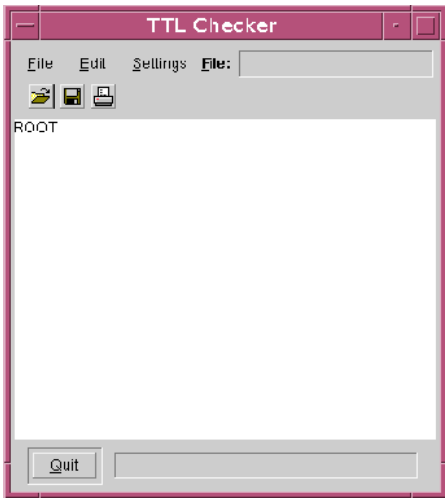
Checking a simple dynamic properties against a "leads-to" trace

This exercise is meant as an introduction to the checker software. For this reason, you will start by checking several simple properties against simple traces. In particular, you are going to check whether certain dynamic properties, expressed in the *Temporal Trace Language* hold for the trace you generated in week 1, describing *stimulus-response behaviour*.

To start, suppose we are interested in the property "*the animal eventually goes to p2*". As such, the property is denoted in an informal format. In semi-formal notation, the property looks as follows:
"*a time point t exists such that at t the animal goes to p2*". Going one step further, in formal notation (the TTL language), the property can be expressed as follows:
`∃t state(trace1,t) |= goes_to_p2`, where *trace1* refers to the trace against which we want to check. In order to check this property, execute the following steps:

a) Before you start, make sure you know where you can find the trace you generated in week 1 (you saved this trace in step e of exercise 1.2, remember?) In the current exercise, we will use this trace in step k. For simplicity, we will call the trace **`ex1.tr`**.
b) First, we have to define the property. On Windows, double click the **`ttlchecker`** icon on your desktop. On unix, execute **`/usr/local/ai/lt/bin/ttlchecker`**. For detailed installation instructions, read the user manual at www.cs.vu.nl/~lourens/usermanual.html. The initial window will look like this (unix version):

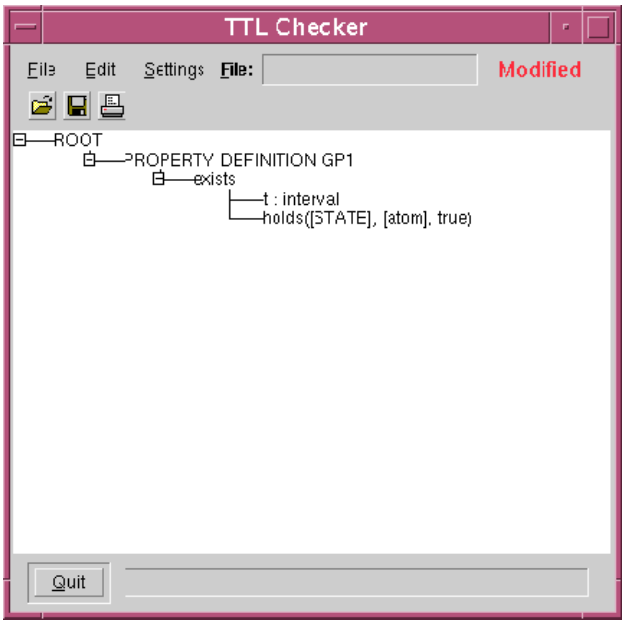


c) Mouse over **ROOT** and click the right mouse button. Select **Property definition** and release the mouse button.
d) Define your own name for the property (edit **[name]**). For instance, **GP1**.

e) Right-click on **[formula]** and select **Change exists**.

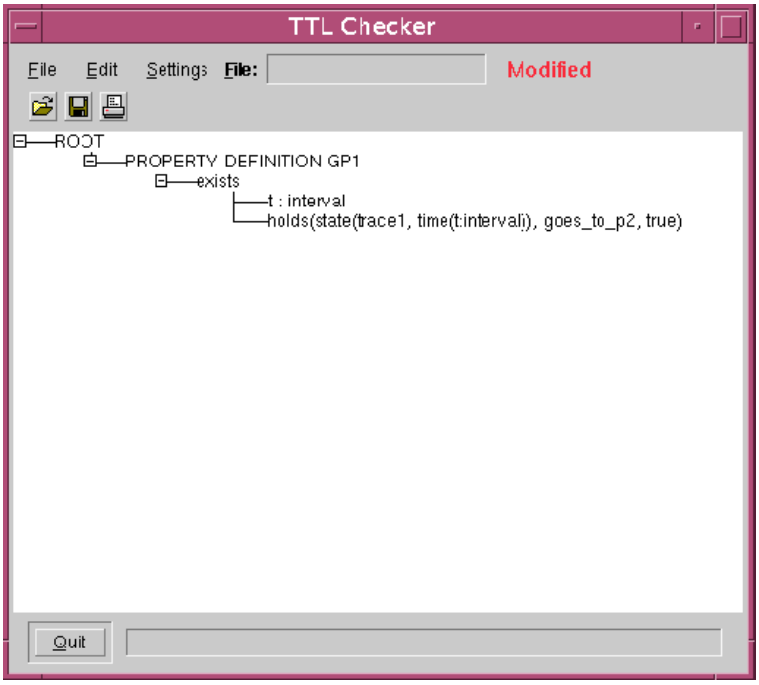
f) Edit **[var] : [sort]** so that the result is **t : interval**. (Note that, when referring to time points, we will use the predefined sort **interval**. This sort simply consists of all distinct time intervals for a formula to be checked (numbered between 0 and the last interval). For the last interval the constant **last_interval** is available. For more information, see the user manual.)

g) Right-click on the next **[formula]** and select **Change holds**. The canvas now looks like this:



h) Edit **holds([STATE], [atom], true)**. In the window that pops up, change **[STATE]** into **state(trace1, time(t:interval))** and change **[atom]** into **goes_to_p2**. As a result, the window should now look as follows:

i) Press button **Ok**. Your property is now completed. If everything went well, the main canvas looks as follows:



j) It is now time to start checking the formula. However, lots of errors in your specification may lead to exiting the checker program. If you do not wish your editing work to get lost, save your ttl specification by clicking on the second icon below the **File** entry (looks like a floppy) and specify a file name (without extension), say **`ex1check`**.

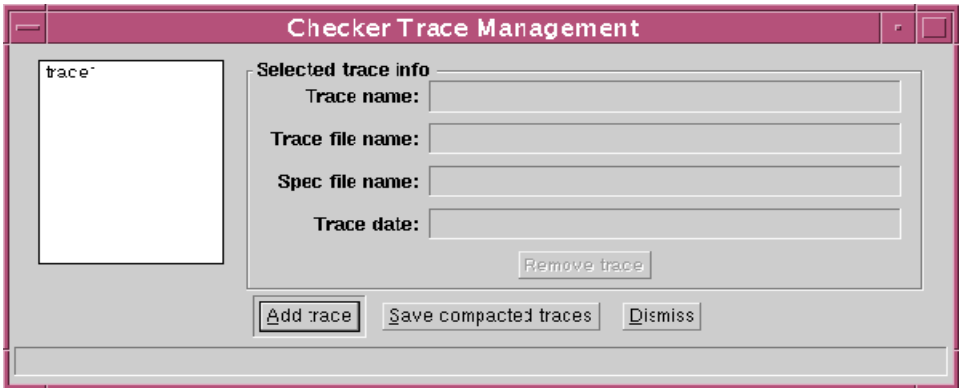
k) Load the trace against which we will check the property:

l) Select **File => Trace Management...**: a "**Checker Trace Management**" window should appear.

m) Press the **Add trace** button: a File chooser window should appear.

n) Select the trace file you generated in week 1 (e.g. **`ex1.tr`**) to add (double click, or open).

The "**Checker Trace Management**" window now looks like this:



Notice the name of the trace in this window, which is **`trace1`**. The editor has automatically assigned this name to **`ex1.tr`** as its identifier. Thus, in each property, if you want to talk

about **`ex1.tr`**, just write **`trace1`**. Likewise, if we would load a second trace into the checker, the identifier **`trace2`** would be assigned to it.

o) Finally, you may check your formula by right clicking on the property (e.g. **GP1**) and selecting **"Check property verbose"** or **"Check property quiet"**.
This will compile the formula the property represents and will start checking. The "verbose" variant will give lots of output on the background window, which may provide you with some information on why a *ttl-formula* is not satisfied.
The result of your check will appear on the bottom of the main window. Did your check succeed?

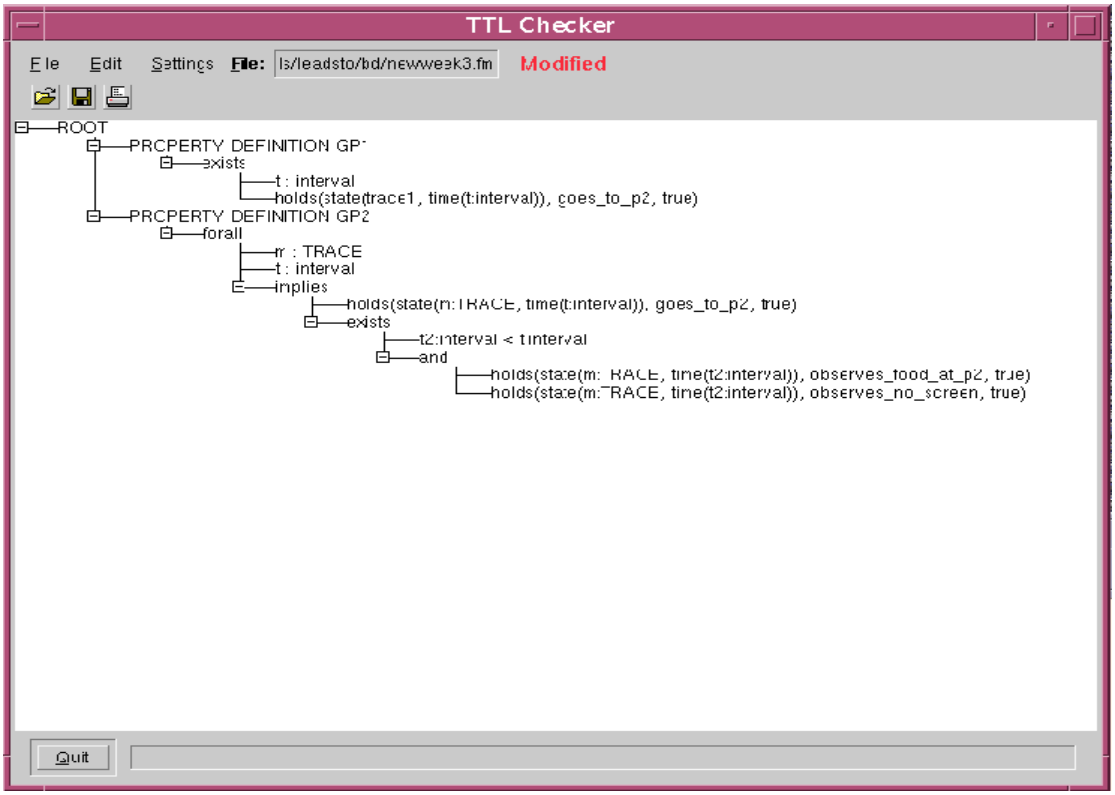
p) As an additional exercise, try if you can check a more complex property as well. For instance, the property GP2 stating the following (in informal notation):
"*in any case that the animal goes to p2, then earlier on it observed food at p2 and did not observe a screen*".
In semi-formal notation, this property can be expressed as follows:
"*for all traces m and all time points t, if at t the animal goes to p2, then there was an earlier time point t2 such that at t2 the animal observed food at p2 and did not observe a screen*".
In formal notation (TTL), the property can be expressed as follows:

```
∀m ∀t state(m,t) |= goes_to_p2 ⇒ [∃t2 < t state(m,t2) |=
observes_food_at_p2 & state(m,t2) |= observes_no_screen].
```

Here are some useful hints you will need in order to define this property:

- Like in the previous property, for each variable you have to specify the corresponding sort. So, for **m** you can use the predefined sort **TRACE**, and for **t** you can use the predefined sort **interval**.
- In order to specify the expression **∀m ∀t**, you can introduce the **forall** only once (this can be done in the same way as the **exists** in the previous property) and then add the two variables.
- In order to specify the expression **t2:interval < t:interval**, click on **[var] : [sort]** and then edit the field **Whole term**.

If everything goes well, then your solution should look like this:



Does this property hold as well? Discuss why. Hand in your discussion as well as the file containing the properties GP1 and GP2.