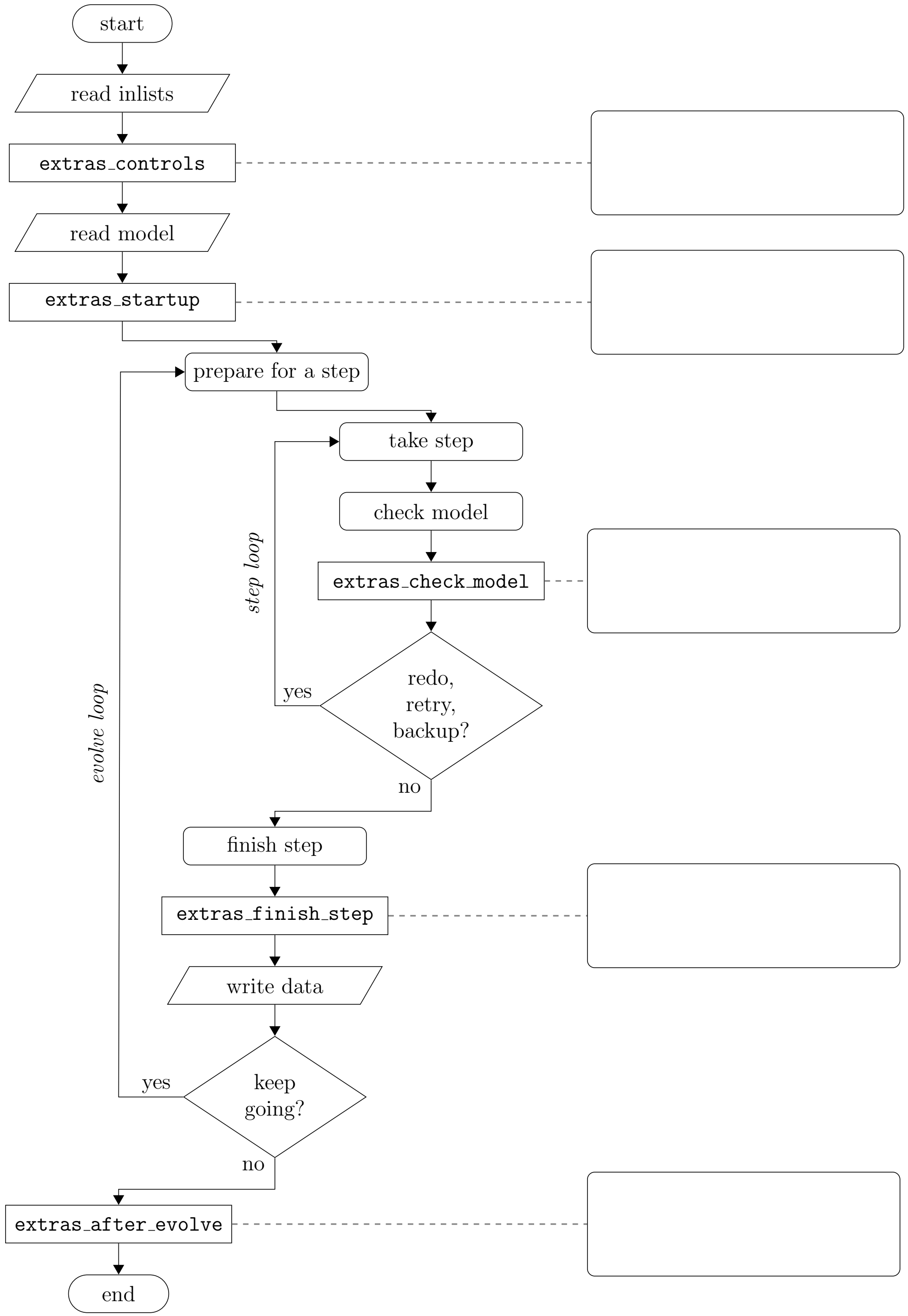
run_star_extras

run_star_extras gives you the possibility to add commands that aren't already available. Moreover, you can use it to override MESA's built-in physics routines.

Ilaria Caiazzo



MESA

The different run_star_extras.f routines get called at different points during MESA execution.

They give you hooks to customize the software at different stages of the execution.

Ilaria Caiazzo

Example

You want the run to terminate when the star reaches a certain radius. If you look at the previous flowchart you can see that the correct hook for that is extras_finish_step

- You can use the same folder you were using for pgplot, 1M_pre_ms_to_wd.
- open src/run_star_extra.f
- If you look at the subroutines and functions defined, they are the same listed in the flowchart.
- look for extras_finish_step
- The only output possible by default is keep_going.

We want to add a check on the star's radius and add a possible new output that terminates the execution.

Example

But how do we do that? First, we have to find out how to check the radius' value. The star_info structure contains all the information about the star that is being evolved. By convention, the variable name s is used throughout run_star_extras.f to refer to this structure. In Fortran, the percent (%) operator is used to access the components of the structure. (So you can read s% x = 3 in the same way that you would read s.x = 3 in C.).

The star_info structure contains the stellar model itself (i.e., zoning information, thermodynamic profile, composition profile). These components are listed in the file $MESA_DIR/star/public/star_data.inc. In addition, star_info contains the values for the parameters that you set in your controls inlist (i.e., initial_mass, xa_central_lower_limit). Recall that the list of controls is located in $MESA_DIR/star/defaults/controls.defaults.

- open up star/public/star_data.inc and start looking around.
- if you search for the word radius, you'll see that MESA says "r(k) is radius at outer edge of cell k". (In MESA, the outermost zone is at k=1 and the innermost zone is at k=s% nz.) Therefore, the radius of the star is s% r(1).

MESA uses cgs units unless otherwise noted. The most common non-cgs units are solar units. MESA defines its constants in $MESA_DIR/const/public/const_def.f. Since the run_star_extras module includes the line use const_def, we will be able to access these values. Using the built in constants lets us make sure we're using exactly the same definitions as MESA. The constant with the value of the solar radius (in cm) is named Rsun.

MESA

Example

Now we check the value of the radius at the end of each step and, if it's greater than 1.2 Rsun, we tell MESA to terminate the run:

```
! returns either keep_going or terminate.
! note: cannot request retry or backup; extras_check_model can do that.
integer function extras_finish_step(id, id_extra)
   integer, intent(in) :: id, id_extra
   integer :: ierr
   type (star_info), pointer :: s
   ierr = 0
   call star_ptr(id, s, ierr)
   if (ierr /= 0) return
   extras_finish_step = keep_going
   call store_extra_info(s)

   ! stop when the star grows larger than 1.2x solar radii
   if (s% r(1) > 1.2 * Rsun) extras_finish_step = terminate

   ! to save a profile,
      ! s% need_to_save_profiles_now = .true.
   ! to update the star log,
      ! s% need_to_update_history_now = .true.

   ! see extras_check_model for information about custom termination codes
   ! by default, indicate where (in the code) MESA terminated
   if (extras_finish_step == terminate) s% termination_code = t_extras_finish_step
end function extras_finish_step
```

Ilaria Caiazzo

MESA

Example

If you try it out, the run will end immediately because the model during the initial relaxation part and also during the pre-main sequence has a radius bigger than 1.2 solar radii. If you try 30 Rsun, the run should stop towards the end of the RGB.

MESA

Assignment: changing neutrino production from inlist

You can communicate with run_star_extras from the inlist. There is one set of controls that will prove useful time and time again when using run_star_extras.f and that is x_ctrl, x_integer_ctrl, and x_logical_ctrl. These are arrays (of length 100 by default) of double precision, integer, and boolean values. You can set the elements in your inlists

```
&controls
  x_ctrl(1) = 3.14
  x_ctrl(2) = 2.78
  x_integer_ctrl(1) = 42
  x_logical_ctrl(1) = .true.
/ ! end of controls inlist
```

and access them later on as part of the star structure (i.e., s% x_ctrl(1), etc.).

Assignment: changing neutrino production from inlist

The assignment asks you to Write your own run_star_extras.f so that you can vary the neutrino production rates by a constant factor that you can give in your inlist file.

MESA provides hooks to override its built-in physics routines. These are referred to as "other" routines. There are two main steps needed to take advantage of this functionality: (1) writing the other routine and (2) instructing MESA to use this routine.

- Navigate to $MESA_DIR/star/other, where you will see a set of files named with the pattern other_*.f. In general, find the one corresponding to the physics (or numerics) that you want to alter. Open it up and read through it. Many of the files contain comments and examples.
- Note that we do not want to directly edit these files. Instead we want to copy the template routine into our copy of run_star_extras.f and then further modify it there. The template routines are named either null_other_* or default_other_*.
- For this assignment, you'll have to focus on other_neu.f. Open up this file. Copy the subroutine null_other_neu and paste it into your run_star_extras.f. It should be at the same "level" as the other subroutines in that file (that is, contained within the run_star_extras module.). Rename it to something different, like assignment_other_neu.

Ilaria Caiazzo

MESA

Assignment: changing neutrino production from inlist

You can put it after contains for example:

```
contains

subroutine assignment_other_neu(  &
      id, k, T, log10_T, Rho, log10_Rho, abar, zbar, z2bar, log10_Tlim, flags, &
      loss, sources, ierr)
   use neu_lib, only: neu_get
   use neu_def
   integer, intent(in) :: id ! id for star
   integer, intent(in) :: k ! cell number or 0 if not for a particular cell
   real(dp), intent(in) :: T ! temperature
   real(dp), intent(in) :: log10_T ! log10 of temperature
   real(dp), intent(in) :: Rho ! density
   real(dp), intent(in) :: log10_Rho ! log10 of density
   real(dp), intent(in) :: abar ! mean atomic weight
   real(dp), intent(in) :: zbar ! mean charge
   real(dp), intent(in) :: z2bar ! mean charge squared
   real(dp), intent(in) :: log10_Tlim
   logical, intent(inout) :: flags(num_neu_types) ! true if should include the type of loss
   real(dp), intent(out) :: loss(num_neu_rvs) ! total from all sources
   real(dp), intent(out) :: sources(num_neu_types, num_neu_rvs)
   integer, intent(out) :: ierr
   call neu_get(  &
      T, log10_T, Rho, log10_Rho, abar, zbar, z2bar, log10_Tlim, flags, &
      loss, sources, ierr)
end subroutine assignment_other_neu
```

Assignment: changing neutrino production from inlist

Then you have to tell MESA to use the other routine. Substitute the subroutine extras_controls with this:

```
subroutine extras_controls(id, ierr)
   integer, intent(in) :: id
   integer, intent(out) :: ierr
   type (star_info), pointer :: s
   ierr = 0
   call star_ptr(id, s, ierr)
   if (ierr /= 0) return

   ! this is the place to set any procedure pointers you want to change
   ! e.g., other_wind, other_mixing, other_energy  (see star_data.inc)
   s% other_neu => assignment_other_neu

end subroutine extras_controls
```

Assignment: changing neutrino production from inlist

You also have to state that you want to use the other routine in your inlist (inlist_1.0), in the controls section:

```
use_other_neu = .true.
```

Ilaria Caiazzo

Assignment: changing neutrino production from inlist

If you compile and run it now, nothing will have changed. In fact, the other_neu routine that you have copied in run_star_extras.f is equal to the default one.

Now you have to modify the other_neu routine in order to change the neutrino loss rate of an amount given in the inlist. You can define the amount in the inlist using x_ctrl(:) in your commands section.

Ilaria Caiazzo

```
subroutine assignment_other_neu(  &
    id, k, T, log10_T, Rho, log10_Rho, abar, zbar, z2bar, log10_Tlim, flags, &
    loss, sources, ierr)
  use neu_lib, only: neu_get
  use neu_def
  type (star_info), pointer :: s
  integer, intent(in) :: id ! id for star
  integer, intent(in) :: k ! cell number or 0 if not for a particular cell
  real(dp), intent(in) :: T ! temperature
  real(dp), intent(in) :: log10_T ! log10 of temperature
  real(dp), intent(in) :: Rho ! density
  real(dp), intent(in) :: log10_Rho ! log10 of density
  real(dp), intent(in) :: abar ! mean atomic weight
  real(dp), intent(in) :: zbar ! mean charge
  real(dp), intent(in) :: z2bar ! mean charge squared
  real(dp), intent(in) :: log10_Tlim
  logical, intent(inout) :: flags(num_neu_types) ! true if should include the type of loss
  real(dp), intent(out) :: loss(num_neu_rvs) ! total from all sources
  real(dp), intent(out) :: sources(num_neu_types, num_neu_rvs)
  integer, intent(out) :: ierr
        call star_ptr(id, s, ierr)
  if (ierr /= 0) then ! OOPS
    return
  end if
  call neu_get(  &
    T, log10_T, Rho, log10_Rho, abar, zbar, z2bar, log10_Tlim, flags, &
    loss, sources, ierr)

  loss = loss*(s% x_ctrl(1))
end subroutine assignment_other_neu
```

MESA

Ilaria Caiazzo