A Summary of the Work on the Proof Theory for the Language POOL

Frank de Boer
Centre for Mathematics and Computer Science
P.O. Box 4079
1009 AB Amsterdam
The Netherlands
March 28, 1989

Under the guidance of Jaco de Bakker I have been working with Pierre America in the ESPRIT project 415 on the proof theory of the language POOL (a Parallel Object-Oriented Language [Am]).

A program of the language POOL describes the behaviour of a system consisting of objects. An object operates upon a local state which assigns objects to variables. A local state is directly accessible only to the object to which it belongs.

An object comes into existence by some "creative act" of some other object. The identity of the newly created object is stored into the local state of the "creator". When an object is created a local process is started up which then will run in parallel with the local processes of the already existing objects.

Objects interact by some form of remote procedure call (also called rendezvous). A call consists of a specification by the sender of the receiver, a procedure (also called a method), and some parameters. When such a call is answered by the receiver the local process of the sender is suspended until the execution of the specified method (by the receiver) has terminated, and the result has sent back.

Objects are grouped into classes. All objects in one class (the instances of that class) have the same kind of variables, the same methods, and the same local process. In this way a class describes the behaviour of its instances.

One of the main proof theoretical problems of such an object-oriented language is how to reason about dynamically evolving pointer structures. To master the complexity of this problem we investigated several approximations of the language POOL.

First we studied the proof theory of a version of POOL called P ([B1]). The objects of the systems described by this language may interact only by a
synchronous communication mechanism which consists of sending and receiving
objects. When an object wants to send an object it explicitly states to which
object. On the other hand receiving an object does not require in general the
identification of the communication partner. In those cases then the communi-
cation partner is selected non-deterministically. This communication mecha-
nism is similar to the one embodied in the language CSP. The main difference is that
the communication partner in CSP is identified statically, which is not the case
in the language P.

To describe a system of objects we view variables as dynamic one-dimensional
arrays. The objects themselves are identified by integers in such a way that the
value of a variable $x$ of an object identified by the number $n$ is given by the $n^\text{th}$
element of the array denoted by $x$. Using this scheme we showed how to apply
the proof theory developed for CSP ([AFR]) to this language P.

However, this coding mechanism of objects makes the abstraction level of the
reasoning about program correctness less high than that of the programming
language. Pierre America developed a proof theory for the language SPOOL (a
Sequential version of POOL) in which one reasons about a system of objects at
a higher abstraction level ([A1]). In more detail, this means the following:

- The only operations on "pointers" (references to objects) are
  - testing for equality
  - dereferencing (looking at the value of a variable of the referenced
    object)
- In a given state of the system, it is only possible to mention the objects
  that exist in that state. Objects that do not (yet) exist never play a role.

Strictly speaking, direct dereferencing is not even allowed in the programming
language, because each object only has access to its own variables. But
to dispense with this feature would ask for even more advanced techniques to
reason about the correctness of a program.

The completeness proof of this proof system for SPOOL requires quite an
elaborate use of the standard techniques ([Ba]), one might almost say that these
techniques are, in this application, "stretched to their utmost limits".

This abstract way of reasoning about dynamically evolving pointer struc-
tures we then applied to the language P ([A2]). Described very briefly the
resulting proof method consists of the following elements:

- A local stage. Here we deal with all statements that do not involve com-
munication or object creation. These statements are proved correct with
respect to pre- and postconditions in the usual manner of sequential pro-
grams [Ba, Ho]. At this stage, we just use assumptions to describe the
behaviour of the communication and creation statements. These will be
verified in the next stage. In this local stage, a local assertion language is
used, which only talks about the current object in isolation.
• An intermediate stage. In this stage the above assumptions about communication and creation statements are verified. Here a global assertion language is used, which reasons about all the objects in the system. For each creation statement and for each pair of possibly communicating send and receive statements is verified that the specification used in the local proof system is consistent with the global behaviour.

• A global stage. Here some properties of the system as a whole can be derived from a kind of standard specification that arises from the intermediate stage. Again the global assertion language is used.

Finally we showed how to generalise the proof theory developed for the Ada rendez-vous ([G]) to the rendez-vous mechanism of POOL ([B2]). The main difference between these two mechanisms being that in the language Ada we have a statically fixed recursion depth of a rendez-vous, whereas in POOL we do not have such a static bound to the recursion depth.

Acknowledgement

I want to stress in particular Jaco’s persistent insistence on a high quality of the presentation without which most of this work would be a mere solipsistic activity. Especially this field of proof theory, which gives rise to rather complicated formalisms, requires special care concerning the presentation.

References


