Ch. 11: System management model

Table of Contents

11.1 Introduction

The manager model of Daisy can be considered as a special language that allows for building rather complex scenarios of management actions. The management language comprises two different language elements, viz. direct management actions, described in part 11.3 and conditional statements of the type if-then-else or wait-until, described in section 11.2. The condition statements make it possible to govern the execution of the direct management actions, e.g. only allowing irrigation to take place when a certain condition is fulfilled, for example when the soil water pressure potential at a given depth is below a certain limit and the crop is within a certain development phase of its life cycle. The conditions are described in section 11.2.3. In addition, there are several special actions (described in section 11.4) that makes it possible to combine multiple conditional statements and direct management actions thereby defining new complex actions (see Figure 1).

A management scenario

```
;; Spring Barley setup – the information is stored under the name "SBarley Management" activity

(defaction "SBarley Management" activity
   (wait_mm_dd 3 20)                    ; Wait until March 20 – no actions before this date is reached
   (plowing)                            ; Plow the field
   (wait_mm_dd 4 15)                    ; Wait until April 15
   (seed_bed_preparation)               ; Do a seed bed preparation
   (sow "Spring Barley")                ; Sow a spring barley crop
   (wait_mm_dd 4 20)                    ; Wait until April 20
   (fertilize (N25S (weight 95 [kg N/ha])))  ; Broadcast 95 kg N/ha (fertilizer N25S)
   (wait (or                            ; Wait until one of the following conditions is fulfilled
      (crop_ds_after "Spring Barley" 2.0)     ; Condition 1: the crop is ripe
      (mm_dd 8 20)))                    ; Condition 2: we have reached the date: August 20
   (harvest "Spring Barley"             ; Harvest the crop (Spring Barley)
      (stub 8 [cm])                     ; 8 cm of the stubble is left in the field
      (stem 0.0)                        ; No stem (straw) is removed from the field
      (leaf 0.0))                       ; No leaf is removed from the field.
)
```

Figure 11.1: A management scenario composes of a sequence (activity) of actions.

11.2 Conditional statements

Conditional statements can be divided in two types: wait-until and if-then-else. The wait-until conditional statements include statements to wait for a certain condition, for example a specific crop stage or soil pressure below a specific value, or a certain time or period and are described in section 11.2.1. The if-then-else conditional statement tests whether a certain condition is true and performs a specific action, that could be to cut a grass if the grass is below a certain development stage, these are described in section 11.2.2. The conditions to wait for or test are described in section 11.2.3.

11.2.1 Wait-until

wait

The *wait* action waits until the specified condition is TRUE, the condition can either be a specific time (using the condition *at*) or for example a specific crop development stage (using the condition *crop_ds_after*).

wait_mm_dd

The *wait_mm_dd* action waits until a specific month and day in the year. It takes three parameters *month* (user specified), *day* (user specified) and *hour* (default 8).

wait_period

The *wait_period* action waits a specific period defined by the parameter's *days* (user specified) and *hours* (user specified). In addition, it takes the optional parameter *end_time* that overwrites the days and hours parameters and wait until the specified time.

wait_days

The *wait_days* action waits the number of days specified by the parameter *days*.

wait_hours

The *wait_hours* action waits the number of days specified by the parameter *hours*.

The *at* conditional action statement

In addition to the wait conditional action statements described above, the *at* conditional action statement can be used to specify a specific time (defined by the parameters *year*, *month, day* and *hour*) where a certain action should be performed, specified with the *do* parameter. This is especially useful when simulating field experiments where information on timing and type of field management operations is known.

11.2.2 If-then-else

The if-then-else conditional statement evaluate if a condition is TRUE and perform actions accordingly.

if

The *if* action takes tree parameters *if*, *then* and *else*. Where *if* is a condition component (see section 11.2.3) determining which action to perform, *then* is the action to perform if the condition is TRUE and *else* is the action to perform if the condition is FALSE.

cond

The *cond* action takes a list of *clauses* that each consist of a *condition* and a sequence of *actions*. The action of the first clauses which condition is TRUE will be performed.

assert

The *assert* action evaluates if a certain condition (defined by the *condition* parameter) is TRUE, if not, the simulation is stopped and an error message (defined by the *message* parameter) is printed.

11.2.3 Conditions

The conditions to wait for or test whether is fulfilled are presented below. The conditions can be grouped in conditions related to time, conditions related to crop status, conditions related to soil status (water, temperature and N) and conditions related to weather. In addition, several conditions exist that helps build multiple conditions statements.

The *and* and *or* condition models

The *and* and *or* condition models can be used to combine conditions to test for. Both models test multiple conditions in a sequence. The *and* model tests the conditions in the sequence listed, until a FALSE is found, or the end of the list is reached. If all conditions are fulfilled it returns a TRUE. If at least one conditions is not fulfilled it returns a FALSE. Similarly, the *or* model tests the conditions in the sequence listed, until a TRUE is found, or the end of the list is reached. If at least one condition is fulfilled it returns a TRUE. If all conditions are not fulfilled it returns a FALSE.

The *not* condition model

The *not* condition model takes a condition component and returns a TRUE if the condition is FALSE. Thus, it can be combined with all conditions to test if the condition is not fulfilled.

The *if* condition model

The *if* condition model test whether a certain condition is TRUE. If the *if* parameter condition is TRUE, the model tests the condition specified with the *then* parameter. If the *if* parameter condition is FALSE the model test the condition specified with the *else* parameter. Thus, contrary to the *if* conditional action statement, that performs a certain action based on the evaluation of the *if* parameter condition, the *if* condition model just tests a second condition based on the evaluation of the *if* parameter condition.

The *with* conditions model

The *with* condition model tests the condition, specified with the *condition* parameter on the column specified with the *where* parameter. Thus, is makes it possible to only test certain conditions on specific columns in simulations with multiple columns.

Further, a number of conditions exist that are not used to build management operations but used by the Daisy program to write log-files, check for time, timesteps and write a message. These are *check, extern, false, finished, running, timestep*, *true, secondly, minutely, hourly, daily, weekly, monthly*, *yearly* and *every*. For a definition of these are referred to the reference manual distributed with Daisy.

11.2.3.1 Time conditions

The time conditions are used to control actions that should take place at a specific time. It is used, for example, if simulating a known management plan for a field with known dates and time for sowing, fertilizing, spraying and harvesting. The time conditions can also be used to test if a certain time have passed or to set up recurrent actions.

The *time, at, before* and *after* conditions

The *time* condition describes which time to test for and thus takes the parameter *time* defined by a *Time* component with the parameter's *year, month, mday, hour, minute, second, microsecond.* The first four parameters (*year, month, mday,* and *hour*) are defined by their position and the last four parameters (*hour, minute, second* and *microsecond*) has the default value 0. Thus, the time 01-01-2024 00:00:00:00 can be written as: (Time 2024 01 01 0) or (Time 2024 01 01 0 (minute 0) (second 0) (microsecond 0)). The *at*, *before* and *after* conditions are *time* conditions that are TRUE at, before and after the specified time, respectively.

The *year, month, yday, mday, hour, minute, second* and *microsecond* conditions

The conditions *microsecond, second, minute, hour, mday, yday, month* and *year* all take the parameter *at* specifying which microsecond [0- 999999], second [0-59], minute [0-59], hour [0-23], day in the month [1-31], day in the year [0-366], month [1-12] and year, respectively, the condition is TRUE.

The *mm_dd_base, mm_dd, before_mm_dd* and *after_mm_dd* conditions

The *mm_dd_base* condition is TRUE at a specific month and day in the year. It takes the parameters *month*, *day*, *hour* (default 8), *minute* (default 0), and *second* (default 0). The *mm_dd_base* condition is the base for the conditions *mm_dd*, *before_mm_dd* and *after_mm_dd* that are TRUE at, before and after, respectively, of the specified month, day, hour, minute and second.

11.2.3.2 Crop conditions

The crop conditions test if a crop has reached a certain size or development stage. These are typically used to govern spraying or harvest actions (section 11.3.5 and 11.3.6, respectively).

crop_ds_after

The *crop_ds_after* condition test if the crop have reached a certain development stage (DS). The DS is a counter, and it only has the physical/biological meaning we assign to it. The Daisy development stage is -1 at sowing, 0.01 at emergence, 1 at flowering (or heading in grain crops) and 2 for a mature crop. The scale is continuous from emergence to harvest (see Chapter 10). The *crop_ds_after* condition is TRUE if the crop, specified by the *crop* parameter, has reached the development stages specified by the *ds* parameter.

crop_stage_after

The *crop_stage_after* condition is TRUE if the crop, specified by the *crop* parameter, has reached the phenological development stage specified by the

stage parameter. The *crop_stage_after* condition differs from the *crop_ds_after* condition in the sense that the *stage* parameter refers to the phenological stage of the crop and not the development stages defined by DS [-1.0:2.9] (see Chapter 10). The *stage* would typically be the BBCH stage of the plant. Daisy does not simulate BBCH stages, but a table can be included, linking DS to BBCH-values. So far, such tables have been developed for winter wheat and spring barley (see Chapter 10).

crop_dm_over and crop_dm_sorg_over

The *crop_dm_over* and the *crop_dm_sorg_over* conditions test if the crop, specified by the *crop* parameter, or the storage organ of the crop, respectively, have reached the amount of dry matter specified by the *weight* parameter. In addition, the *crop_dm_over* condition takes a *height* parameter specifying the height above which the DM weight should be tested.

BBCH_after

The *BBCH_after* condition test if the crop have reach as certain BBCH stage. Daisy does not simulate BBCH stages, but a table can be included, linking DS to BBCH-values. So far, such tables have been developed for winter wheat and spring barley (see Chapter 10). The *BBCH_after* condition reads the crop BBCH stage from a file, specified by the *file* parameter, and checks if the crop, specified by the *crop* parameter, has reached the BBCH stages, specified by the *bbch* parameter. In addition, the *BBCH_after* condition takes the parameter *start* that defines if the condition should be TRUE (if start = true) when the crop enters the specified BBCH or if the condition should be TRUE (if start = false) when the crop leaves the specified BBCH stage.

11.2.3.3 Soil Conditions

The soil conditions test the status of the soil in terms of either inorganic N content, temperature, water content or water pressure. Typically, this is done before performing a soil cultivation or sow action.

soil_temperature_above

The *soil_temperature_above* condition tests if the soil temperature at a specified soil depth, defined with the *height* [cm] parameter, is above the temperature defined with the *temperature* parameter [dg C].

soil_water_pressure_ above

The *soil_water_pressure_above* condition tests if the soil pressure potential at a specified soil depth, defined with the *height* [cm] parameter, is above the pressure potential, defined with the *potential* parameter [cm].

soil_water_content_ above

The *soil_water_content_above* condition tests if the soil in a depth interval, specified by the parameter *to* [cm] and *from* [cm] contains more water than the amount specified with the *water* parameter [mm].

soil_inorganic_N_above

The *soil_inorganic_N_above* condition tests if the soil in a depth interval, specified by the parameter *to* [cm] and *from* [cm], contains more mineral nitrogen than the amount specified with the *amount* parameter [kg N ha^{-1}].

The trafficable condition

The *trafficable* condition tests if the soil conditions allow for traffic in terms of soil water content and temperature and are defined in the *tillage.dai* file distributed with Daisy (See Figure 2). Specifically, it combines the conditions *soil_water_pressure_above* and *soil_temperature_above* to tests that the soil

water pressure potential at -10 cm is not above -50 cm and that the soil temperature at -10 cm is above 0 °C.

```
(defcondition trafficable and
   (not (soil_water_pressure_above (height -10.0) (potential -50.0)))
   (soil_temperature_above (height -10.0) (temperature 0.0)))
```

Figure 11.2: The trafficable condition defined in the tillage.dai file distributed with Daisy.

11.2.3.4 Weather conditions

The weather conditions tests whether the air temperature or precipitation fulfil certain criteria.

Tsum_above

The *Tsum_above* condition tests if the temperature sum is above the temperature specified with the *Tsum_limit* [°C d^{-1}] parameter. The temperature sum is reset once a year at the time defined by the parameters *reset_mday* (default 1) and *reset_month* (default 3). The temperature sum is updated daily at the hour defined by the parameter *check_hour* (default 6).

Daily_air_temperature_above

The *daily_air_temperature_above* condition tests if the daily air temperature is above the temperature specified with the *temperature* [°C] parameter.

Daily_precipitation_above

The *daily_precipitation_above* condition tests if the daily precipitation is above the precipitation specified with the *precipitation* [°C] parameter.

11.3 Direct management actions

The direct management actions include all actions needed to cultivate a field from tillage, seedbed preparation and sowing, over irrigation, fertilization and pesticide applications to harvest. These are presented below.

11.3.1 Soil cultivation

The soil cultivation models build on two actions: *mix* or *swap*, defined below.

mix

The *mix* action model mixes the soil content down to the depth specified by the *depth* parameter, such that temperature, water, nitrogen, SOM, pesticides and other substances are averaged in the interval. In addition to depth, the action takes the parameters *random_roughness* (default 0.012), *penetration* (default 1) and *surface_loose* (default equal to *penetration*). The *penetration* parameter defines how big a fraction of the organic matter on the soil surface that should be incorporated in the soil by the *mix* operation. By default, all organic matter on the surface in incorporated in the soil during a *mix* action. The *surface_loose* parameter defined the fraction of surface being loosened by the mix action, by default all. The *random_roughness* [m] parameter defines how organized or random the loosened particles are spread on the soil surface. The *surface_loose* and *random_roughness* parameters can be used if it is wished to include the effect of tillage on soil water dynamics (See Appendix 4.1 part 6).

seed_bed_preparation and rotavation

The *seed_bed_preparation* and the *rotavation* actions are *mix* action models defined in the *tillage.dai* file distributed with Daisy. They have a fixed *depth* of -8 [cm] and -15 [cm], respectively.

stubble_cultivation and disk_harrowing

The *stubble_cultivation* and *disk_harrowing* actions are *mix* action models defined in the *tillage.dai* file distributed with Daisy. They have a fixed *depth* of -8 [cm] and -10 [cm] and a *penetration* fraction of 0.6 [] and 0.8 [], respectively.

swap

The *swap* action model swaps to soil layers, such that temperature, water, nitrogen, SOM, pesticides and other substances are averaged in each layer and the bottom layer is placed on top of what used to be the top layer. The top layer starts at the soil surface and goes down to the depth defined by the parameter *middle*, the bottom layer starts at the middle depth and goes down to the depth defined by the *depth* parameter. In addition, the *swap* action model takes the parameter *random_roughness* (default 0.024 [m]) that, as for the *mix* model, defines the random roughness generated by the tillage operation.

plowing

The *plowing* action is a swap action model defined in the *tillage.dai* file distributed with Daisy. It averages the temperature, water, nitrogen, SOM, pesticides and other substances in the soil layer 0 – 9 cm depth (*middle* = -9 [cm]) and the soil layer 9 – 18 cm depth (*depth* -18 [cm]) and swap the two.

set_porosity

The *set_porosity* action sets the porosity of a soil horizon. The porosity will thus be changed for the complete horizon where the point defined by the with the *depth* parameter is located. The porosity of the soil horizon will continue to be the porosity defined by the *set_porosity* action until a new porosity is defined. The *set_porosity* action should be used in combination with a hydraulic model that support porosity variations (currently the *Mvg_compact* model where α, n and K_{sat} will change with changed porosity, see more in Appendix 4.1 part 6). It can be used in simulations where tillage actions should influence the physical properties of the soil.

set_surface_detention

The *set_surface_detention* action sets the max height for ponding water on the surface before runoff starts. This can be used, for example, to define a changed surface detention capacity after tillage. The surface detention capacity will continue to be the capacity defined by the *set_surface_detention* action until a new detention capacity is defined.

11.3.2 Sowing and emerging

Several sow models controlling the seed amount, rows and plant position on the field exist, all building om the *sow_base* action model.

The *sow_base*, *sow* and *plant* action models

The *sow* and *plant* action models only differ by their names and build on the *sow_base* action model. The actions sow the crop defined by the *crop* parameter with the amount of seed defined by the *seed* parameter. The *seed* parameter [g w.w. m^{-2}] is optional if using the old *"LAI"* function for emergence (see Chapter 10), but the new *"release"* function for emergence (see Chapter 10) requires information about the seed amount. Be aware that the model will not give and warning or error message if the seed amount is not specified, but no crop will be grown as you have sown a crop with no seed. In addition, the *sow* models take several parameters controlling the plants position in the field: *row_width*, *row_position*, *plant_distance* and *plant_position*. The *row_with* and *row_position* define the distance between the rows and the row position on the x-axes,

respectively. By default, they are both 0, giving an equal spread of plants over the simulated area. Similarly, the *plant_distance* and *plant_position* define the distance between the plant and the plant position on the x-axes, respectively. These will overwrite the row placement defined by *row_width* and *row_position*. The only purpose of the *plant_distance* and *plant_position* parameters is to have a more intuitive name for *row_width* and *plant_position* for the 2D simulations, where the x-axis is parallel with the actual rows in the field, rather than orthogonal to the rows as otherwise assumed by Daisy.

emerge

The *emerge* action model forces a crop to emerge. This can be useful when simulating fields with observations of crop development such as time of emergence.

11.3.3 Irrigation

Irrigate_base

The irrigation action models all build on the *irrigation_base* model defining an irrigation action on the field in terms of amount of water, time of irrigation and possible solutes in the irrigation water. The amount of water and type of solutes are defined with the *flux* and *solute* parameters, respectively. The time is defined with the *days* and *hours* parameters. By default, time of irrigation is one hour (*hours* = 1), but if *days* are specified, *hours* will by default be 0 and irrigation time will be the specified by the number of days.

Irrigate_overhead, irrigate_surface and irrigate_subsoil

The *irrigate_overhead, irrigate_surface* and *irrigate_subsoil* are all *irrigate_base* models irrigating the field from above (over canopy height), at the surface (below canopy) or directly in the soil. Thus, the irrigation overhead flux is simulated as irr^{OH} in eq. 3.1 and 3.2 in Chapter 3 increasing the amount of water in the incoming flux above the canopy and potential snow cover; the irrigate surface flux is simulated as irr^{surf} in eq. 3.4 in Chapter 3 increasing the amount of water above the soil surface an potential litter/mulch layer; the irrigate subsoil flux is added as a sink-source term in Richards equation (see eq. 4.1, Chapter 4). An example of an irrigation activity for a winter wheat with overhead irrigation is shown in Figure 3. Be aware that the irrigation models do not have a default unit, so the user should specify a unit, this should be mm h^{-1} or another flux unit that can be converted to mm h^{-1}. Potential solutes, for example N or pesticides, can be added with the irrigation water using the *solute* parameter that takes a sequence of the *name* of the solute and the concentration (*value* [ppm]) of the solute. Like the water flux, the solute in irrigation overhead is simulated as c_{irrOH} [g m^{-2} mm^{-1}] (Eq. 3.47, Chapter 3) describing the concentration in overhead irrigation above the canopy and potential snow cover; the solute in irrigation surface is simulated as $c_{irrsurf}$ [g m^{-2} mm^{-1}] (Eq. 3.54, Chapter 3) as the concentration in surface irrigation water above the soil surface and potential litter/mulch layer; and the solute in irrigation subsoil is added as a source-sink term in the advection-dispersion equation (Chapter 6).

In addition, the temperature of the irrigation water can be specified with the *temperature* parameter (default equal to air temperature). Lastly, the *irrigate_subsoil* action takes the extra parameter *volume* specifying the volume of

soil to add irrigation to. The *volume* parameter is by default defined as a *box* component with the parameters *top, bottom, left, right, front* and *back*.

```
(defaction "Irrigation Winterwheat" activity
  (wait (and (after_mm_dd 05 01)
             (before_mm_dd 09 01)
  (not (soil_water_pressure_above (height -25 [cm])(potential -300 [cm])))))
  (irrigate_overhead 10 [mm/h] (hours 3))
  (wait_days 3)
)
```

Figure 11.3: An irrigation activity including the irrigate_overhead action defined for a winter wheat.

11.3.4 Fertilization

The fertilizer action models fertilize with either a *mineral* fertilizer or an *organic* fertilizer, typically manure or slurry. The fertilizers are *am* components that in general describes various kinds of fertilisers and other added matter such as crop residues.

Mineral fertilizers

A *mineral* fertilizer is defined with a name, an NH_4^+-N-fraction, defined with the *NH4_fraction* [] parameter and a volatilization fraction, defined with the *volatilization* [] parameter. The amount that is not NH_4^+-N is expected to be NO_3^--N. A number of mineral fertilizers have been defined in the *fertilizer.dai* fil distributed with Daisy (see Figure 3). The defined fertilizers do not have volatilization specified. To introduce volatilization of the ammonium fraction, the user must add a line, e.g. (volatilization 0.05 []), in the fertilizer definition (see Chapter 7).

```
(defam NPK02 mineral
  (description "Various NPK fertilizers")
  (NH4_fraction 0.57 []))

(defam NP mineral
  (description "Typical NP or NPS fertilizer")
  (NH4_fraction 0.650 []))
```

Figure 11.4: Example of parameterized mineral fertilizers from the fertilizer.dai file distributed with Daisy.

Organic fertilizers

An *organic* fertilizer is defined with a name, and the dry matter fraction, the total C-fraction (of dry matter), the total N-fraction (of dry matter) and the fractions of NH_4^+ and NO_3^-. The N that is not mineral, is assumed to be organic. The fraction of ammonium that will volatilize should be specified. It is also necessary to specify how the organic material should be distributed between different AOM-pools, that is the name of each pools, the fraction of C allocated to the pools (total number of pools -1), the C/N-ratio governing the distribution of N for the (total number of pools -1), and the turn-over rate for each pool.

For organic fertilizers it is also possible to specify an expected 1st year and 2nd year utilization (mineral fertilizer equivalency). In Denmark, these values are specified in the legislation for different types of slurry and manure. The 1st year utilization can then be used to calculate "backwards" from the amount of effective N required on the soil for a specific crop to the total amount of organic fertilizer

applied. 2nd year utilization can be used to calculate an expected contribution from organic fertilizer to N supply of next year's crop.

A range of organic fertilizer materials are described in the reference manual under *am* and in the *fertilizer.dai* file and *dk-fertilizer.dai* file, all distributed with Daisy. Additional parameterisations can be found on the Daisy website under contributions.

For more on organic fertilizers see Chapter 9.

Fertilize_base, fertilize and incorporate_fertilizer

The fertilize action models (*fertilize* and *incorporate_fertilizer*) both build on the *fertilize_base* action and apply fertilizer to the soil surface or subsoil, respectively. An example of a fertilizer action, fertilizing the surface with ammonium nitrate and pig slurry after sowing of a winter wheat is shown in Figure 5.

```
(sow "Winter Wheat")
(wait_mm_dd 3 15)
   (fertilize (AmmoniumNitrate (weight 106 [kg N/ha])))
(wait_mm_dd 5 1)
   (fertilise (pig_slurry (weight 21932 [kg w.w./ha])))
```

Figure 11.5: Fertilization on the soil surface with ammonimum nitrate and pig slurry after sowing of winter wheat.

The fertilizer action model takes the parameter *am* defining the fertilizer to apply (e.g. type of mineral or organic fertilizer to apply). For the ammonium nitrate, the weight of added N is specified as (weight 106 [kg N/ha]), while the slurry is specified as (weight 21932 [kg w.w.ha]) in the example.

Equivalent weight

For organic fertilizers the parameter *equivalent_weight* [kg N ha^{-1}] can be used instead of *weight* to calculate the amount of added N and dry matter. The calculates can be done at three levels increasing in complexity:

1. *equivalent_weight* [kg N ha^{-1}] is used to calculate the total amount of added dry matter assuming 100 % utilization of N.
2. *equivalent_weight* [kg N ha^{-1}] is used together with a specified *first_year_utilization* [fraction] to calculate the total amount of added N and dry matter.
3. *equivalent_weight* [kg N ha^{-1}] is used together with a specified *first_year_utilization* [fraction] and a *second_year_utilization* [fraction] to calculate the total amount of added N and dry matter, as well as the second-year effect. Note that the *second_year_utilization*, *second_year_compensate* and second-year effect do not refer to the second calendar year, but just to the next fertilization.

For level one and two total N and dry matter added is calculated as:

$$Total\ N\ added\ [\text{kg N ha}^{-1}] = \frac{equivalent\ weight\ [\text{kg N ha}^{-1}]}{1.\ year\ utilization} \qquad 11.1$$

$$Total\ dry\ matter\ [\text{kg DM ha}^{-1}] = \frac{total\ N\ [\text{kg N ha}^{-1}]}{total\ N fraction} \qquad 11.2$$

Where the total N-fraction is specified for the fertilizer type. By default, *first_year_utilization* is one and the total added N will be equal to the equivalent weight. If the organic fertilizer has specified a first-year utilization factor this will be used to calculate the total added N (level two). If the organic fertilizer in addition has specified a second-year utilization factor the "second-year effect" will increase with the second-year utilization factor at each fertilization. Then, if a later fertilization action is specified with the *second_year_compensation* parameter sat to TRUE, total added N will be calculated as:

$$Total\ N\ added\ [\text{kg N ha}^{-1}] = \frac{equivalent\ weight\ [\text{kg N ha}^{-1}]}{1.\ year\ utilization} - 2.\ year\ effect\ [\text{kg N ha}^{-1}] \quad 11.3$$

The effect of the three calculation levels on applied N and DM are shown below for an organic fertilizer with a N fraction of 0.1 (Table 11.1). In the third calculation level, the fertilizer is added in a three-split application and in the last application is *second_year_compensate* sat to TRUE. The accumulated second year effect from the two previous fertilizations is therefore subtracted from the total added N (Eq. 11.3) and the second-year effect is zeroed.

Table 11.1: Calculation of added N and DM using equivalent weight (level 1), equivalent and first year utilization (level 2) and equivivalent weight, first year utilization and second year utilization in a three split application (level 3).

Calculation level		Eq. Weight	1. year utilization	2. year utilization	Total N	Total DM	2. year compensate	2. year effect
		kg N/ha	fraction	fraction	kg N/ha	kg DM/ha		kg N/ha
Level 1		170.0	1.0	0.0	170.0	1700.0	FALSE	0.0
Level 2		170.0	0.8	0.0	212.5	2125.0	FALSE	0.0
Level 3	1. application	56.7	0.8	0.1	70.8	708.3	FALSE	7.1
	2. application	56.7	0.8	0.1	70.8	708.3	FALSE	14.2
	3. application	56.7	0.8	0.1	56.7	566.7	TRUE	0.0

Precision fertilization

Fertilization in precision agriculture can be simulated using the *fertilizer_base* submodel *precision*. In the model the amount of applied N can be calculated based on a target N amount (*target* [kg N ha^{-1}]) and the N content present in the soil layer defined with by the parameter *height* [cm] (default 0) and *to* [cm] (default -100). Then N will be applied so the N content in the soil layer will correspond to the target N.

Minimum N

To ensure that there is not fertilized with unrealistic small amounts a minimum amount of N to fertilize with can be specified with the *minimum_weight* [kg N ha^{-1}] parameter. If the calculated fertilization amount (when including *second_year_utilization* or by the difference in target N and N content when

simulating precision agriculture) is smaller than the *minimum_weight* [kg N ha^{-1}] fertilization will not be simulated.

incorporate_fertilizer

The *incorporate_fertilizer* action model takes an additional parameter (*volume*) defining the soil volume to incorporate fertilizer in to. The *volume* parameter is by default defined as a volume *box* component with the parameters *top, bottom, left, right, front* and *back*.

For the fate of applied N and dry matter after fertilization is referred to Chapter 7 and Chapter 9, respectively.

11.3.5 Spray

The *spray* and *spray_surface* action models apply a chemical (typically a pesticide) on the field either above (*spray*) or below (*spray_surface*) the canopy. Both models take two parameters: *chemical* specifying the name of the chemical and *amount* [g ha^{-1}] specifying the amount of chemical to apply. In reality, spraying of pesticides will typically include water and solute. The amount of water allocated during pesticide application is typically around 200 L/ha or 0.02 mm, hence the amount of water applied during spraying is considered negligible. If larger amounts of water are applied with the solute, the operation can be defined as irrigation instead (see section 11.3.3).

A chemical

A chemical can be defined with different parameters depending on the relevant processes for the considered chemical. In general, for the above-ground processes, the most important factors are solubility, canopy dissipation rate (or half-life) and canopy wash-off coefficient, litter decomposition rate (or half-life) and wash-off coefficient and surface decomposition rate (or half-life). The most important parameters for solute fate in the soil are decomposition, sorption, diffusion coefficient and the crop uptake reflection factor. Several chemicals are predefined in the *chemistry-base.dai* and *chemistry.dai* files distributed with Daisy. For a full description of the definition of a chemical and a pesticide is referred to Chapter 6 and Chapter 8.

11.3.6 Harvest

Harvest_base, harvest and cut

The *harvest_base, harvest* and *cut* action models are all names for the same harvest model. The harvest models harvest the crop specified by the *crop* parameter. If *crop* is specified with “all”, all crops on the field will be harvest. The fraction of the crop that is harvested is specified for the storage organ, leaf and stem. For storage organ the *sorg* parameter (default 1) is the fraction of the total storage organ that is harvested, while the *leaf* parameter (default 1) and *stem* parameter (default 1) specify the harvested fraction of stem and leave above the stub. The stub height is specified with the *stub* parameter (default 0 [cm]). The leaves and stem below the stub height are left on the field. In Figure 6 is an example of a harvest action for winter wheat. In the example, the harvest action is performed either when the crop DS is above 2 or 20th of August. 99 % of the storage organ is harvested, together with 70 % of the leaf and 70 % of the stem above the stub height (10 cm). Leaf and stem below 10 cm are left on the field.

```
(wait (or (crop_ds_after "Winter Wheat" 2.0)
          (mm_dd 8 20)))
          (harvest "Winter Wheat"
          (sorg 0.99) (leaf 0.7) (stem 0.7) (stub 10)))
```

Figure 11.6: Example of a harvest action for a winter wheat

The harvest models take an additional parameter (combine, default FALSE) specifying if the harvest amount should be combined in the stem column in the log-file. This can be relevant when harvesting clover grass or other crops for feed.

Pluck

The *pluck* model is a harvest action that allows you to pluck selected parts of the above ground biomass (default *sorg* 1 and *stem* and *leaf* 0) without killing the crop. It is intended for crops like tomatoes, that are harvested multiple times.

11.3.7 Other management actions

In addition, a number of management actions exist that can be used for meta-modelling for example when coupling Daisy with other programs such as Mike-She or OpenMI or when combining Daisy with automatic parameterization tools or other models. These are *extern, extern_ferigation, extern_subsoil*, *sow_name* and *sow_object*.

crop

The *crop* action model lets you manage a specific crop or multi crop (primary and secondary crops) in terms of tillage, sowing, spraying, fertilization, irrigation and harvesting of both annual and perennial crops. It is intended to be used for simulating fields with mixed cropping. For a description of the parameters of the *crop* action model, please go to the reference manual distributed with Daisy.

11.4 Combine actions

The direct management actions can be combined with the *while, repeat, progn, sequen*ce and *activity* models, where *activity* (see Figure 1 and Figure 3), *repeat* and *while* (see Figure 5) are the most common used.

activity and sequence

The *activity* action model is a *sequence* model that takes a sequence of actions and performs them in the sequence listed. Each action is performed until done. Only one action can be performed at each time step. The *activity* action model is used to build complete management scenarios with multiple wait conditional statements (Section 11.2) and direct management actions (Section 11.3) as done in Figure 1.

while

The *while* action model performs multiple actions in parallel. A typical example would be if crop management and an irrigation scheme are defined separately, as done in Figure 7. With *while*, the two operations are combined into one irrigated crop management. The duration of the *while* action is identical to the duration of the first of the actions it combines.

repeat

The repeat action model performs a single action repeatedly until the simulation ends, or if it is inside a *while* action, as in Figure 7, until the first nested action in the *while* action model ends. The repeat action can be an activity including one direct management action, as in Figure 7, or it can be a complete management scenario, as in Figure 1, performing multiple actions over multiple timesteps. The

latter can be useful in long-term simulations by repeating a single crop or crop rotation for the whole duration.

```
(defection "A irrigated Winterwheat" activity
        (while "A Winterwheat" (repeat ("Irrigation Winterwheat")))
```

Figure 11.7: An example of an activity combining the winter wheat activity (earlier defined sowing, fertilization and harvest of winter wheat) with an earlier defined irrigation activity with the while action.

Progn

The *progn* action model performs all the specified actions in the sequence listed in the same time step.

with-column

The *with-column* action model makes it possible to specify that certain actions (defined with the *actions* parameter) should only be performed on a specific column (defined with the *column* parameter).

table

The *table* action model reads the management actions from a Daisy data file (ddf). In the ddf can be specified date of sowing, emerging, fertilization, irrigation, and harvest. For further information on how to specify the management actions in the ddf see the reference manual distributed with Daisy.

11.5 Parameter overview

Table 11.2. Related Parameter names in Daisy.

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
CONDITIONAL STATEMENTS					
	Condition to wait for.	wait	*condition*	User specified	Condition component
month	Month to wait for.	wait_mm_dd	*month*	User specified	[month]
day	Day in month to wait for.	wait_mm_dd	*day*	User specified	[day]
hour	Hour to wait for.	wait_mm_dd	*hour*	Default 8	[hour]
days	Number of days to wait	wait_period	*days*	User specified	[day]
hours	Number of hours to wait	wait_period	*hours*	User specified	[hour]
end_time	Wait until the specified time.	wait_period	*end_time*	User specified	Time component
days	Number of days to wait.	wait_days	*days*	User specified	[day]
	Number of hours to wait.	wait_days	*hours*	Default 0	[hour]
hours	Number of hours to wait.	wait_hours	*hours*	User specified	[hour]
	Number of days to wait.	wait_hours	*days*	Default 0	[day]
hour	Hour to perform action.	at	*hour*	Default 8	[hour]
day	Day in month to perform action.	at	*day*	User specified	[day]
month	Month to perform action.	at	*month*	User specified	[month]
year	Year to perform action.	at	*year*	User specified	[year]
do	Action to perform	at	*do*	User specified	Action component
if	Condition determining which action to perform.	if	*if*	User specified	Condition component

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
else	Action to perform if condition is true.	if	*else*	User specified	Action component
then	Action to perform if condition is false.	if	*then*	User specified	Action component
condition	Condition determining which action to perform.	clauses (cond)	*condition*	User specified	Condition component
actions	Actions to perform if condition is true.	clauses (cond)	*action*	User specified	Action component
condition	Condition determining which action to perform.	assert	*condition*	User specified	Condition component
message	Error message to print if condition is false.	assert	*message*	Default "Required condition not fulfilled"	[]
CONDITIONS					
	Conditions to test.	and, or, not	*operand*	User specified	Condition component
if	Condition to test.	if	*if*	User specified	Condition component
then	Condition to test.	if	*then*	User specified	Condition component
else	Condition to test.	if	*else*	User specified	Condition component
where	Column to test condition on.	with	*where*	User specified	[]

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
condition	Condition to test.	with	*condition*	User specified	Condition component
time	Time to test if TRUE	time, at, before and after	*time*	User specified for year month and day. Default 0 for hour, minute, second and microsecond.	Time component
year	The year of the time.	Time	*year*	User specified integer	[year]
month	The month of the time.	Time	*month*	User specified integer	[month]
mday	The day in the month of the time.	Time	*mday*	User specified integer	[mday]
hour	The hour of the time.	Time	*hour*	Integer, Default 0	[hour]
minute	The minute of the time.	Time	*minute*	Integer, Default 0	[minute]
second	The second of the time.	Time	*second*	Integer, Default 0	[second]
microsecond	The microsecond of the time.	Time	*microsecond*	Integer, Default 0	[microsecond]
at	Microsecond when condition is true.	microsecond	*at*	User specified	[microsecond]
at	Second when condition is true.	second	*at*	User specified	[second]
at	Minute when condition is true.	minute	*at*	User specified	[minute]
at	Hour when condition is true.	hour	*at*	User specified	[hour]
at	Day of month when condition is true.	mday	*at*	User specified	[mday]

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
at	Day of year when condition is true.	yday	*at*	User specified	[yday]
at	Month when condition is true.	month	*at*	User specified	[month]
at	Year when condition is true.	year	*at*	User specified	[year]
microseconds	Number of microseconds.	Every	*microseconds*	Default 0	[microsecond]
seconds	Number of seconds.	Every	*seconds*	Default 0	[second]
minutes	Number of minutes.	Every	*minutes*	Default 0	[minutes]
hours	Number of hours.	Every	*hours*	Default 0	[hour]
days	Number of days.	Every	*days*	Default 0	[day]
next	Time for next match.	Every	*next*	User specified	Time compontent
second	Second to test if TRUE.	mm_dd_base, mm_dd, before_mm_dd, after_mm_dd	*second*	Default 0	[second]
minute	Minute to test if TRUE.	mm_dd_base, mm_dd, before_mm_dd, after_mm_dd	*Minute*	Default 0	[minute]
hour	Hour to test if TRUE.	mm_dd_base, mm_dd, before_mm_dd, after_mm_dd	*hour*	Default 8	[hour]
day	Day in month to test if TRUE.	mm_dd_base, mm_dd,	*day*	User specified	[mday]

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
		before_mm_dd, after_mm_dd			
month	Month to test if TRUE.	mm_dd_base, mm_dd, before_mm_dd, after_mm_dd	*month*	User specified	[month]
crop	Name of crop of the field to test.	crop_ds_after	*crop*	User specified. Specify “all” to us a combined weight of all crops on the field.	[]
ds	Development stage to test if reached.	crop_ds_after	*ds*	User specified.	[DS [-1.0:2.0]]
crop	Name of crop of the field to test.	crop_stage_after	*crop*	User specified. Specify “all” to us a combined weight of all crops on the field.	[]
stage	Development stage to test if reached.	crop_stage_after	*stage*	User specified. Specify “all” to us a combined weight of all	[]

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
				crops on the field.	
crop	Name of crop of the field to test.	crop_dm_over, crop_dm_sorg_over	*crop*	User specified. Specify “all” to us a combined weight of all crops on the field.	[]
weight	Amount of dry-matter required for the condition to be TRUE.	crop_dm_over, crop_dm_sorg_over	*weight*	User specified.	[kg DM ha^{-1}]
height	Height above which dry matter should be checked	crop_dm_over	*height*	Default 0	[cm]
crop	Name of crop to check BBCH for.	BBCH_after	*crop*	User specified	[]
file	Name of file with information converting the Daisy calculated DS stage of crop to BBCH stage (see Chapter 10).	BBCH_after	*file*	User specified	[]
start	If true, the interval begins when the crop enters the specified BBCH. If false, the interval begins when the crop leaves the specified BBCH.	BBCH_after	*start*	User specified	boolean, TRUE or FALSE
bbch	BBCH value for which the condition is TRUE.	BBCH_after	*bbch*	User specified	[]

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
height	Soil depth in which to test the temperature.	soil_temperature_above	*height*	User specified	[cm]
temperature	Lowest soil temperature for which the condition is TRUE.	soil_temperature_above	*temperature*	User specified	[°C]
height	Soil depth in which to test the pressure potential.	soil_water_pressure_above	*height*	User specified	[cm]
potential	Lowest soil pressure potential for which the condition is TRUE.	soil_water_pressure_above	*potential*	User specified	[cm]
to	Bottom of interval to test soil water content in.	soil_water_content_above	*to*	User specified	[cm]
from	Top of interval to test soil water content in.	soil_water_content_above	*from*	User specified	[cm]
water	Lowest water content for which the condition is TRUE.	soil_water_content_above	*water*	User specified	[mm]
to	Bottom of interval to test mineral N in.	soil_inorganic_N_above	*to*	User specified	[cm]
from	Top of interval to test mineral N in.	soil_ inorganic_N_above	*from*	User specified	[cm]
amount	Lowest mineral N content for which the condition is TRUE.	soil_ inorganic_N_above	*amount*	User specified	[kg N ha^{-1}]
Tsum_limit	The temperature sum above which the condition is TRUE	Tsum_above	*Tsum_limit*	User specified	[°C day]
reset_mday	Day in the month to reset temperature sum	Tsum_above	*reset_mday*	Default 1	[day]
reset_month	Month to reset temperature sum	Tsum_above	*reset_month*	Default 3	[month]

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
check_hour	Hour in day to update temperature sum	Tsum_above	*check_hour*	User specified	[hour]
temperature	Lowest temperature for which the conditions is TRUE	daily_air_temperature_above	*temperature*	User specified	[°C]
precipitation	Lowest precipitation for which the conditions is TRUE	daily_precipitation_above	*precipitation*	User specified	[mm]
MANAGEMENT ACTIONS					
depth	The depth below surface to mix the soil (a negative number)	mix, seed_bed_preparation stubble_cultivation disk_harrowing rotavation	*depth*	User specified -8 -8 -10 -15	[cm]
random_roughness	Random roughness generated by the mix action.	mix, seed_bed_preparation, stubble_cultivation, disk_harrowing, rotavation	*random_roughness*	Default 0.0012	[m]
penetration	Fraction of organic matter on surface incorporated by the mix action.	mix, seed_bed_preparation, stubble_cultivation, disk_harrowing, rotavation	*penetration*	Default 1 Default 1 0.6 0.8 Default 1	[fraction]

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
surface_loose	Fraction of surface loosned by the mix action.	mix, seed_bed_preparation, stubble_cultivation, disk_harrowing, rotavation	*surface_loose*	Default equal to *penetration*.	[fraction]
middle	The depth of the top layer to swap.	swap, plowing	*middle*	User specified -9	[cm]
depth	The depth of the bottom layer to swap.	swap, plowing	*depth*	User specified -18	[cm]
random_roughness	Random roughness generated by the swap action.	swap, plowing	*random_roughness*	Default 0.024	[m]
depth	A depth in the soil horizon to modify.	set_porosity	*depth*	Default 0	[cm]
	The porosity (non-solid fraction) of the soil.	set_porosity	*porosity*	User specified	[fraction]
	Max ponding height before runoff begins.	set_surface_detention_capacity	*height*	User specified	[cm]
crop	The crop to sow.	sow, plant, sow_object	*crop*	User specified	Crop component
seed	Amount of seed applied. Required if using the new “release” function for emergence (see Chapter 10)	sow_base, sow, plant, sow_name, sow_object	*seed*	User specified	[g w.w. m^{-2}]

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
row_width	Distance between rows.	sow_base, sow, plant, sow_name, sow_object	*row_width*	Default 0	[cm]
row_position	Position of row on x-axes.	sow_base, sow, plant, sow_name, sow_object	*row_position*	Default 0	[cm]
plant_width	Distance between plants. Overwrites *row_width*.	sow_base, sow, plant, sow_name, sow_object	*plant_width*	User specified	[cm]
plant_position	Position of plant on x-axes. Overwrites *row_position*.	sow_base, sow, plant, sow_name, sow_object	*plant_position*	User specified	[cm]
crop	The name of the crop to sow.	sow_name	*crop*	User specified	(string)
flux	The amount of irrigation applied	irrigate_base, irrigate_overhead, irrigate_surface, irrigate_subsoil	*flux*	User specified	[mm h^{-1}]
solute	Potential solutes in irrigation water given as sequence with name of solute and concentration of solute [ppm]	irrigate_base, irrigate_overhead, irrigate_surface, irrigate_subsoil	*solute*	User specified	Sequence: name (string) and value [ppm]
hours	Number of hours to irrigate.	irrigate_base, irrigate_overhead,	*hours*	Default 1, but if days > 0, defaults to 0.	[hour]

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
		irrigate_surface, irrigate_subsoil			
days	Number of days to irrigate	irrigate_base, irrigate_overhead, irrigate_surface, irrigate_subsoil	*days*	Default 0	[day]
remaining_time	Number of hours to irrigate. Overwrites *days* and *hours*.	irrigate_base, irrigate_overhead, irrigate_surface, irrigate_subsoil	*Remaining_time*	User specified	[hour]
temperature	The temperature of the irrigation water.	irrigate_base, irrigate_overhead, irrigate_surface, irrigate_subsoil	*temperature*	Default equal to air temperature	[°C]
volume	The soil volume where subsoil irrigation is added.	Irrigate_subsoil	*volume*	*box* component	[]
	Upper boundary on the z-axis.	box	*top*	none	[]
	Lower boundary on the z-axis.	box	*bottom*	none	[]
	Lower boundary on the x-axis.	box	*left*	none	[]
	Upper boundary on the x-axis.	box	*right*	none	[]
	Lower boundary on the y-axis.	box	*front*	none	[]
	Upper boundary on the y-axis.	box	*back*	none	[]
am	The fertilizer to apply.	fertilize_base, fertilize,	*am*	User specified	am component

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
		incorporate_fertilizer			
equivalent_weight	The amount of fertilizer to apply.	fertilize_base, fertilize, incorporate_fertilizer	*equivalent_weight*	User specified	[kg N ha^{-1}]
minimum_weight	The minimum amount of fertilizer to apply.	fertilize_base, fertilize, incorporate_fertilizer	*minimum_weight*	Default 0	[kg N ha^{-1}]
second_year_compensation	Compensate for the second-year effect of previous fertilizations.	fertilize_base, fertilize, incorporate_fertilizer	*second_year_compensation*	Defalt FALSE	Boolean FALSE or TRUE
precision	The amount of fertilizer to apply is calculated based on N in soil.	fertilize_base, fertilize, incorporate_fertilizer	*precision*	User specified	precision submodel
target	The target N amount in the soil.	precision (fertilize_base, fertilize, incorporate_fertilizer)	*target*	User specified	[kg N ha^{-1}]
from	The top of the soil layer to calculate N in. Negative number.	precision (fertilize_base,	*from*	Default 0	[cm]

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
		fertilize, incorporate_fertilizer)			
to	The bottom of the soil layer to calculate N in. Negative number.	precision (fertilize_base, fertilize, incorporate_fertilizer)	*to*	Default -100	[cm]
chemical	The name of the chemical to spray.	spray, spray_surface	*chemical*	User specified	[]
amount	The amount of chemical to spray.	spray, spray_surface	*amount*	User specified	[g ha^{-1}]
crop	The crop to harvest.	harvest_base, harvest, cut, pluck	*crop*	User specified. Specify “all” to harvest all crops of the field.	[]
sorg	Fraction of storage organ to harvest.	harvest_base, harvest, cut, pluck	*sorg*	Default 1	[fraction]
leaf	Fraction of leaf above stub to harvest.	harvest_base, harvest, cut, pluck	*leaf*	Default 1 Default 0	[fraction]
stem	Fraction of stem above stub to harvest.	harvest_base, harvest, cut, pluck	*stem*	Default 1 Default 0	[fraction]
stub	The height of the stub left on the field.	harvest_base, harvest, cut	*stub*	Default 0	[cm]
combine	Combine all harvest crop parts into stem in the log file.	harvest_base, harvest, cut	*combine*	Default FALSE	Boolean TRUE or FALSE

Name and explanation		Model (in Daisy)	Parameter name (Daisy reference manual)	Default	Default unit
	Sequence of actions to perform.	sequence, activity,	*do*	User specified	Action component
actions	Sequence of actions to perform.	while, progn, with-column	*actions*	User specified	Action component
	The action to perform repeatedly.	repeat	*action*	User specified	Action component
column	Name of column to perform actions on.	with-column	*column*	User specified	[]

Original text from		
Updated by	Date	For Daisy version
Holbak, M & Styczen, M	2025 20 01	7.00