The EML Kit
Version 1.1
Release Notes[*]

Mikołaj Konarski[†] Sławomir Leszczyński[†]
mikon@mimuw.edu.pl leszczyk@mimuw.edu.pl

July 9, 1998

1 The announcement of the 1.1 release

We are announcing release 1.1 of the EML Kit[1].

The EML Kit aims to be (the basis and test-bed for) a comfortable framework for the formal development of programs, using the Extended ML[2] language, formalism and methodology.

Version 1.1 is a complete implementation of the Extended ML seen as a 'programming language'. It allows for parsing, type-checking and evaluation of arbitrary EML code, be it pure Standard ML or full-blown EML with its axioms and other specification constructs. All of this is done in a strict adherence to the formally defined EML Syntax, Static Semantics and Dynamic Semantics, as of the EML Definition, version 1.15 + 0.2i. See Section 2 for a list of changes to the EML Kit from version 1.0 to 1.1.

Now, there is a great need for an "implementation" of the EML Verification Semantics. We hope the EML Kit will help in as well as benefit from the current work on the EML proof theory, proof obligation generators, prototype provers, etc. Yet, as for now only the syntax and type correctness of axioms is checked by the EML Kit - there is no extra aid in proving them correct, sufficient or whatever.

[*]This work, and the development of The EML Kit, has been partially supported by KBN grants 2P30100704 and 8T11C01811

[†]Institute of Informatics, Warsaw University, Warsaw, Poland.

[1]http://zls.mimuw.edu.pl/~mikon/ftp/EMLKit/README.html

[2]http://www.dcs.ed.ac.uk/%7Edts/eml/

The EML Kit 1.1 Sources Release is intended for everybody, while the i386 Binaries Release and SPARC Binaries Release are for i386 (Linux) and Sun SPARC (Solaris) users respectively. They are available through anonymous ftp from:

`ftp://zls.mimuw.edu.pl/pub/mikon/EMLKit`

or by WWW from our home-page. There is also some documentation available from the `/pub/mikon/EMLKit/doc` ftp directory or by WWW.

Acknowledgments

The development of the EML Kit has been partially supported by KBN (Polish State Research Committee) grants 2P30100704 — "Formal Methods of Software Development"[3] and 8T11C01811 — "Logical Aspects of Software Specification and Development Methods"[4].

The EML Kit is based on the ML Kit[5], a flexible, clean and modularized interpreter of the Standard ML, written in Standard ML.

The authors of the EML Kit: Marcin Jurdzinski, Mikolaj Konarski, Slawomir Leszczynski and Aleksy Schubert welcome any bug-reports and comments (signs of appreciation will also be accepted :-). Write to:

`eml@zls.mimuw.edu.pl`

2 A list of changes to the EML Kit from version 1.0 to 1.1

The following changes were made.

- New keyword `iff` added. A phrase

  ```
  (exp1 iff exp2)
  ```

  works as

  ```
  (if exp1 then exp2 else not exp2)
  ```

- In two places bound module traces were collected instead of simple module traces. Fixed.

[3] `http://wwwat.mimuw.edu.pl/~tarlecki/kbn93grant/index.html`
[4] `http://wwwat.mimuw.edu.pl/~tarlecki/kbn96grant/index.html`
[5] `http://www.diku.dk/research-groups/topps/activities/mlkit.html`

- The error message raised by `CoreTrace.onScheme` function suggested, that it was generated by the `CoreTrace.onScheme_opt` function. Fixed.
- The structure of the EML Kit ftp and http site has been changed a bit. In particular, current address of the EML Kit home page is now

  `http://zls.mimuw.edu.pl/~mikon/ftp/EMLKit/README.html`

- Names of the basis are collected in a different way now. Appendix A contains a detailed record of the history of this change.

A The correspondence over the "Basis Names" issue

There are several e-mails describing this issue, in reverse chronological order:

```
Date: Wed, 25 Jun 1997 18:51:41 +0200 (MET DST)
From: Mikolaj Konarski <mikon@mimuw.edu.pl>
To: bkk@dcs.ed.ac.uk
CC: eml@herbata.mimuw.edu.pl, dts@dcs.ed.ac.uk, smk@dcs.ed.ac.uk
In-reply-to: <Pine.SOL.3.94.970617173730.11320H-100000@dubh.dcs.ed.ac.uk>
        (message from Ben Kleinman on Tue, 17 Jun 1997 17:41:24 +0100 (BST))
Subject: Re: a bug?
Content-Type: text
Content-Length: 877

All right :-)

[snip]

The most important change looks like this (file is ModuleEnvironments.sml):
...
    fun C_of_B (B as BASIS {N, F, G, E}) =
      let
        val T_of_B = C.T_of_N N
      in
        C.mkC(T_of_B, E)
      end

(* mikon#1.pl1.1 Variant 2 *)
    fun namesB(BASIS {N, F, G, E}) =
      C.NameSetUnion (N, C.NameSetUnion
```

```
          (namesF F, C.NameSetUnion (namesG G, namesE E)))

    fun C_of_B_enriched (B as BASIS {N, F, G, E}) =
      let
        val T_of_B = C.T_of_N (namesB B)
      in
        C.mkC(T_of_B, E)
      end
(* end mikon#1.pl1.1 Variant 2 *)
...
```

Everything seems to work.

Thanks to everybody,
Mikolaj

--
mikon@mimuw.edu.pl
http://zls.mimuw.edu.pl/~mikon/home.html
The one good thing about repeating your mistakes is that you know when
to cringe.

Date: Tue, 24 Jun 1997 17:26:59 +0100
To: Mikolaj Konarski <mikon@mimuw.edu.pl>
Cc: smk@dcs.ed.ac.uk, dts@dcs.ed.ac.uk, eml@herbata.mimuw.edu.pl
Subject: Re: a bug?
In-Reply-To: <199706241500.RAA12765@herbata.mimuw.edu.pl>
References: <6193.199706191522@kishorn.dcs.ed.ac.uk>
X-Mailer: VM 6.22 under 19.15p6 XEmacs Lucid
From: Stefan Kahrs <smk@dcs.ed.ac.uk>
Mime-Version: 1.0 (generated by tm-edit 7.68)
Content-Type: text/plain; charset=US-ASCII
Content-Length: 2930

>> The obvious fix seems to be to replace the "+" in rules 77 and 81 by
>> \oplus.
>>
>> I don't see any problem in having additional (non-rigid)
>> type names lurking around in the type-name component of the basis;
>> certainly not in the Kit, which forgets them after the entire
>> signature has been dealt with, but even if they stayed there for ever it
>> would not really matter.

mikon> I might have been wrong in saying, that in the SML Definition

mikon> N of B is meant to contain the names of rigid structures,
mikon> but surely the ML Kit relies heavily on this property.

I don't think you were wrong, although strictly speaking it does not
really matter: SML simply only puts rigid type names (including the
formal parameters of a functor) into a base.

```
mikon> signature HOMOMORPHISM = sig
mikon>     structure From : sig end
mikon>     structure H: sig end
mikon>     sharing H = From
mikon> end;
```

mikon> the ML Kit takes the set of rigid structure names,
mikon> needed for the unification of H and From,
mikon> from the basis collected up to the sharing.
mikon> In the result after the changes you propose
mikon> the above example does not elaborate.

I see, ML Kit refuses to unify because it checks that
at least one of the newly shared names is not in the base.

mikon> The solution might be to pass the set of rigid structure names
mikon> as a separate argument, and not implicitly in the basis.
mikon> This would be *very* troublesome to code and verify =:0 .

Alternatively, one could add another attribute (beside equality)
to names: unifiability, thus you could only share names
one of which is unifiable, no rigid name is unifiable, etc.
In other words: one doesn't change the number of parameters of other
operations, one changes the datatype of names.

Yet another possiblity would be to drop the rigidity check in the
unification and instead check posthumously (after the whole signature
has been dealt with) that no rigid equation has been imposed.
This, of course, would require to store these equations in semantic
signatures somehow.

mikon> I would rather implement the (C of B enriched with names of B)
mikon> in rule 61.2 thing, and pray it's equivalent.

Good enough for me.
But that's just the implementation perspective, i.e. what do you do in

the ML Kit, and that's entirely up to you. I'm sure there isn't
any difference overall between both ways of doing it (considering that
I'm an atheist -- praying for their equivalence probably wouldn't help).
Implementationwise I would have thought that "C of B + names B"
is a bit awkward to compute, or is this info neatly available via some
counter?

mikon> Actually, according to the principle of preserving
mikon> as much as possible and reasonable of the original SML Definition
mikon> the solution which changes only rule 61.2 might be quite acceptable.
mikon> What do you think?

Again, fine with me; it should work either way.

Stefan

Date: Tue, 24 Jun 1997 17:00:12 +0200 (MET DST)
From: Mikolaj Konarski <mikon@mimuw.edu.pl>
To: smk@dcs.ed.ac.uk
CC: dts@dcs.ed.ac.uk, bkk@dcs.ed.ac.uk, eml@herbata.mimuw.edu.pl
In-reply-to: <6193.199706191522@kishorn.dcs.ed.ac.uk> (message from Stefan
Kahrs on Thu, 19 Jun 1997 16:22:27 +0100)
Subject: Re: a bug?
Content-Type: text
Content-Length: 6007

[snip]
> You're wrong.
> It's not a bug in the EML Kit.

I just got used to be blamed by my students for what
they do not like or do not understand in EML.

> Because you've implemented what's in the EML definition.
> [Bad move! :-]

Now I do see that :-].

> Alas, we didn't define what we should have defined, so there is indeed
> a bug in the EML definition. Well-spotted, Mikolaj!
> The obvious fix seems to be to replace the "+" in rules 77 and 81 by
> \oplus.
>
> I don't see any problem in having additional (non-rigid)

> type names lurking around in the type-name component of the basis;
> certainly not in the Kit, which forgets them after the entire
> signature has been dealt with, but even if they stayed there for ever it
> would not really matter.

I might have been wrong in saying, that in the SML Definition
N of B is meant to contain the names of rigid structures,
but surely the ML Kit relies heavily on this property.

Here:

```
signature HOMOMORPHISM = sig
    structure From : sig end
    structure H: sig end
    sharing H = From
end;
```

the ML Kit takes the set of rigid structure names,
needed for the unification of H and From,
from the basis collected up to the sharing.
In the result after the changes you propose
the above example does not elaborate.

The solution might be to pass the set of rigid structure names
as a separate argument, and not implicitly in the basis.
This would be *very* troublesome to code and verify =:0 .
I would rather implement the (C of B enriched with names of B)
in rule 61.2 thing, and pray it's equivalent.

Actually, according to the principle of preserving
as much as possible and reasonable of the original SML Definition
the solution which changes only rule 61.2 might be quite acceptable.
What do you think?

> The only place where one has to take some care
> is when principality is imposed: we abstract all type names that are
> not in the T of B component. This is still correct, but after the
> change it is now important that it is the T component of the basis
> _before_ the elaboration of the signature expression took place.
> I would expect that this is done in the MLKit anyway.

You are right, this is not the problem.

```
  > BTW the problem also causes soundness hickups, because datatypes local
  > to axioms could be freely identified with specified (non-rigid) types
  > from the surrounding signature.

Sure.

  > Regards and thanks for locating the bug
  >
  >     Stefan

Regards and thanks for you help,
Mikolaj
```

At lat, the original message follows:

```
Date: Thu, 19 Jun 1997 16:25:00 +0200 (MET DST)
From: Mikolaj Konarski <mikon@mimuw.edu.pl>
To: bkk@dcs.ed.ac.uk
CC: eml@herbata.mimuw.edu.pl, smk@dcs.ed.ac.uk
In-reply-to: <Pine.SOL.3.94.970617173730.11320H-100000@dubh.dcs.ed.ac.uk>
        (message from Ben Kleinman on Tue, 17 Jun 1997 17:41:24 +0100 (BST))
Subject: Re: a bug?
Content-Type: text
Content-Length: 2993

  > Date: Tue, 17 Jun 1997 17:41:24 +0100 (BST)
  > From: Ben Kleinman <bkk@dcs.ed.ac.uk>
  > MIME-Version: 1.0
  > Content-Type: TEXT/PLAIN; charset=US-ASCII
  > Content-Length: 1509
  >
  > This, we think, is a bug in the EML kit.  Of course, we're hoping it's
  > we're wrong :-).
  >
  > The following signature (Which no longer does what the comments claim it
  > does) generates: local tyname(s) escape from a let expression.
  >
  > chf,cht,...,x,y are explicitly universally quantified only in an attempt
  > to eliminate the error -- I don't think they should be.
  >
  > (* Exports a sig for a character type *)
  > signature CHAR =
  > sig
  >     eqtype char;
```

```
> end;
>
> (* should this perhaps be a functor parameterized on the CHAR struct? *)
> signature REGEXP =
> sig
>     structure Char : CHAR;
>
>     datatype regexp = embed of Char.char
>                 | compose of regexp*regexp
>                 | union of regexp*regexp
>                 | iterate of regexp;
>
>     (* intended to return the longest prefix of a given list matching
>        the regexp, followed by the suffix *)
>     val longMatch: ((Char.char list) * regexp) ->
>                        (Char.char list * Char.char list)
>
>     (* returns a list of all the initial matches
>        and remainders for a regexp and a string *)
>     val match: ((Char.char list) * regexp) ->
>                    (Char.char list * Char.char list)list
>
>     axiom forall chs => forall reg => forall chf => forall cht => forall
> rest => forall x => forall y =>
>     (match(chs,iterate(reg)) =
>         let fun helper [] = []
>               | helper (([],cht)::rest) = helper rest
> (* trouble *)     | helper ((chf,cht)::rest) =
>                     (map (fn(x,y)=>(x,y)) (match (cht,reg)))
>         in helper(match (chs, reg))
>         end)
>
> end;
>
>
> Ben Kleinman
> bkleinman@writeme.com
> http://www.dcs.ed.ac.uk/home/bkk
>

Yes, this is a bug.
Thank you. >:-}
```

```
Actually, this seems to be a bug in the EML definition.
Here is a simpler version of your program :-)
(nota bene chf,cht,...,x,y have to be explicitly
universally quantified in your program):

signature S =
sig
    datatype chf = x
    axiom x == let in x end
end;

which still causes:

      axiom x == let in x end
                 ^^^^^^^^^^^^
Local tyname(s) escape from a let expression

The problem is, that in rules 77 and 81
environment E is added to basis B with a simple plus,
not with the plus in circle, because the N of B
is meant to contain the names of rigid structures, or smth.
On the other hand in rule 61.2 (C of B) is used
as the context for axiom elaboration,
instead of (C of B enriched with names of B).
And so the side-condition from rule 6
is not met in programs such as the ones above.

As soon as Stefan decides what to do,
we'll patch the EML Kit and inform you.

Thanks again,
Mikolaj
--
mikon@mimuw.edu.pl
http://zls.mimuw.edu.pl/~mikon/home.html
The one good thing about repeating your mistakes is that you know when
to cringe.
```