Automatic datatype versioning

An adventure in Ocaml, generic programming and preprocessors.

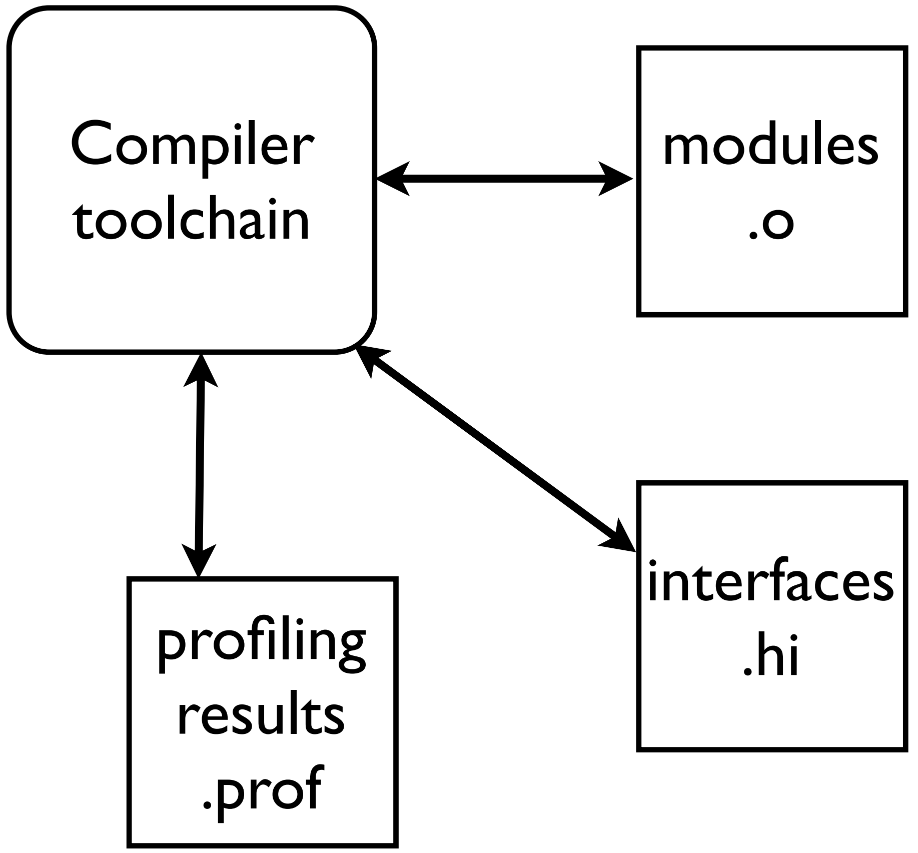
FP-dag, Nijmegen, January 8, 2010

Alexey Rodriguez Yakushev
Vector Fabrics

Vector Fabrics

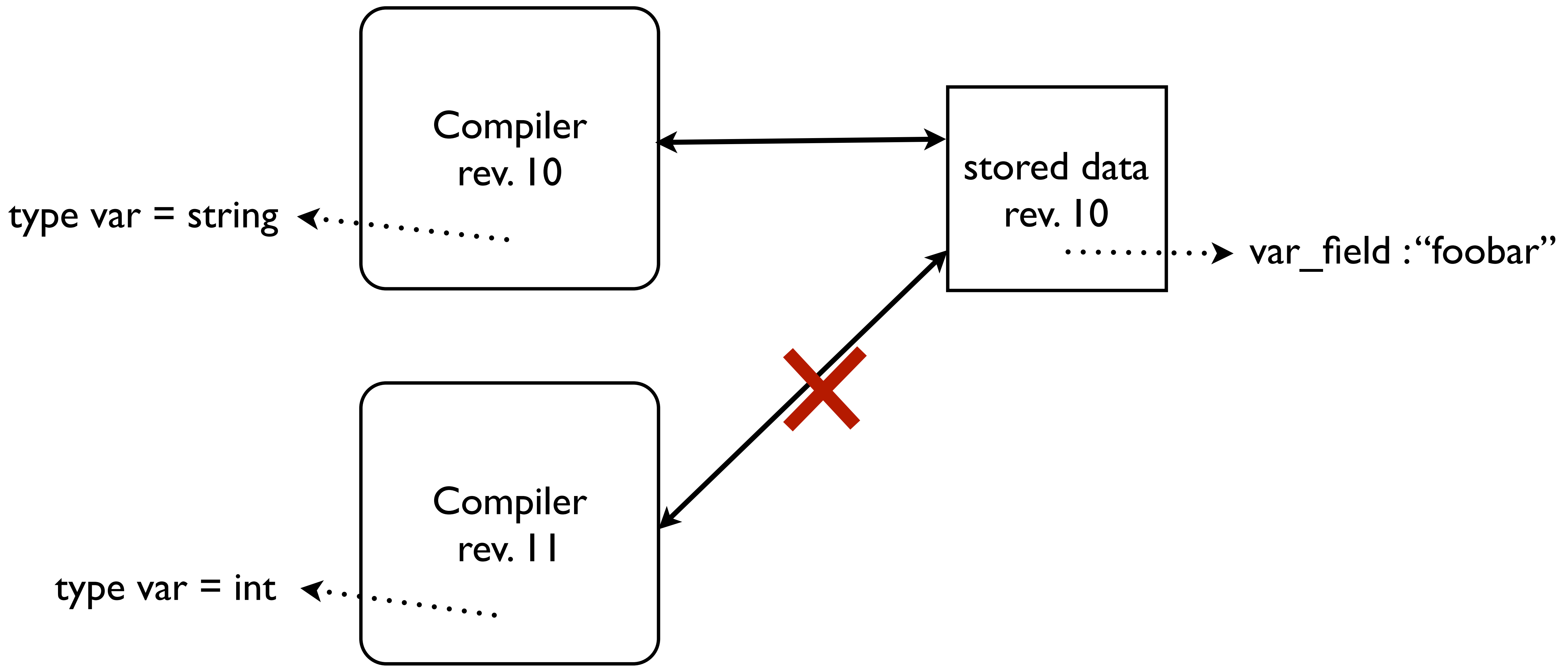
- Produces an embedded systems compiler: C to hardware + software.
- The compiler itself is written in Ocaml.

Data persistence



- Two styles:
- File format. ⋯→ Stability
  - Design format and code reader/writer.
- Marshaling. ⋯→ Flexibility
  - (Semi)automatic from data definitions.

Version mismatch (marshaling)



How serious is the problem?

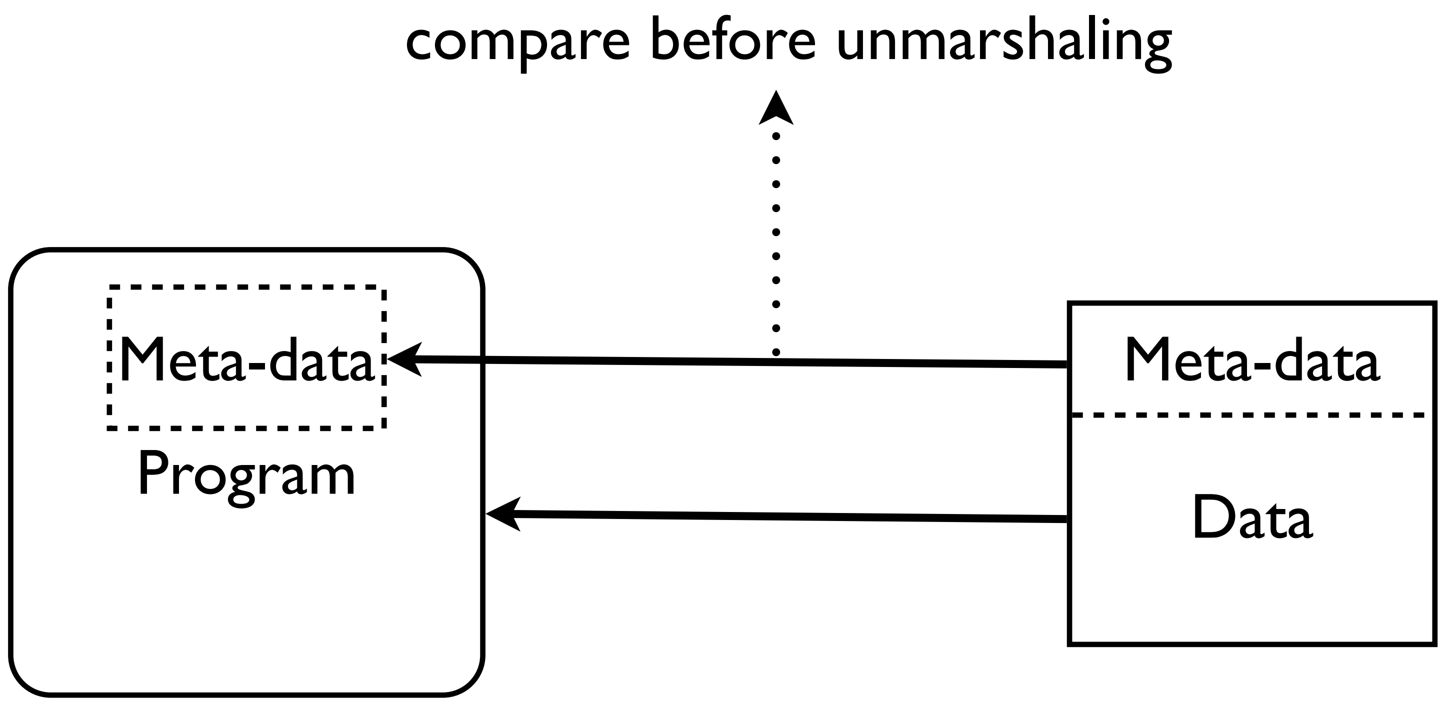
- Vector fabrics: 8 active developers, 50 patches a day.
- Conservative: regenerate files on every code update. Very time consuming.
- Practical: do not regenerate and hope for the best.

How to deal with marshaling and evolving datatypes in an automatic and non-invasive way?

VectorFabrics

Designing a solution

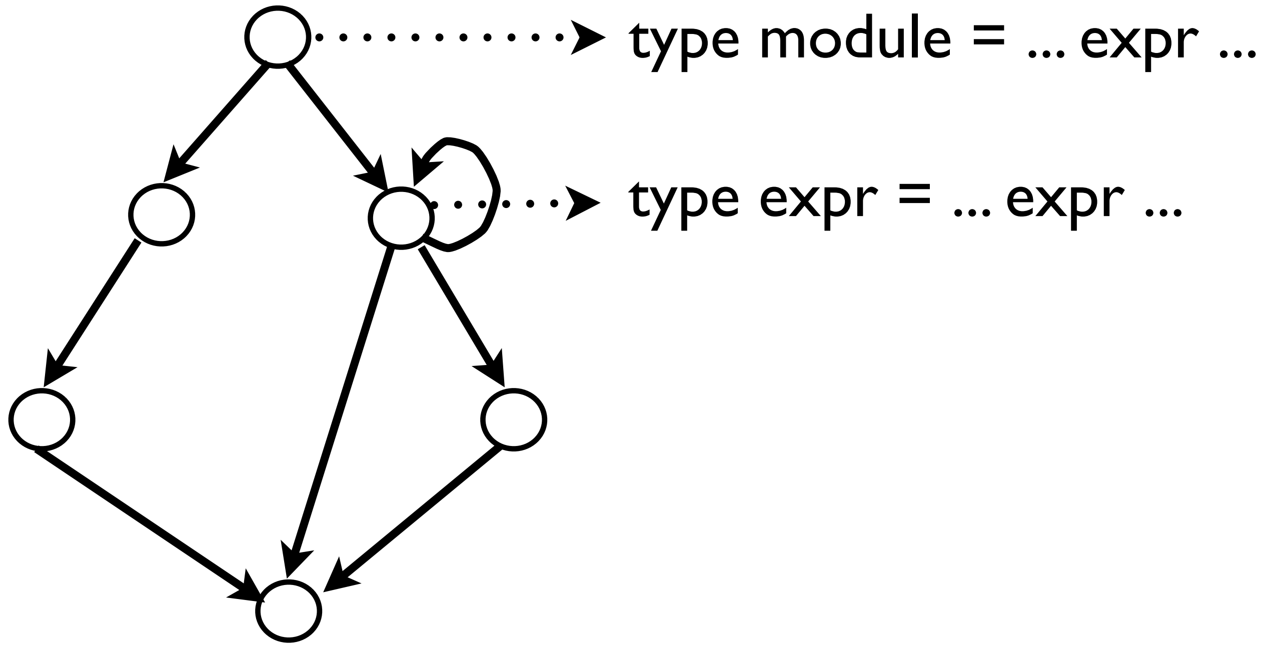
Meta-data



What meta-data to use?

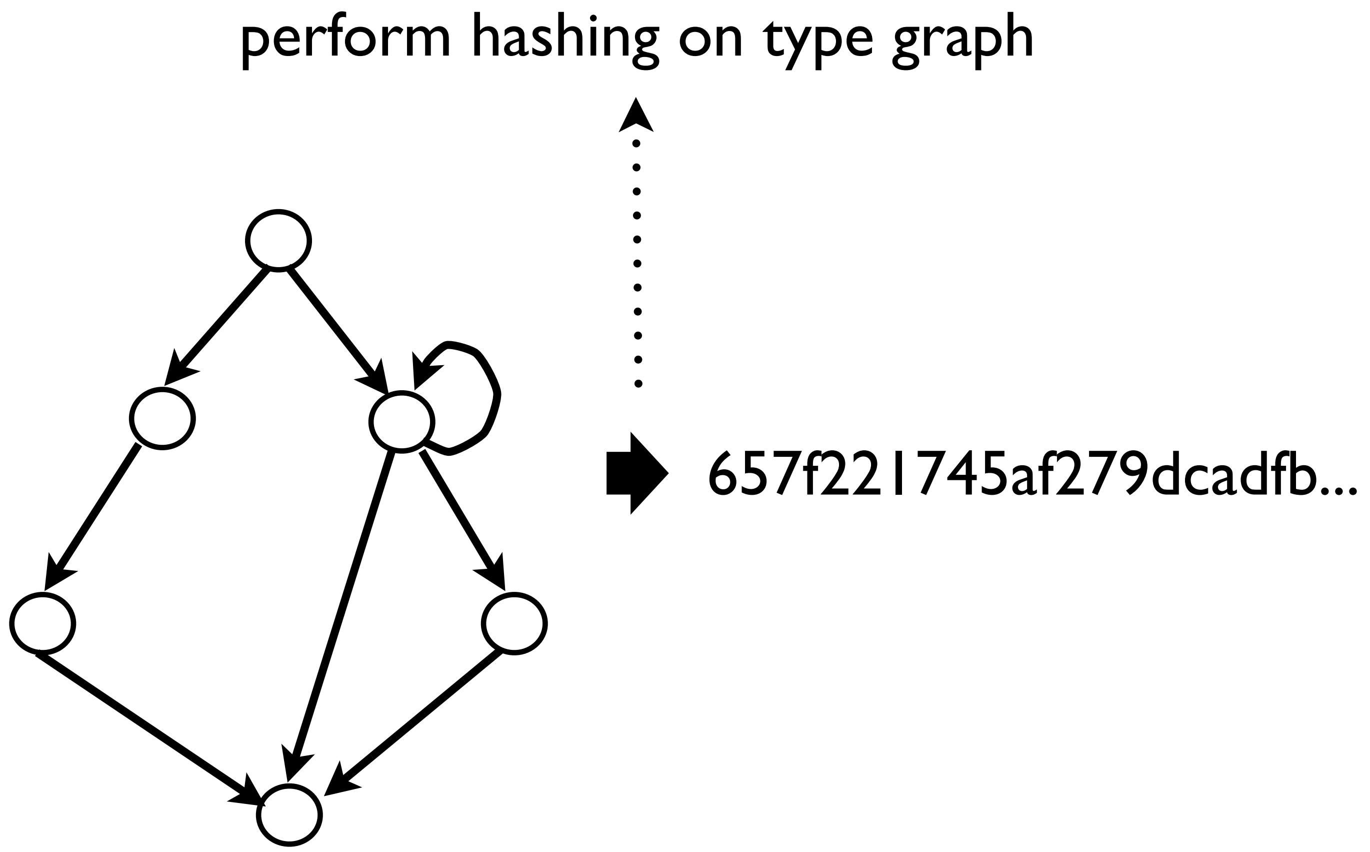
Type graph as meta-data

- Store type graph as meta-data.
- Version checking is structural comparison.
- May be used for backward compatibility.



Hashing as meta-data

- Only version checking.
- Efficient.
- Collisions not likely. Also, no attackers.
- Pay attention to transitivity.



Strategy

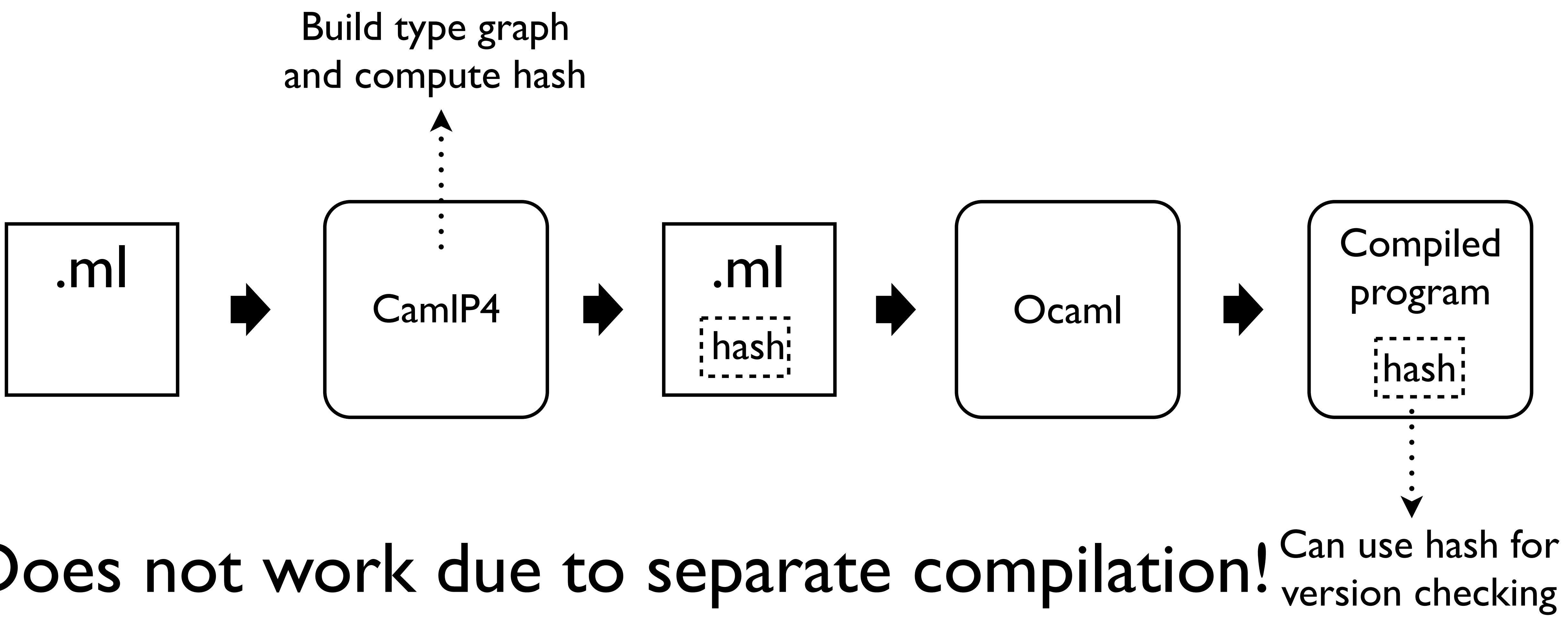
- Build type graph.
- Compute the MD5 hash of the graph.
- Use hash for version checking.

Implementation

Tools

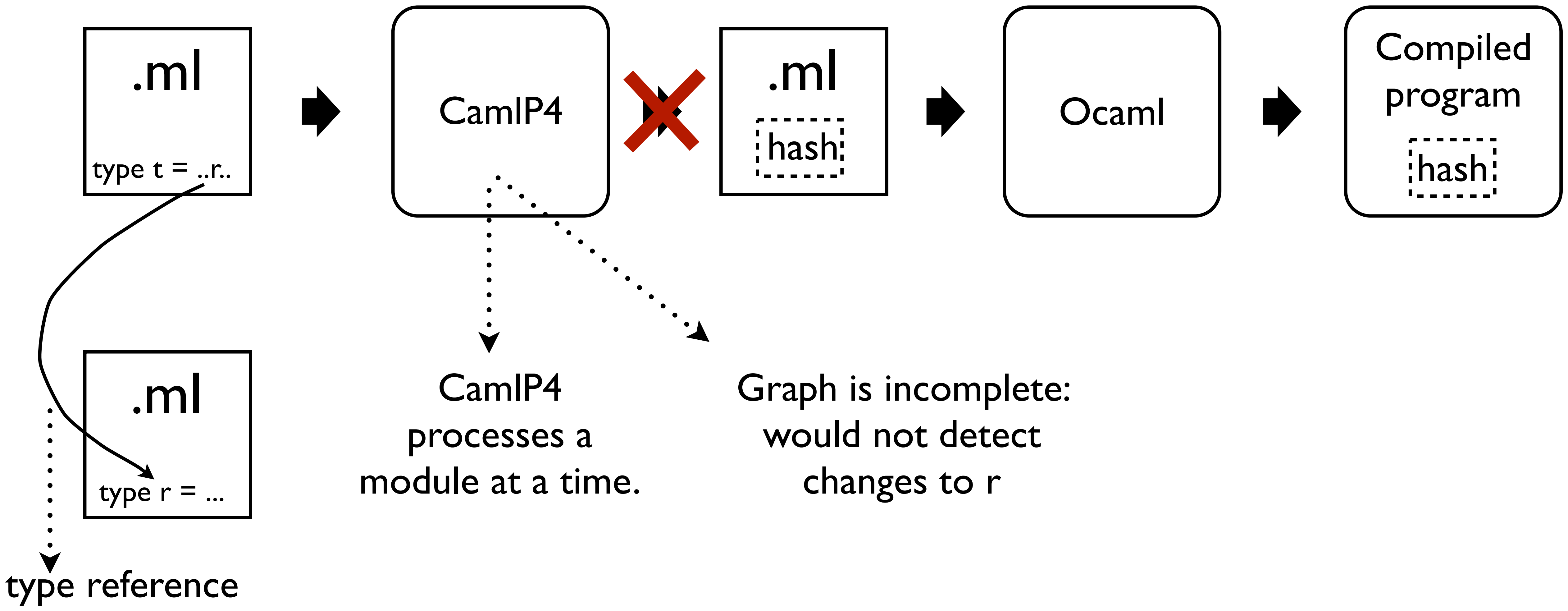
- Ocaml.
- CamlP4 (Ocaml preprocessor).
- Jane Street Capital's type-conv (CamlP4 plugin).

Hashing at compile time

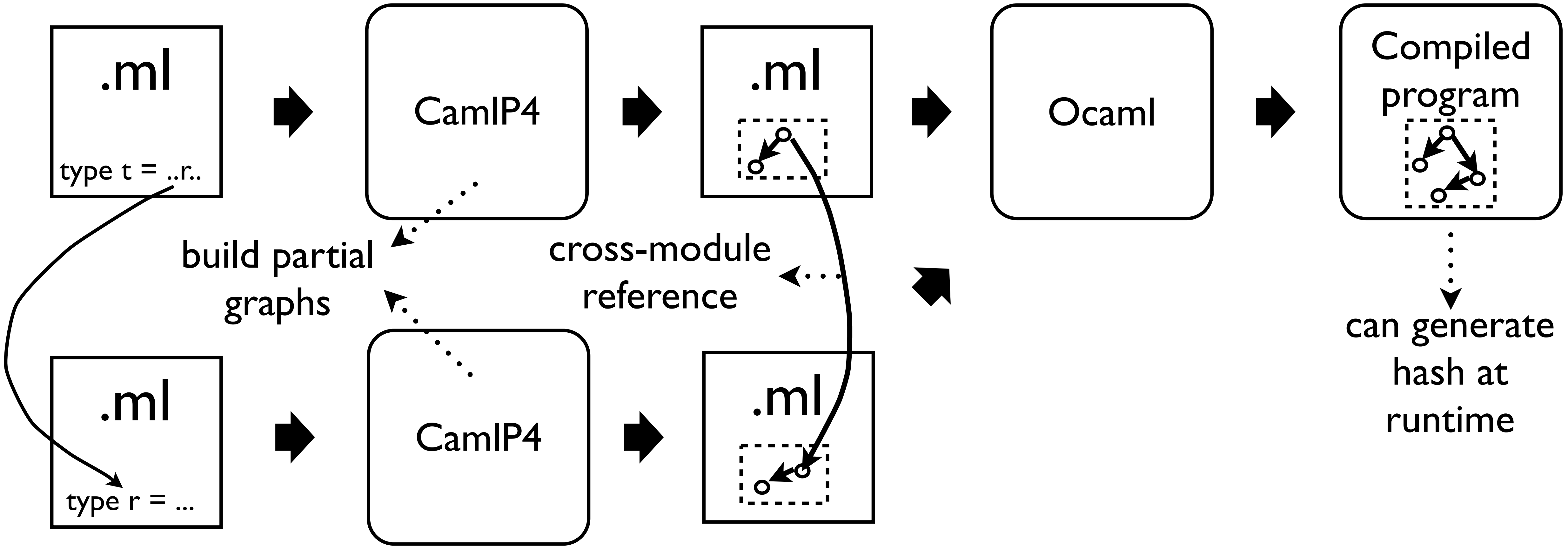

Build type graph and compute hash
.ml
CamlP4
.ml
hash
Ocaml
Compiled program
hash
Can use hash for version checking

Does not work due to separate compilation!

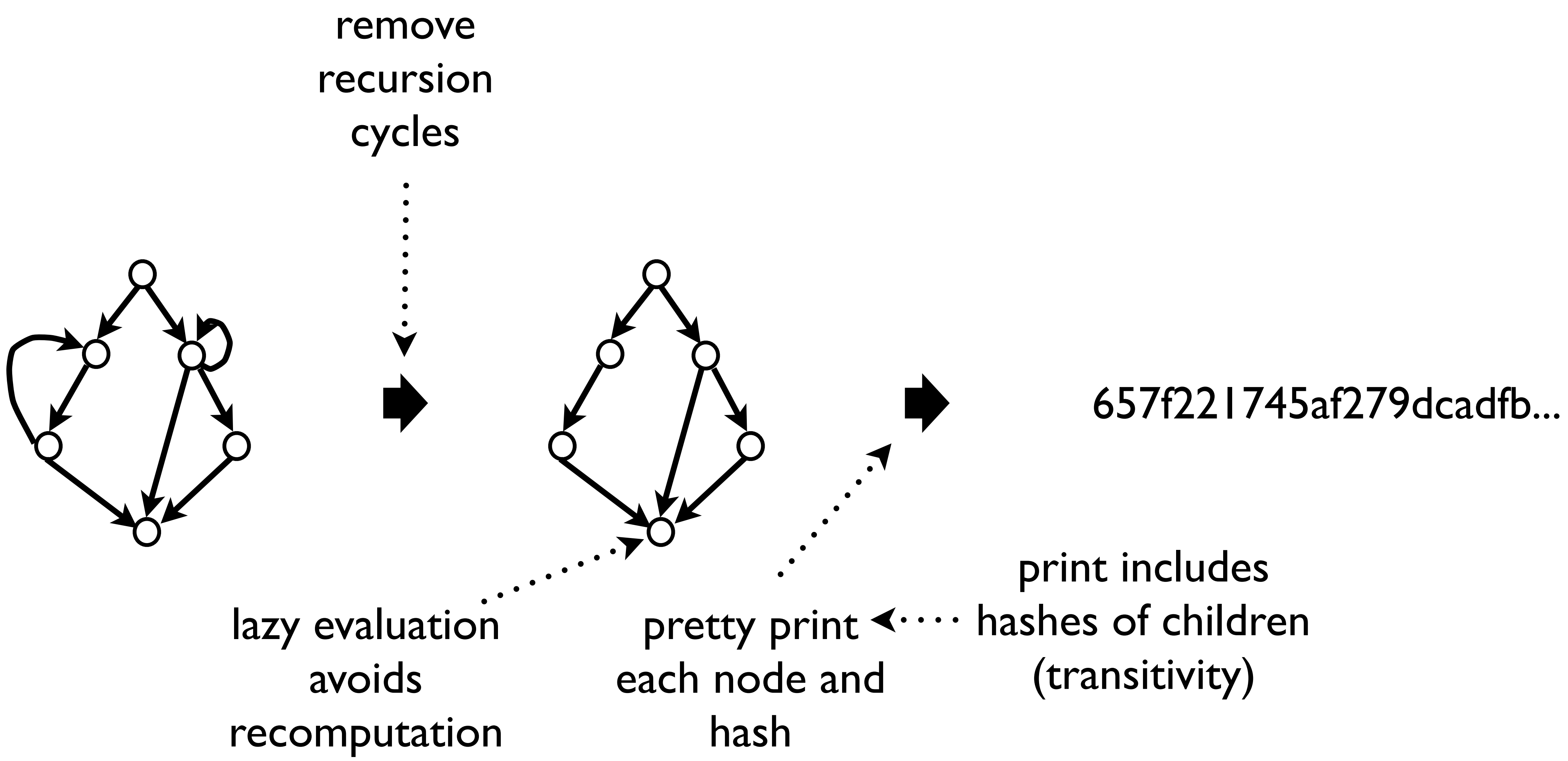
Separate compilation


.ml
type t = ..r..
CamlP4
.ml
hash
Ocaml
Compiled program
hash
.ml
type r = ...
type reference
CamlP4 processes a module at a time.
Graph is incomplete: would not detect changes to r

Hashing at runtime



Computing the hash



remove recursion cycles
657f221745af279dcadfb...
lazy evaluation avoids recomputation
pretty print each node and hash
print includes hashes of children (transitivity)

Results

- Used daily with 120,000 lines of Ocaml code and 2890 datatypes (1/5 marshaled).
- Supports polymorphic datatypes, functors, mutual recursion, ADTs.
- Shows where the difference is located (part of type-graph is stored).

Shortcomings

- Error messages can be hard to understand at first.
- Harder to integrate with other tools (ocamldoc, ocamltags).

Future work

- Open source the hashing framework.
- Implement backward compatible unmarshaling.

Related work

- Jane St. Capital's Type-conv & Bin_prot [1].
- Generic programming: PolyP, GH, Clean.
- GHC: ABI checking with MD5.
- Java serialization.

[1] http://www.janestcapital.com/ocaml/index.html

VectorFabrics

Student projects

- Declarative graph rewriting (Vali Georgescul).
- Automatic generation of C programs.
- Distributed memory utilization in dedicated embedded systems.

Vector Fabrics

Conclusions

- Successful automatic and seamless version control of datatypes based on CamlP4.