Translation of C#-light language’s syntactic constructions into USL expressions as the stage of C#-light programs verification

Ivan S. Zapreev, Institute of Informatics Systems, Novosibirsk, Russia

C#-light is a subset of C# language. It allows writing sequential programs. This language contains all C# language constructs except threads, attributes, unsafe code, destructors, lock and resource statements, and checked and unchecked constructs.

Unified Semantic Language (USL) has been created as the result of generalization of different approaches to programming languages’ formal semantics specification.

Main constructs of USL language are names and expressions.

In C#-light programs verification USL is used to describe a C#-light operational semantics, it is used as the internal language of the verification tool, and it is also used to represent rules of C#-light-kernel language axiomatic semantics.

The C#-light language operational semantics definition requires an abstract machine specification that in its turn demands to determine abstract machine’s states and behavior. Each state is defined in terms of a language entity and a language object.

Each language entity is a language syntactical construction. For example it is variable, statement, class definition, etc. A language entity is defined by its type and a set of attributes. For instance, a “class definition” entity of C#-light language has a type `class-declaration` and the following set of attributes: `attribute-sections`, `modifiers`, `name`, `inherited-class`, `implemented-interfaces` and `members`.

Each language object is an instance of a language entity, in other words it may be a concrete class definition, variable and etc. A value of a language object is represented as a USL expression of the following kind: `<[name-type, t], [`a_1`, `e_1`], …, [`a_n`, `e_n`]>` where `t` is a type of a corresponding language entity, $a_1, \dots, a_n$ are attribute names, and $e_1, \dots, e_n$ are corresponding attribute values.

Rules that translate C#-light language constructs (language entities) into USL expressions are defined depending on a C# language specification.

Example (a C#-light language construction translation to the USL expression):

the local variable definition

```
int i = 0;
```

is translated into the next USL expression

```
<[name-type,
  local-variable-declaration],
 [type,
  int],
 [declarators,
  [<[name-type,
     local-variable-declarator],
    [name,
     i],
    [initializer,
     0]>]]>]
```

The following results were gathered while development of the C#-light to USL translator:

1. A common approach to translation of C#-light programs into USL expressions has been developed;
2. A USL classes library has been developed;
3. A USL expression parser has been developed;
4. A prototype of the “C#-light to USL” translation component has been developed on the bases of an open source C# compiler (Mono, Ximian).