Requirements 3](#_Toc130828788)

[Domain 4](#_Toc130828789)

[Ontology 4](#_Toc130828790)

[Conceptualization 5](#_Toc130828791)

[Analysis 5](#_Toc130828792)

[Predicates 6](#_Toc130828793)

[Rules 8](#_Toc130828794)

[Overview 8](#_Toc130828795)

[Monitor mode 16](#_Toc130828796)

[Logger 18](#_Toc130828797)

[Utilities 18](#_Toc130828798)

[Inference 19](#_Toc130828799)

[Knowledge Base 20](#_Toc130828800)

[Structure 20](#_Toc130828801)

[Plant domain 20](#_Toc130828802)

[Nutrient deficiencies 20](#_Toc130828803)

[Pests 23](#_Toc130828804)

[Diseases 26](#_Toc130828805)

[Conclusions 31](#_Toc130828806)

[Future developments 31](#_Toc130828807)

# Introduction

The present case study illustrates the development details of a Prolog-based *Expert System* (ES) within the domain of the ambient intelligence, in the context of the management of a greenhouse.

The plant domain has been inevitably oversimplified to be addressed in both size and complexity, and several assumptions also take place, like those relative to light (intensity, duration, spectrum), water (composition, hardness, pH) and so on.

It is possible to interact with the ES in three different ways: by submitting queries, by consulting the knowledge base and through sensors.

## User mode

The user is guided by an interactive shell interface to submit the symptoms that are manifested on a plant, to obtain a diagnosis of its health problems. The user may provide any number of captions.

For the sake of explainability, every diagnosis is provided with the set of rules that led to the actual results.

## Knowledge base mode

The user can fully explore every information stored in the knowledge base.

## Monitor mode

To ensure safety and the correct lifecycle of hosted plants, the agent carries out the following actions:

* continuous monitoring by the means of sensor devices
* storing environment readings
* identification of the health statuses and conditions that may affect the plants based on stored environment readings
* controlling the environment thanks to actuator devices
* showing notifications to the user through the Command Line Interface
* log all details in a text file.

Sensors devices provides the inputs: readings for temperature and humidity, image captions for textual descriptions of any strange manifestation occurred on the plant. All readings are simulated through simple random sampling strategy performed with timed cadence. At the end of every batch the user is asked if he wishes to continue with the sampling.

Problems that have been identified and that can be directly addressed through actuator devices are automatically fixed, like a low humidity level can be fixed activating the watering system.

At the end of the process, a detailed report is shown.

#### Debug mode

A debug mode has been introduced to speed up the testing process: it interactively asked to the user to activate it and, if so, the sampling process is skipped and a set of readings is loaded from the *plant\_reading\_samples* file.

## Reasoning

The identification of pathologies that affect the plant occurs by inductive reasoning, based on the information gathered by sensors with respect to the knowledge base.

## Task Environment

Fully observable

## Requirements

The software was developed with YAP (Yet Another Prolog) v.6.2.2.

Some SWI Prolog libraries were also included.

# Domain

*Health problems* are families of issues that may affects the plants. There are three main categories:

* **Abiotic disorders**, caused by nonliving factors, like freeze, nutrient deficiencies, overwatering, ...
* **Biotic diseases** caused by living organisms
* **Pests’ infestations**

Every health problem has several unique condition *types,* and every type has one or more *symptoms* and may have multiple *treatments*, if any.

Symptoms are manifested through *signs* on *sections* of the plants and may show different *colors*.

## Immagine che contiene diagramma Descrizione generata automaticamenteOntology

# Conceptualization

**Domain**: Plants

**Goal**: distinguishing the health conditions of a plant.

## Analysis

* Each **plant** is characterized by:
  + name
  + species
  + growth stage
  + health status
  + set of connected sensor devices
  + set of connected actuator devices
* **Species** have:
  + name
  + temperature range
* **Growth stage:**
  + identifier
  + humidity range
* **Sensor devices** perform readings of the environment properties
  + identifier
  + metric (temperature, humidity)
  + sensor devices
* **Actuator devices** alter the properties of the environment:
  + identifier
  + type
  + activation status
  + class
* **Health problems**, that may affect plants, have:
  + category
  + name
  + set of manifested conditions
* **Condition**
  + set of symptoms
  + set of treatments
* **Symptoms** are physical manifestations of health problems, are denoted by:
  + sign
  + section
  + color

## Predicates

* **plant**(X, Y, Z) = plant X is of species Y and has growth stage Z
  + Constants: {trinidad1, sunflower, p1, p2, …}
* **health\_status**(X, Y) = plant X has status Y
  + Constants: {healthy, abiotic, biotic, pests

{hot, cold, wet, dry, infestation, disease, nutrient\_deficiency}

* **species**(S, X, Y) = species S has ideal range of temperature [X - Y]
  + Constants: {Rudbeckia hirta, …,

-5, …, 40}

* **growth\_humidity**(S, X, Y) = stage S has ideal range of humidity [X - Y]
  + Constants: {flowering\_mature, vegetative\_growing, seed\_germination,

…,

0 - 100}

* **problem\_condition**(X, Y) = problem X is due to condition Y
  + Constants: {healthy, nutrient\_deficiency, disease, infestation,

nitrogen, phosphorus, potassium, sulfur, …}

* **sign\_location**(X, Y) = sign X occurs on location Y
  + Constants: {none, altered\_color, angular\_lesions, black\_leathery\_spot, …

all, branches, leaves, lower\_leaves, roots, … }

* **sign\_color**(X, Y) = sign X has color Y
  + Constants: {altered\_color, cotton\_like\_downy\_substance, flies, …

chlorotic, dark\_green, blotchy\_chlorosis, interveinal\_chlorosis, …}

* **symptom**(X, Y, Z) = symptom occurs on section X with sign Y and color Z
* **treatment**(X, Y) = condition X may be solved with treatment Y
  + Constants: {'spray with neem oil and insecticidal soap', 'cut off and remove the infected leaves or flowers', 'shower the plant once a week', …}
* **sensor**(X, Y) = sensor X reads percept Y
  + Constants: {t11, t12, t13, t14, …

temperature, humidity, caption}

* **plant\_sensor**(X, Y) = plant X is connected to input device Y
* **actuator**(X, Y, W, Z) = actuator X manages environment property Y with status W and has class Z
  + Constants: {act1, act2, act3, …

low, normal, high,

thermostat, fan, sprinkler}

* **plant\_actuator**(X, Y) = plant X is connected to actuator device Y
  + Constants: {act1, act2, act3, act4}
* **actuator\_status**(X, Y) = actuator X has status Y
  + Constants: {on, off}

## Rules

Conditions are characterized by a limited set of symptoms (up to three within the case study), which are matched with the provided inputs, by users or sensors, to diagnose a health problem.

### Overview

The program starts by welcoming the user, cleaning up the working memory and asking which mode is to be executed.

The choices represented are between the diagnostic *user mode*, the *knowledge base mode* and the automatic *monitor mode*.

Once the selected task is completed, the user is asked if he wants to run the program again.

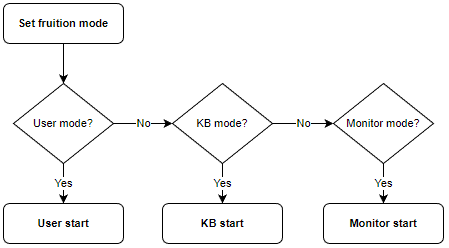


Immagine che contiene diagramma

Descrizione generata automaticamente**user\_start**

*% user\_mode\_start/0*

user\_mode\_start :-

    symptomatology,

    user\_diagnosis.

The user mode starts by asking how the issue has been manifested and of which color it was, if any.

The **sign** is then temporarily stored to build up the symptom by asking its localization.

To reduce the search and not let the user be overwhelmed by the listing of every possible combination of signs, locations and colors that are to be merged to build every symptom, every choice that is provided to the user will show only the subset of combinations present in real symptoms that manifest health conditions.

Once the symptom has been saved, the temporary information is deleted and the user may keep registering new ones, if any.

The diagnostic process is performed at the end of the gathering phase.

**symptomatology**

*% symptomatology/0*

symptomatology :-

    repeat,

    ask\_symptom,

    \+ (once\_again).

*% once\_again/0*

once\_again :- askif(new\_symptom).

Asks repeatedly the user to provide symptoms, until the user specifies otherwise.

**ask\_symptom**

*% ask\_symptom/0*

ask\_symptom :-

    user\_input(signs, Sign),

    assertz(current\_sign(Sign)),

    symptomatology\_forward,

    symptomatology\_cleanup.

The user is asked to select the sign shown on the plant. Stores it in the working memory and proceeds with the next steps.

**symptomatology\_forward**

*% symptomatology\_forward/0*

*% If the sign is not associated to a color, sets it to none.*

symptomatology\_forward :-

    current\_sign(Sign),

    \+ sign\_color(Sign, \_),

    user\_input(sign\_locations, Location),

    assertz(symptom(Location, Sign, none)).

*% If the sign is associated to a color, asks it.*

symptomatology\_forward :-

    current\_sign(Sign),

    sign\_color(Sign, \_),

    user\_input(sign\_colors, Color),

    user\_input(sign\_locations, Location),

    assertz(symptom(Location, Sign, Color)).

After the user provided the sign, the KB is consulted to check whether that sign is associated to a color. If so, the user is asked to also provide it.

After that, the user is asked to select the section on which the sign occurs.

**symptomatology\_cleanup**

*% symptomatology\_cleanup/0*

symptomatology\_cleanup :-

    retractall(asked(\_,\_)),

    retractall(current\_sign(\_)).

Removes temporary informations to prepare to eventually register the next symptom.

**user\_input(+Relation, -UserChoice)**

*% user\_input/2*

user\_input(Relation, UserChoice) :-

    call(Relation, Options),

    show\_title(Relation), nl,

    show\_options(Options),

    read(UserInput),

    input\_choice(Options, UserInput, UserChoice),

    write\_message(option\_selected), write(UserInput), write(': '), writeln(UserChoice), nl.

Unifies *Relation* with a list of options, reads the user selection from that list and returns it as UserChoice.

**sign\_location(-Locations)**

*% sign\_location/1*

sign\_locations(Locations) :-

    current\_sign(Sign),

    all(Location, sign\_location(Sign, Location), Locations).

Unifies *Locations* with the list of all possible locations in which the temporary stored sign can manifest.

**sign\_colors(Colors)**

*% sign\_colors/1*

sign\_colors(Colors) :-

    current\_sign(Sign),

    all(Color, sign\_color(Sign, Color), Colors).

Unifies C*olors* with the list of all possible colors that the temporary stored sign can manifest.

The match between colors and locations

Based on the choice that the user must make (sign, sign color, sign section), calls the menu to show the appropriate list of values and returns the user’s pick.

**current\_sign\_sections(-L)**

*% current\_sign\_sections/2*

current\_sign\_sections(L) :-

    current\_sign(M),

    all(S, sign\_section(M, S), L).

Returns the list of all sections on which the selected sign may appear, from which the user must choose.

**current\_sign\_colors(-L)**

*% current\_sign\_colors/2*

current\_sign\_colors(L) :-

    current\_sign(M),

    all(C, sign\_color(M, C), L).

Returns the list of all colors that a sign may present, from which the user must choose.

**menu\_ask(+T, +L, -Y)**

*% menu\_ask/3 - Menu title T, options' list L, entry name X*

menu\_ask(T, L, Y) :-

    menu\_with\_title(T, L), nl,

    message\_code(item\_number\_no\_exit, M),

    read(X),

    menu\_input\_handler(L, X, Y),

    message\_code(option\_selected, N),

    write(N), write(X), write(': '), writeln(Y), nl.

Provided the title T of the menu, shows the options from list L and returns the user choice Y.

**menu\_with\_title(+T, +L)**

*% menu\_display/2 - Shows the title and goes ahead with the diplaying part.*

menu\_with\_title(T, L) :-

    message\_code(T, X),

    writeln(X),

    menu\_display(L).

menu\_with\_title(T, L) :-

    \+ message\_code(T, X),

    writeln(T),

    menu\_display(L).

Writes the appropriate message for the menu or the string itself if none is available.

**menu\_display(+L)**

*% menu\_display/1 - Options' list L, counter starts from 1*

*% Shows a numbered list of ordered options, stripping atom names from their underscores.*

menu\_display(L) :- menu\_display(L, 1), !.

menu\_display([], \_).

menu\_display([H|T], N) :-

    atomic\_concat([N, '. ', H], A),

    writeln(A),

    N1 is N + 1,

    menu\_display(T, N1).

Shows the entries of the list associated with an index number, counting starts from 1.

**menu\_input\_handler(+L, +X, -Y)**

*% menu\_input\_handler/3 - The entry number X of list L unifies with list entry name Y.*

menu\_input\_handler(L, X, Y) :-

    integer(X),

    nth1(X, L, Y),

    validate\_entry(L, Y).

Provided the list of options L and the index number X selected by the user, retrieves the value of the list with that index, checks if is actually part of the list and returns it.

**validate\_entry(+L, +X)**

*% validate\_entry/2*

validate\_entry(L, X) :-

    member(X, L),

    !.

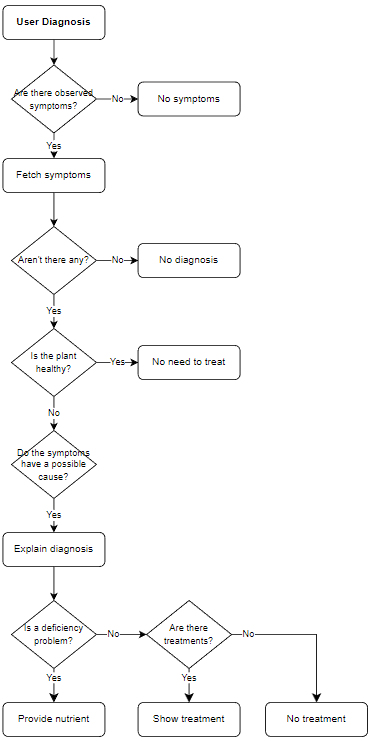
validate\_entry(L, X) :-

    not(member(X, L)),

    message\_code(not\_recognized\_value, M),

    writeln(M).

Checks if the value is part of the list, otherwise writes a notice.

**user\_diagnosis**

*% user\_diagnosis/0*

user\_diagnosis :-

    observed\_symptoms,

    all(diagnosis(C, B), (condition\_body(C, B), observed\_symptoms(S), match(B, S)), D),

    maplist(explain\_diagnosis, D).

user\_diagnosis :-

    \+ observed\_symptoms,

    message\_code(no\_symptom, M),

    writeln(M).

user\_diagnosis :-

    observed\_symptoms,

    \+ (condition(X)),

    message\_code(no\_diagnosis, M),

    writeln(M).

The diagnosis starts by checking whether there is any registered symptom or not. If so, all the observed symptoms are collected and the types of the problem are inferred.

A diagnosis, exposes whether there is an healthy issue, its type, that is directly inferred from symptoms, its class and the health status it corresponds to.

Multiple symptoms may not lead to any diagnosis, to diagnosis which have no treatments because they are healthy or because there is none or, most commonly, to problems that have their own set of treatments to be performed.

In the case of a deficiency problems, the fixed treatment is that of providing nutrients.

**observed\_symptoms**

*% observed\_symptoms/0*

observed\_symptoms :- symptom(\_,\_).

observed\_symptoms :- symptom(\_,\_,\_).

To proceed with the diagnosis, checks if there are any observed symptoms.

**observed\_symptoms(-S)**

*% observed\_symptoms/1 Unifies S with the aggregated lists of symptoms*

observed\_symptoms(S) :-

    findall(symptom(M1, S1), (condition(T1), symptom(M1, S1)), R1),

    findall(symptom(M2, S2, C), (condition(T2), symptom(M2, S2, C)), R2),

    append([R1, R2], L),

    all(X, member(X, L), S).

Unifies S with the aggregated lists of symptoms registered by the user.

**condition\_body(-C, -B)**

*% condition\_body/2 Unifies B with the right side of the type predicate*

condition\_body(C, B) :-

    condition(C),

    clause(condition(C), R),

    conj\_to\_list(R, B).

For every condition that matches the observed symptoms, returns the condition and its symptoms in list format.

**explain\_diagnosis(+X)**

*% explain\_diagnosis/1 - X Right side of the condition(X) rule, to be checked vs the stored symptoms.*

explain\_diagnosis(X) :-

    X = diagnosis(T, [B]),

    T \= healthy,

    problem\_card(T, H, C),

    atomic\_concat([H, ' - ', T, ' ', C], A),

    message\_code(diagnosis\_of, M),

    message\_code(because\_of, N),

    nl, write(M), write(A), write(N), writeln(B),

    show\_treatment(T), nl.

explain\_diagnosis(X) :-

    X = diagnosis(T, [B]),

    T = healthy,

    message\_code(treatment\_healthy, M),

    writeln(M), nl.

The symptoms registered by the user are checked vs the conditions present in the KB.

If there are matches, the diagnosis for each condition will be printed as well as the respective causes and treatments.

**show\_treatment(+X)**

*% show\_treatment/1*

show\_treatment(X) :-

    problem\_condition(nutrient\_deficiency, X),

    message\_code(missing\_nutrient, M),

    writeln(M).

show\_treatment(X) :-

    \+ problem\_condition(nutrient\_deficiency, X), *% nutrient deficiencies have no direct treatment.*

    bagof(Y, treatment(X, Y), L),

    message\_code(treatment, M),

    writeln(M),

    maplist(writeln, L).

show\_treatment(X) :-

    \+ problem\_condition(nutrient\_deficiency, X),

    \+ treatment(X),

    message\_code(treatment\_none, M),

    writeln(M).

Shows the treatment for the provided condition X, if any.

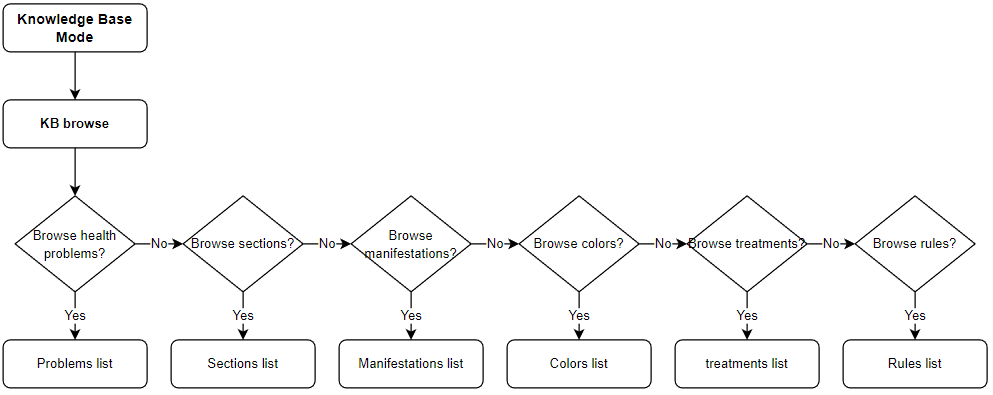
**kb\_start**

*% kb\_start/0*

kb\_start :-

    L = [health\_problems, sections, signs, colors, treatments, rules],

    maplist(kb\_browse, L).

The kb\_mode allows the user, straightforwardly, to browse the knowledge base categories facts.

**kb\_browse(+X)**

*% kb\_browse/1*

kb\_browse(X) :-

    askif(view(X)),

    browse(X).

kb\_browse(X) :-

    asked(view(X), A),

    negative(A).

**health\_problems(-L1)**

*% health\_problems/1*

health\_problems(L1) :-

    all(T, (problem\_condition(C, T), C \= healthy), L),

    maplist(problem\_card, L, L1).

For every non-healthy condition in the KB, shows

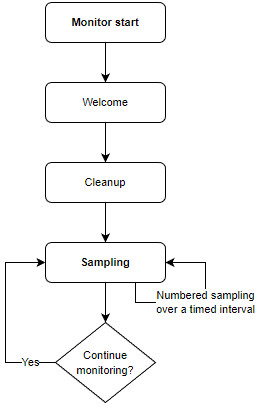
**sections(L)**

**colors(L)**

**rules(L)**

**browse(X)**

### Monitor mode

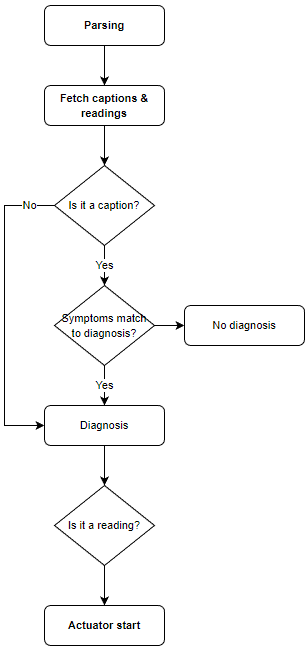
As for the user mode, the monitor mode starts by welcoming the user and cleaning up the cached informations.

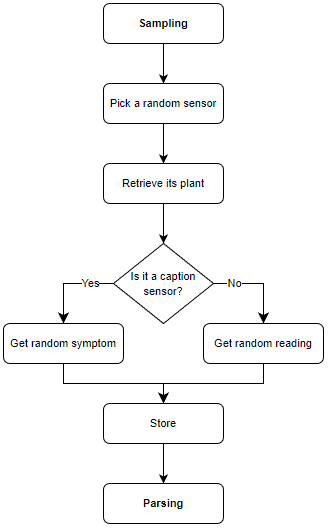
The monitoring activity consists in performing timed readings by the means of the sensor devices.

For the sake of the sampling within a virtualized environment, the readings are simulated by randomly generating symptoms: the same [filtering behavior](#filtering_behavior) exposed in the user\_mode has been implemented here, as it could take a long time before registering a proper symptom that could lead to a diagnosis, therefore making difficult to illustrate the process.

The diagnostic process is performed at each sampling iteration, to allow the actuator devices to adjust the environment to the right measures and save all the informations in a log file.

The sampling continues until stopped by the user.





**welcome\_monitor**

**monitor\_cleanup**

**greenhouse\_init**

**timestamp(T)**

**actuator\_init**

**plants(L)**

**monitor\_loop\_start**

**monitor\_loop(X, Y)**

**sampling\_start**

**random\_list\_element(L, D)**

**sampling(D, T, P)**

**range\_value(T, Min, Max)**

**caption\_forward(M, A)**

**sampling\_sign\_section(M, S)**

**random\_predicate\_element(P, E)**

**store(P, T, D, A)**

**parsing\_start**

**parsing\_readings(L)**

**parse(X)**

**actuator\_start**

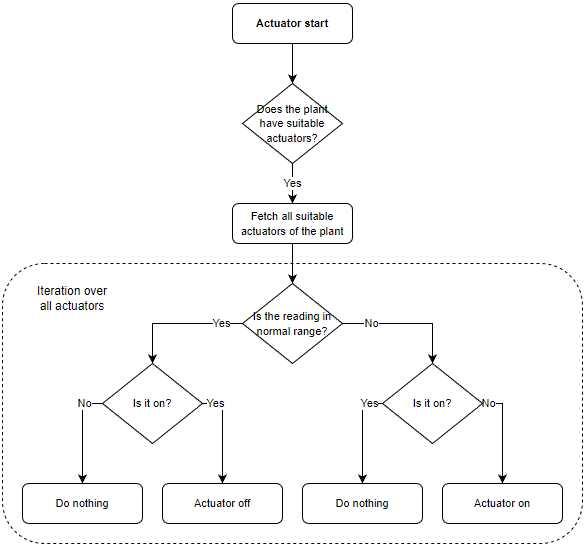
After each diagnosis have been reached, is it time for the actuator devices to be involved.

The diagnosis input provides the plant identifier, the property to be managed and its status - with respect to the ideal range of values of the species and of the humidity stage – and the average value to eventually achieve.

Every plant may have zero or more actuator devices, and every actuator device has its own specific use, therefore each plant may not have any actuator device suited for a task.

In the case that they do, actuators devices are orchestrated to turn on and off based on needs.

Furthermore, changes in their statuses are notified and logged.



**actuator\_forward(X)**

**actuator\_activate(A, T, S, K, H)**

**actuator\_on(A)**

**actuator\_off(A)**

**greenhouse\_status**

### Logger

**lognl(X)**

**log(X)**

### Utilities

**type\_body(T, B)**

**conj\_to\_list((H, C), [H|T])**

**list\_to\_conj([H|T], (H, C))**

# Inference

# Knowledge Base

## Structure

## Plant domain

### Nutrient deficiencies

#### Nitrogen

* pale yellow color (chlorosis)
* older leaves turn completely yellow.
* flowering, fruitings, protein and starch contents are reduced
* reduction in protein results in stunted growth and dormant lateral buds

#### Phosphorus

* smaller leaf sizes
* lessened number of leaves
* slower rate maturation
* leaves and stems appear dark green or purple
* older leaves are affected first

#### Potassium

* reduced growth
* chlorosis and necrosis occurring in older leaves in later growth stages
* older leaves show mottled or chlorotic areas with leaf burn at the margins, usually leaving the midrib alive and green
* brown scorching and curling of leaf tips as well as chlorosis (yellowing) between leaf veins
* purple spots may also appear on the leaf undersides
* plant growth, root development, and seed and fruit development are usually reduced in potassium-deficient plants

#### Sulfur

* resembles nitrogen deficiency except yellowing occurs in new, younger leaves, rather than old, lower leaves.

#### Magnesium

* interveinal chlorosis with green mid-ribs
* Leaf margins become yellow or reddish-purple

#### Boron

* chlorotic young leaves and death of the main growing point
* leaves may develop dark brown, irregular lesions
* whitish-yellow spots may form at the base of the leaves
* leaves may become thickened, distorted and curled
* stems may be stunted
* flower buds may fail to form or be misshapen

#### Calcium

* localized tissue necrosis leading to stunted plant growth
* necrotic leaf margins on young leaves or curling of the leaves, and eventual death of terminal buds and root tips
* new growth and rapidly growing tissues of the plant are affected first
* the mature leaves are rarely if ever affected
* reduced height, fewer nodes, and less leaf area

#### Chloride

* chlorotic and necrotic spotting along leaves with abrupt boundaries between dead and alive tissue
* wilting of leaves along margins
* highly branched roots

#### Copper

* chlorotic younger leaves
* stunted growth
* delayed maturity
* excessive tillering
* lodging and sometimes brown discoloration

#### Iron

* yellowing (Chlorosis) occurs in the newly emerging leaves instead of the older leaves and usually seen in the interveinal region
* fruit would be of poor quality and quantity
* the yellowing may turn a pale white or the whole leaf may be affected

#### Manganese

* plant disorder that is often confused with, and occurs with, iron deficiency
* most common in poorly drained soils, also where organic matter levels are high
* manganese may be unavailable to plants where pH is high
* yellowing of leaves with smallest leaf veins remaining green to produce a ‘chequered’ effect
* younger leaves may appear to be unaffected
* brown spots may appear on leaf surfaces
* severely affected leaves turn brown and wither

#### Zinc

* growth is limited because the plant
* cannot take up sufficient quantities of this essential micronutrient from its growing medium.
* chlorosis
* necrotic spots
* bronzing of leaves
* resetting of leaves
* stunting of plants
* dwarf leaves
* malformed leaves

### Pests

#### Aphids

Aphids live only about a week, but a mature female can reproduce rapidly. The tiny sucking pests, often found growing en masse on the underside of leaves, emit a sticky substance that draws ants and attracts sooty mold. Control aphids with neem oil or insecticidal soap.

* mass in large number
* sticky honeydew deposits
* white or grey "husks" littering the soil
* leaves become chlorotic in random patches
* growth may become distorted

Treatment

* spray with neem oil and insecticidal soap

#### Thrips

Thrips are tiny flying insects with fringed wings. The sap-sucking insects discolor and distort nearly any type of plant. They leave tiny black specks of excrement on the leaves and often create white patches on leaves and petals. Thrips are difficult to control and often require a combination of methods such as sticky traps and insecticidal soap or neem oil.

* mottling, streaking, browning or yellowing on the leaves

Treatment

* cut off and remove the infected leaves or flowers.
* spray with neem oil or natural pyrethrum

#### Spider mites

Spider mites are difficult to see with the naked eye, but they are easily recognized by the fine webs. The pests cause streaking, spotting and discolored leaves that may fall off the plant if not controlled. Neem oil and insecticidal soap are effective. Water properly, as mites are drawn to dry, dusty conditions.

* sticky webbing
* mottled leaves with lots of brown dots

Treatment

* shower the plant once a week
* purchasing the predatory mite Phytosieulus persimilis
* spray with neem oil and insecticidal soap

#### Scale insects

Scale damage can be devastating, as the tiny pests suck out the sweet nectar. There are two types of scale: hard scale, found primarily on woody tissue such as branches, trunks and twigs; and soft scale, which has a waxy protective covering. Control can be difficult, but neem oil works well by suffocating the pests. Regular use of insecticidal soap is also effective.

* sticky honeydew

Treatment

* spray neem oil and insecticidal soap
* dab individual scales with alcohol

#### White flies

Whiteflies are yet another type of sap-sucking pest. Small numbers are relatively harmless but large infestations can cause yellow or dry leaves that may fall off the plants. Like other sap-sucking pests, the sweet substance created by whiteflies attracts ants and sooty mold. To control whiteflies, try sticky traps and insecticidal soap or neem oil.

* leaves chlorosis
* dry leaves

Treatment

* spray neem oil and insecticidal soap
* sticky trap

#### Cutworms

Cutworms are the larval stage of certain moths. The destructive pests hide under leaves or other plant debris, emerging to lay masses of eggs on plants. They eat nearly anything in their paths, often cutting through stems of young plants at ground level. Remove plant debris. Pick off the pests by hand in late afternoon or evening. Create barriers with cardboard collars or gritty substances like eggshells, coffee grounds, or diatomaceous earth. Encourage birds to visit your garden.

#### Fungus gnats

Fungus gnats are tiny, annoying pests that wreak havoc on houseplants or in gardens or greenhouses. The swarms of flying insects are annoying, but it’s the larvae that does the most harm by eating plant roots. Fungus gnats may also carry disease from plant to plant. Control adults with bright yellow sticky traps and/or insecticidal soap.

* small black flies around 2mm long

Treatment

* keep the soil less moist until they leave
* use the bottom watering method
* mix the nematodes with water and water directly

#### Mealy bugs

Mealybugs are common both indoors and outdoors, where they cause stunted growth, withering and yellowing of plants. The pests are easily recognized by the cottony protective covering. Insecticidal soap works well against the pests. Light infestations on indoor plants can also be removed with a toothpick or a cotton swab dipped in rubbing alcohol.

* clustering cottony covering under leaves and in the leaf joints
* plants look dehydrated
* plants may lose leaves rapidly
* stunted growth
* chlorotic leaves eventually drop off
* sticky honeydew residue

Treatment

* poke them off with a shake
* spraying with water
* spray neem oil and insecticidal soap

### Diseases

#### Black spot

Immagine che contiene pianta, albero

Descrizione generata automaticamentefungal, black round spots, upper side leaves (lower ones infected first). Infected leaves turn yellow and fall off. It occurs in extended wet weather periods or when leaves are wet for 6+ hours.

Tips for Controlling Black Spots on Leaves

* Plant in well-draining soil. Keep your plants healthy by providing regular feedings of organic fertilizer. This will help prevent fungal disease in plants.
* The fungus spores overwinter in plant debris. Remove dead leaves and infected canes from around the plants and disguard in the trash. Do not add to the compost pile.
* Disinfect your pruners with a household disinfectant after every use.  Ethanol or isopropyl alcohol can be used straight out of the bottle.
* Because water (not wind) spreads the fungal spores, avoid applying water on the leaves.  When you water, apply water directly to the roots. Use a soaker hose to water plants prone to the disease.

#### Immagine che contiene esterni, pianta, albero, verde Descrizione generata automaticamenteLeaf Spots

Fungal leaf spot disease can be found both indoors on houseplants, and outdoors in the landscape. This occurs during warm, wet conditions. As the disease progresses, the fungal spots grow large enough to touch each other.  At this point the leaf surface appears more like blotches than spots.  Leaf spot may result in defoliation of a plant.  Follow the same tips as the ones to control black spot.

#### Powdery Mildew

Immagine che contiene pianta, fungo, muffa, verde

Descrizione generata automaticamentePowdery mildew is a fungal disease that affects many of our landscape plants, flowers, vegetables and fruits. Powdery mildew is an easy one to identify. Infected plants will display a white powdery substance that is most visible on upper leaf surfaces, but it can appear anywhere on the plant including stems, flower buds, and even the fruit of the plant.  This fungus thrives during low soil moisture conditions combined with high humidity levels on the upper parts of the plant surface.  It tends to affect plants kept in shady areas more than those in direct sun.

#### **Tips for Controlling Powdery Mildew**

* Inspect plants that you buy from a greenhouse before purchasing for mildew (and insects).
* Wiping off the leaves is not an effective powdery mildew treatment as it will return within days of cleaning.
* Because spores overwinter in debris all infected debris should be removed. Trim and remove infected plant parts.
* Do not till the debris into the soil or use in the compost pile.
* Space plants far enough apart to increase air circulation and reduce humidity.

#### Downy Mildew

Immagine che contiene pianta, albero

Descrizione generata automaticamenteDowny mildews produce grayish fuzzy looking spores on the lower surfaces of leaves.  To identify downy mildew, look for pale green or yellow spots on the upper surfaces of older leaves.  On the lower surfaces, the fungus will display a white to grayish, cotton-like downy substance. Downy mildew occurs during cool, moist weather such as in early spring or late fall.  Spore production is favored by temperatures below 65°F and with a high relative humidity.

#### **Tips for Downy Mildew Treatment**

* Downy mildew needs water to survive and spread.  It there is no water on your leaves, the disease cannot spread. Keep water off leaves as much as possible.
* Because the disease overwinters on dead plant debris, be sure to clean around your plants in the fall to help prevent the disease in the following spring.

#### Immagine che contiene verde, insetto Descrizione generata automaticamenteBlight

Blight is a fungal disease that spreads through spores that are windborne.  For this reason, spores can cover large areas and rapidly spread the infection.  Blight can only spread under warm humid conditions, especially with two consecutive days of temps above 50°F, and humidity above 90% for eleven hours or more. No cure exists.  Prevention is the only option.

#### **Tips for Preventing Blight**

* If growing potatoes, grow early varieties because blight occurs during mid-summer and you can harvest your crop before the blight.
* Plant resistant varieties:  Sarpo Mira and Sarpo Axona are two varieties that show good resistance. Practice good garden hygiene.
* Destroy any blight-infected plant parts.  Keep the area clean of fallen debris from your diseased plants and discard in the trash.  Do not add to your compost pile.

#### Canker

Immagine che contiene pianta, albero

Descrizione generata automaticamenteCanker is often identified by an open wound that has been infected by fungal or bacterial pathogens.  Some cankers are not serious while others can be lethal.  Canker occurs primarily on woody landscape plants. Symptoms may include sunken, swollen, cracked or dead areas found on stems, limbs or trunk.  Cankers can girdle branches and kill foliage. Cankers are most common on stressed plants that have been weakened by cold, insects, drought conditions, nutritional imbalances or root rot.  Rodents can also spread the pathogens.

#### **Tips for Controlling Canker in Plants**

* Remove diseased parts in dry weather.
* Grow resistant varieties whenever possible.
* Avoid overwatering and overcrowding; avoid mechanical wounds such as damage from lawn mowers.
* Wrap young, newly planted trees to prevent sunscald. Sunscald creates dead patches that form on trunk and limbs of young trees if the trunks have been shaded, then transplanted to sunny areas.
* Keep plants healthy by planting in healthy soils and maintaining nutritional requirements.

#### Shot Hole

This disease of peach, apricot, plum and cherry spreads in warm wet weather infecting buds, blossoms, leaves, fruit and twigs (not large branches). Leaves develop numerous small, tan to purplish spots about 6 mm in diameter that drop out causing a shot hole appearance. Red to purplish spots also form on the fruit and can be accompanied by a clear, gummy substance. Gummy twig and small branch cankers also occur.

#### **Shot Hole Control**

plant resistant varieties. Rake up and destroy fallen leaves and prune out and destroy infected twigs and branches. To prevent twig and bud infections spray with Copper Spray: Peaches after harvest and all other trees in September before fall rains start.

#### Late Blight, Early Blight

Late Blight and Early Blight these are fungal diseases of tomatoes, potatoes and other related plants. Early blight appears as dark brown to black leaf spots with concentric rings. Black spots develop on stems and large, black, leathery, sunken spots on the fruit. Infections often occur in May or June in wet years. Late blight forms irregular greenish black, water-soaked blotches first on older leaves or stems quickly spreading to the fruit. This disease usually doesn’t appear until August in wet years, but it can destroy entire plants overnight.

#### **Late Blight and Early Blight Control**

Space and prune plans for good air circulation. Avoid overhead watering. If Early blight starts to appear, pick off and destroy the infected leaves. If chemical control is required apply a copper spray at 7 to 10 day intervals. If late blight starts to appear remove diseased leaves or entire plants immediately, seal in a plastic bag and send to the landfill. Do not compost late blight infected plants. Apply a copper spray at every 5 to 10 days till allowed days before harvest.

#### Botrytis Blight or Grey Mold

Botrytis Blight or Grey Mold is a grey fuzzy mold develops on dead and dying plant tissue spreading to healthy tissue when conditions are wet. Infections first appears as water-soaked spots or areas on soft or senescent foliage, flower parts and young stems. On flowering plants, woody ornamentals and small fruit this disease can cause flower, leaf and shoot blights as well as stem and fruit rots. Very susceptible plants include: peonies, roses, hostas, strawberries and raspberries.

#### **Botrytis Blight or Grey Mold Control**

Plant resistant cultivars. Thoroughly clean and discard garden debris and refuse in the fall to reduce the level of grey mold in your garden. Susceptible plants (that are sun loving) should be grown in sunny areas with good air circulation. If practical water at the base of plants not over the foliage. If botrytis appears, remove infected leaves and fruit. It is rarely worth applying fungicides to control this disease.

#### Verticilium Wilt

Verticilium Wilt is a serious fungal disease of many deciduous trees, herbaceous perennials, berries and vegetables. It is of particular concern for flowering cherries. It enters roots from the soil moving upwards in the plant, plugging up the plants transportation system. Visible indication that there is a problem starts with yellowing, wilting and dying back of young twigs and branches often on one side of plant or tree. Many other problems look the same, however Verticillium wilt gets worse from year to year. Cutting into a woody stem with a knife reveal black or brown streaks in the wood are vascular cambium just under the bark.

#### **Verticilium Wilt Control**

Control is all preventative as there is no cure once a plant is infected. Avoid drought stress or flooding on mature landscape trees. Remove dead and dying plants including the infested roots and the soil and replant with tolerant or resistant species. When pruning trees that may have this disease, sterilize your pruning tools between trees to prevent spreading it to an and noninfected tree. Rubbing alcohol, Lysol or a 10% household bleach solution (corrosive) can be used to disinfect pruning tools. Once an area is infected with Verticilium Wilt, we generally suggest not planting the same species in that area for several years.

# Conclusions

## Future developments

* Task environment: PEAS (Performance, Environment, Actuators, Sensors)) description.
* Description of the agent: is it fully or partially observable? Fully observable means that the agent does not need to maintain an internal state.
* Find the most appropriate environment variables’ values.
* Identify new plants from captions.
* Exploit knowledge to reach the goal (diagnosis) faster:
  + If readings are out of range, it’s a abiotic problem that can be solved with actuators
  + If captions
    - Nutrient deficiencies
      * Provide nutrients
    - Pests
    - Diseases
* Conditions have 1 - 3 symptoms
* Fully observable environment
  + Simple reflex agent handles actuators
* Implement learning agent
* Generalize predicate names to properly separate the shell from the kb
* Explain the decision tree process to reach the diagnoses