Mobile Resource Guarantees — MRG

University of Edinburgh
= David Aspinall
Stephen Gilmore
Don Sannella (coordinator)
Ian Stark
and 2.5 researchers

+

LMU Munich
= Martin Hofmann
and 2 researchers

Summary

Topic: Proof-carrying mobile code for resource-related properties

Scenario:

1. A requests code from B, specifying a resource policy p
2. B sends code to A with a proof π that c abides by p
3. A checks that π is valid for c and p
4. If yes, A runs c

This is an active research area for security. It is practical because:

Proofs are unforgeable

Proof checking is easy even if proof generation is hard

The properties in question are modest, not full-blown correctness

Approach

Based on Hofmann's work on linear type systems which guarantee time and space bounds of programs

insert : $\diamond \times A \times A\ list \rightarrow A\ list$

```
fun insert(d,a,nil) = cons(d,a,nil)
  | insert(d,a,cons(d',b,l)) =
        if a <= b then cons(d,a,cons(d',b,l))
        else cons(d,b,insert(d',a,l))
```

sort : $A\ list \rightarrow A\ list$

```
fun sort nil = nil
  | sort(cons(d,a,l)) = insert(d,a,sort l)
```

We will generate proofs of resource bounds from typing derivations

$$p : \tau \xrightarrow[\text{compilation}]{} \text{bytecode} + \text{proof}$$

Workplan sketch

1. Development of framework in which certificates of resource consumption make formal sense: virtual machine (JVM subset, maybe .NET) + cost model + program logic for resource properties

2. Formalization of proofs, implementation of proof checking

3. Development of methods for generating such certificates for high-level code: implemented high-level language (initially functional, then OO) compiling to VM + resource type systems for HLL + translation of typing derivations to proofs

4. Experiments with alternatives to full proofs:

   "prooflets" = code to re-create a proof in the client (cf LCF tactics)
   probabilistic certification (cf probabilistically checkable proofs)
   negotiation via a series of challenges and responses