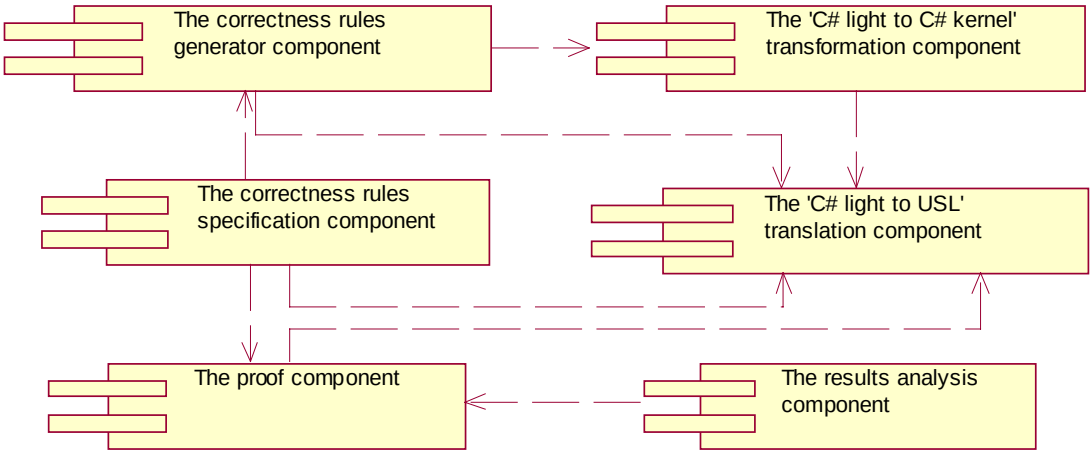
Description of the C#-light programs verification system

The developing C#-light programs verification system consists of the following components:



The verification process is done step by step. Each step is supported by the corresponding system component. In other words the system has so called ‘module’ architecture that allows to develop each of its components almost independently.

While working with the system user adds annotations to a program and starts its verification. The first step of the verification process is translation of the C#-light program source code to the universal USL representation. Further all other components work with this USL representation.

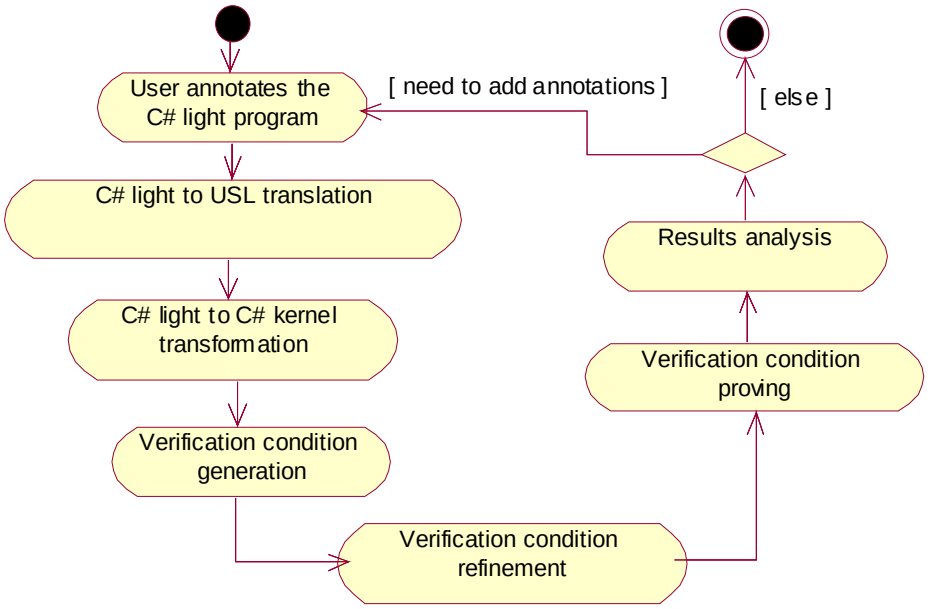
After the ‘C#-light to USL’ translation the ‘C#-light to C# kernel‘ transformer component starts to work. The aim of its work is to transform the C#-light language structures to the C# kernel language structures which are much simpler. During this process the USL representation of the program is transformed via the special interface provided by the component of ‘C#-light to USL’ translation.

When the C#-light program is transformed to the C# kernel the correctness conditions generator component starts to work. The result of its work is the set of lemmas about the verifying program and we need to prove them to verify the program. It is important to note that while the correctness conditions generation process we use the symbolic substitutions for lazy computations of some program characteristics. For example we use them in case of determination of the overloaded function that is called.

Now it is obvious that before the proof component will start to work someone should resolve all substitutions in the correctness conditions. The component that is responsible for this process is called the component of the correctness conditions specification. This component is responsible for resolving of all lazy computations in the generated correctness conditions. This component uses the proof component to get all required auxiliary data about the program. The process of partial program interpretation is used to gather this data.

When the correctness conditions are specified the proof component starts to work. It proves the specified correctness conditions gathered on the previous step. After the proof process finishes the results analysis component determines whether user should have to add additional annotations to the program or the verification process has passed. In the last case we have two possibilities. The first one is that the program conforms to the given specifications and then everything is fine. The second one is that the program does not conform to the given specifications and then corresponding error messages are provided.

The following activity diagram describes the verification system work algorithm:



Further we will describe each of the system's components with more details.

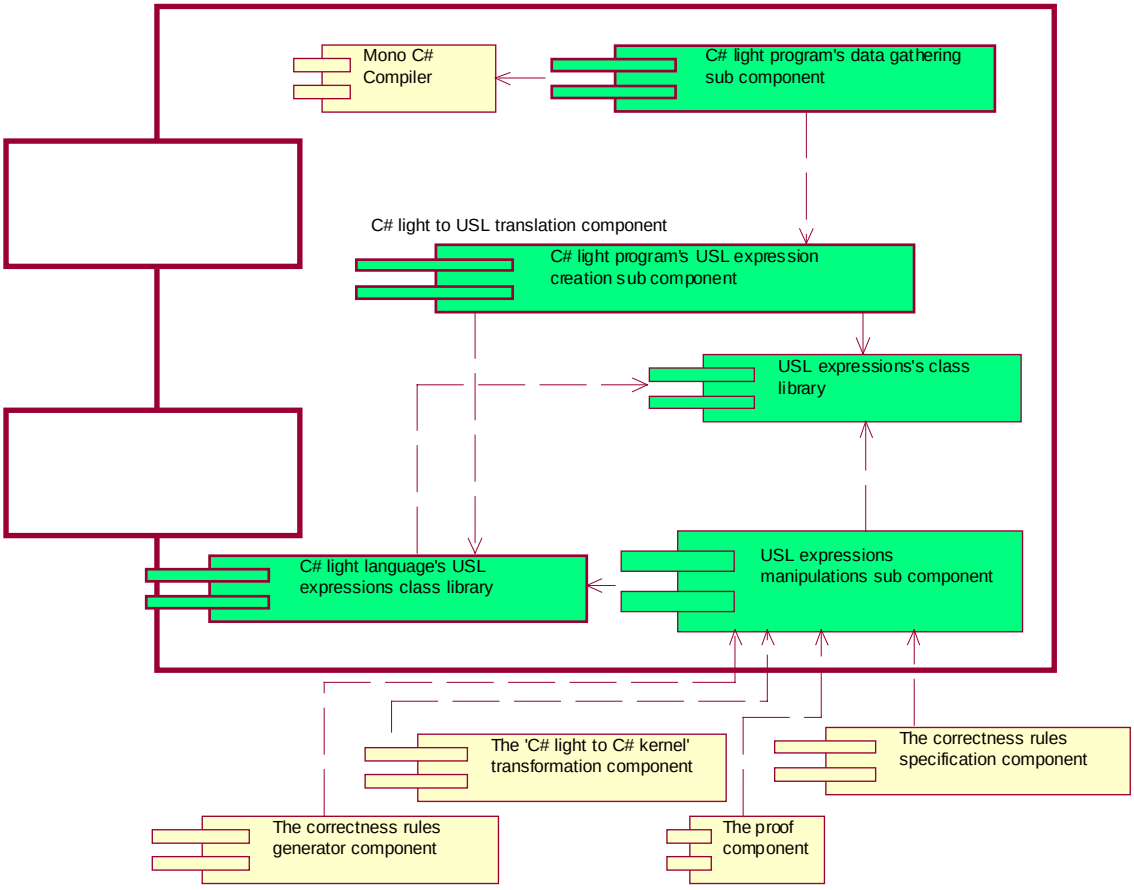
1. The C#-light to USL translation component

Translation of a C#-light program to a USL representation is one of the first stages in the C#-light programs verification process. The main task of this step is translation of a C#-light program to a USL expression according to the rules defined for translation of C#-light language structures to USL.

The C#-light to USL translation component consists of the following parts:

- USL expressions class library
- C#-light language's USL expressions class library
- C#-light program's data gathering sub component
- C#-light program's USL expression creation sub component
- USL expressions manipulations sub component
    - USL expressions by-pass classes
    - USL expressions search classes
    - USL expressions modification classes

The component diagram listed below shows the sub components structure of the C#-light to USL translation component and its relations with the other components:



The C#-light to USL translation component's sub components are marked with the green color. Yellow components are the verification system's components mentioned earlier. The white component is the external component that is used in the system.

The C#-light to USL translation component provides program's data through interfaces used by the C#-light to C# kernel transformation, correctness conditions generator components and others.

It is obvious that an important step of the component's work is the step of gathering of the C#-light program data and creation of the corresponding USL expression. This step presumes either use of the third party C# parser or creation of the special C#-light parser. Currently we use an open source C# compiler called Mono distributed by the Ximian company (http://www.go-mono.org/).

The main goal of the C#-light to USL translation is to separate from the C#-light program's data provider and to store this data in the useful and universal representation.