The Future of PPX

Towards a unified and more robust ecosystem

Nathan Rebours, Tarides

Jeremie Dimino, Jane Street

What is PPX?

What is PPX?

Syntax extensions for PPXes

- Extension points:

```
let x = [%eq: int list] [1; 2] [2; 3]
```

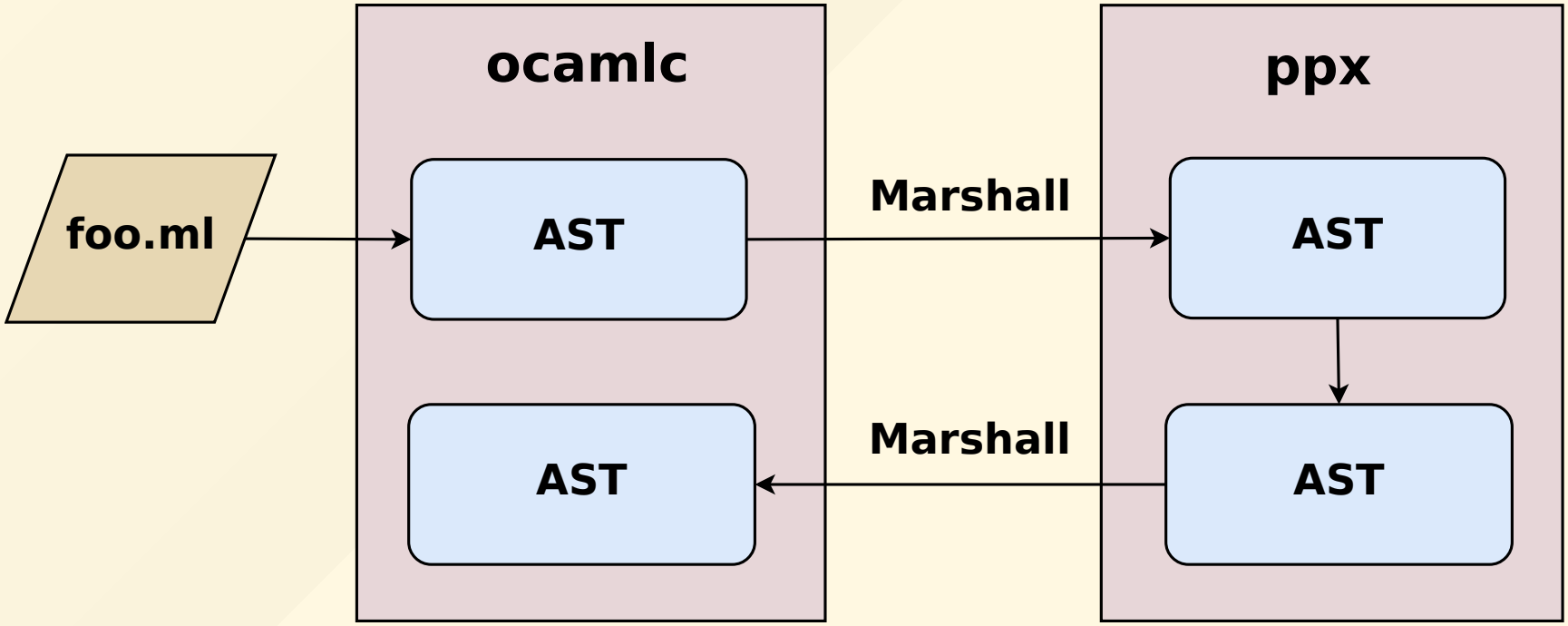
- Attributes:

```
type t = int list [@@deriving eq]
```

What is PPX?

Compiler integration

```
ocamlc -ppx ppx foo.ml
```

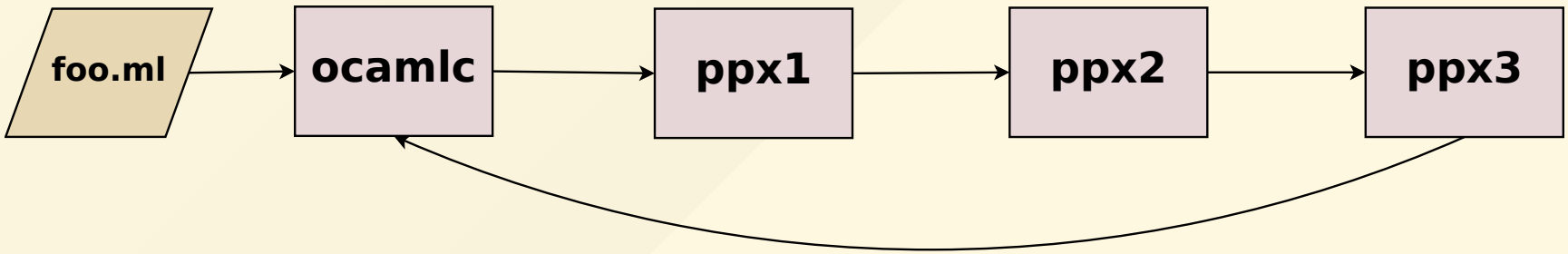


What are the issues with PPX?

What are the issues with PPX?

Combining several PPXes

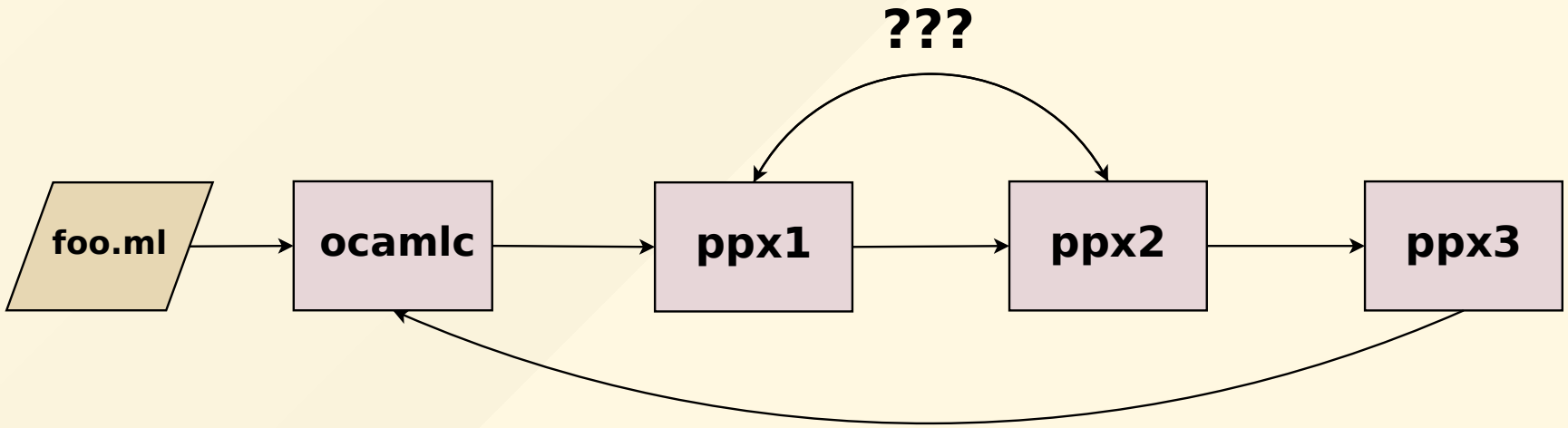
```
ocamlc -ppx ppx1 -ppx ppx2 -ppx ppx3 foo.ml
```



What are the issues with PPX?

Combining several PPXes

Is it equivalent to apply PPXes in different orders?

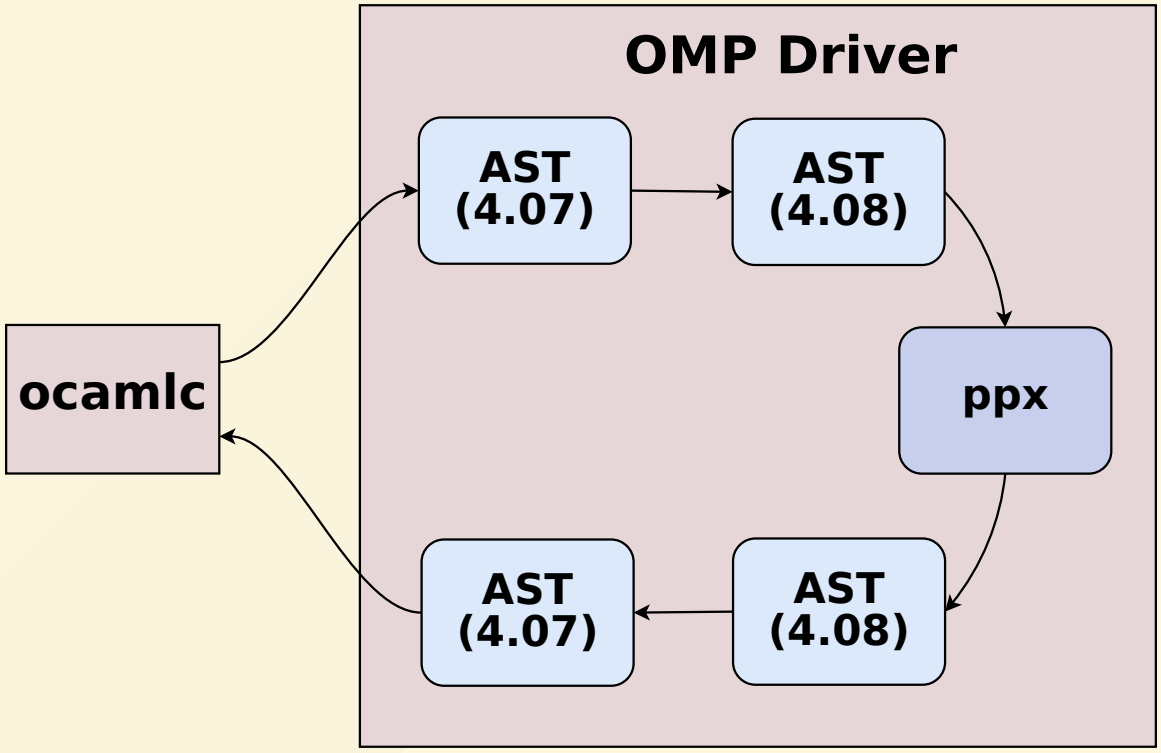


Issue for both PPX authors and users...

ocaml-migrate-parsetree

ocaml-migrate-parsetree

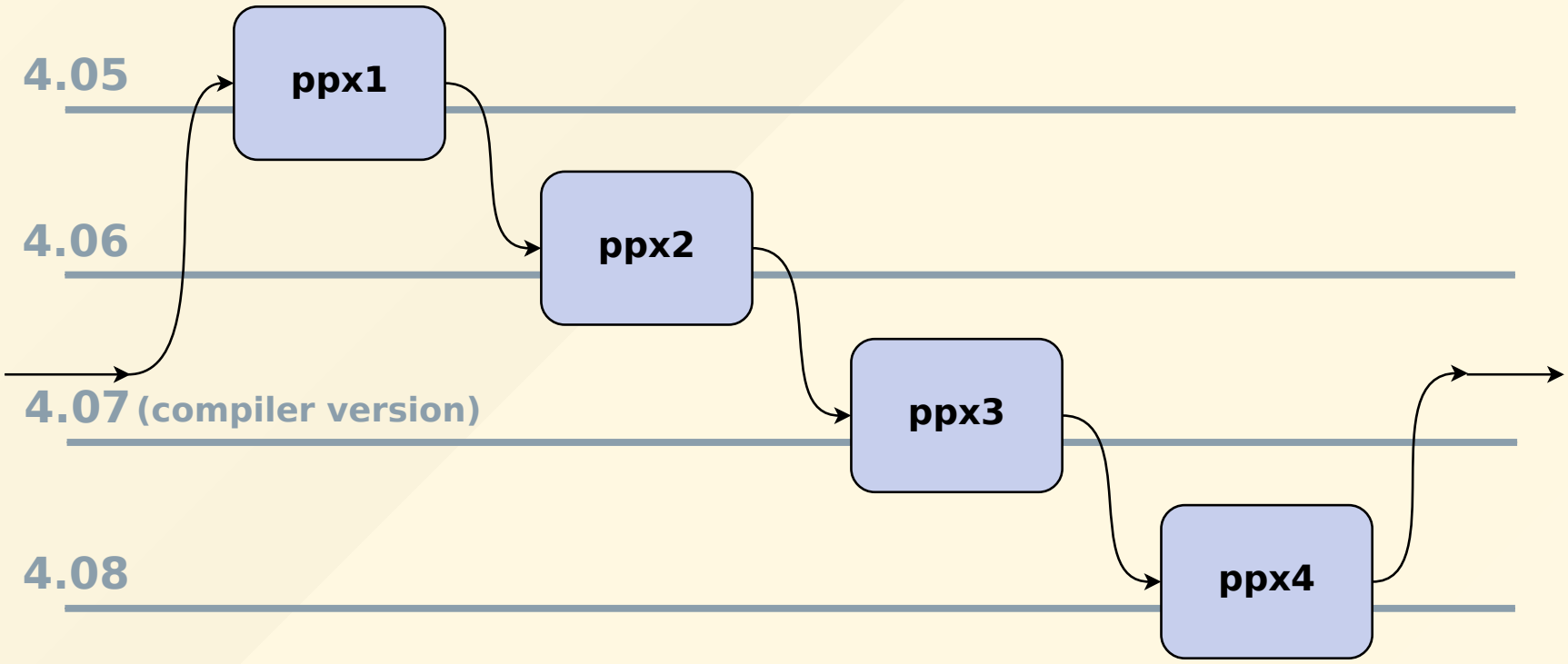
Driver



ocaml-migrate-parsetree

Combining several PPXes

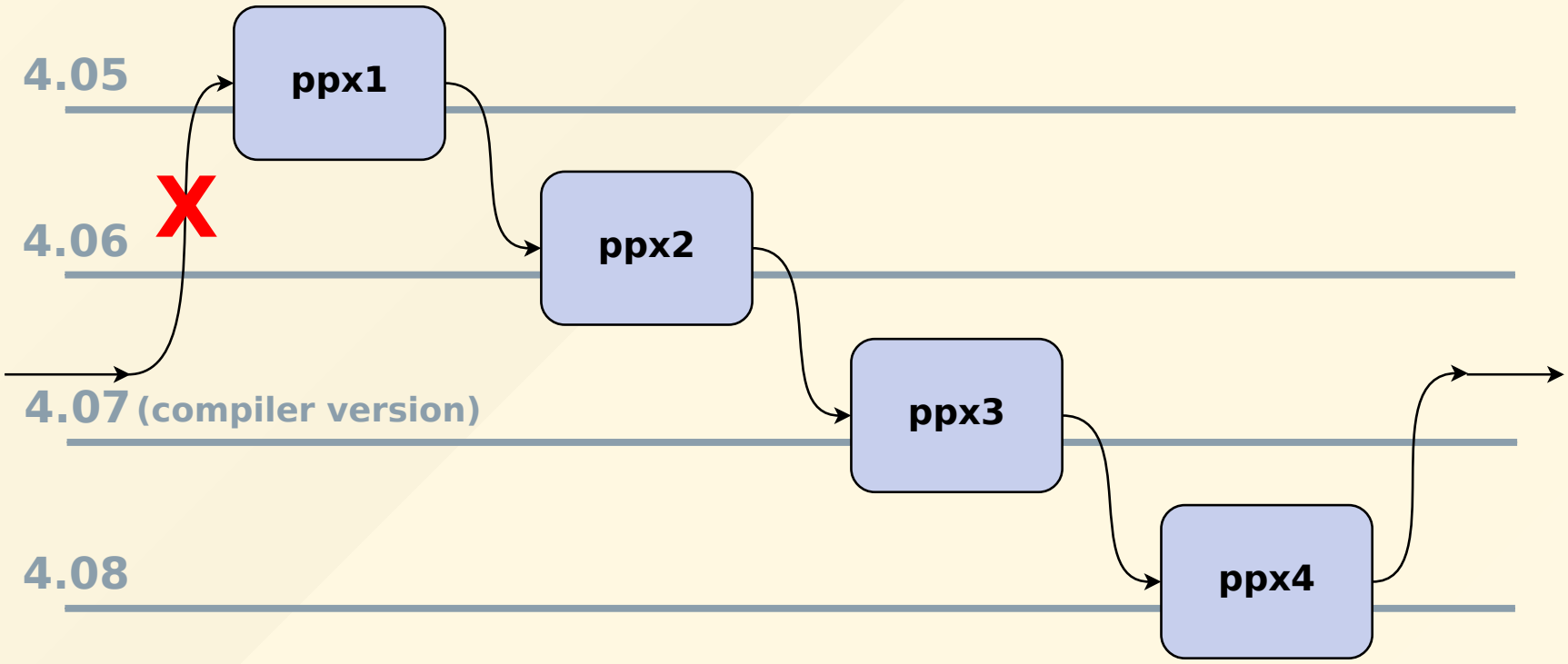
May involve a lot of AST migrations



ocaml-migrate-parsetree

Combining several PPXes

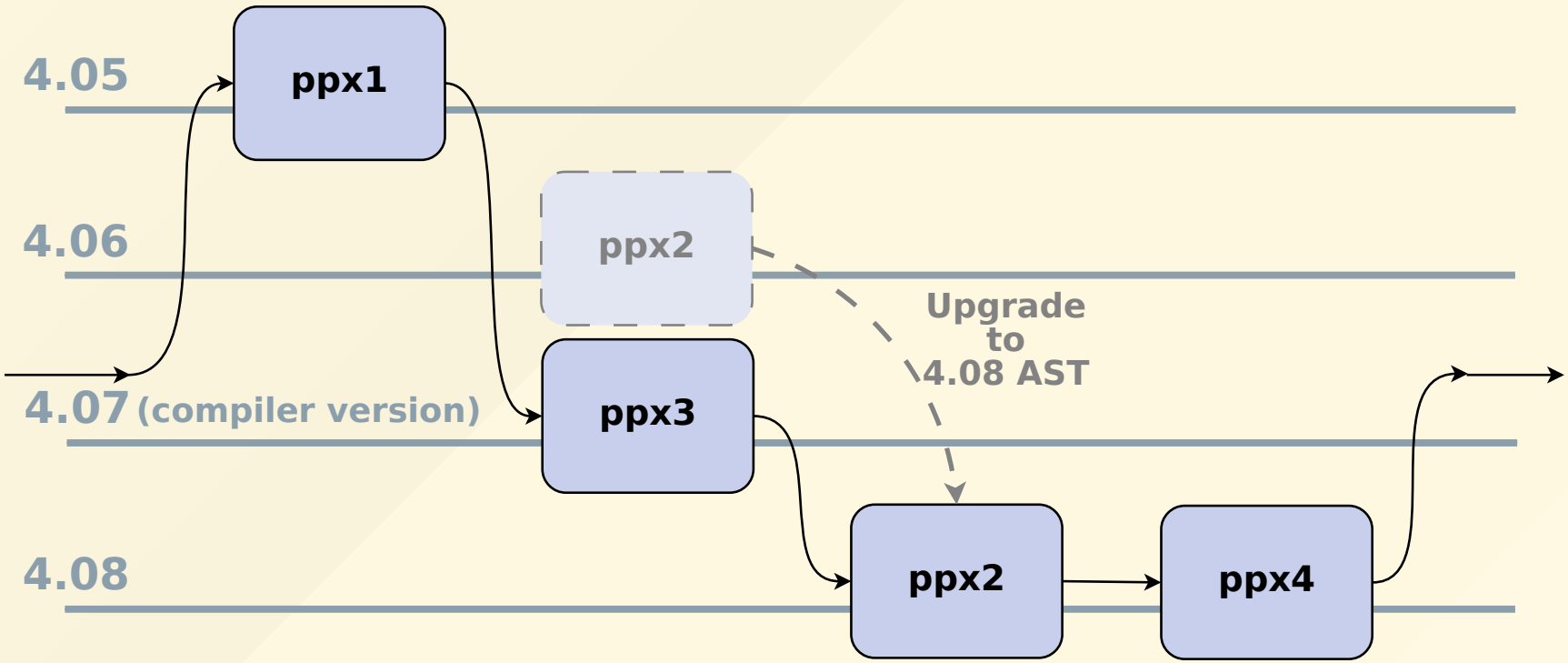
Backward migrations can fail



ocaml-migrate-parsetree

Combining several PPXes

The order is still an issue



ppxlib

ppxlib

- Recursively applies transformation to generated code.

```
let x = [%something ()] in
...
```

expands into

```
let x = 1 + [%something_else ()] in
...
```

ppxlib

- Quality of life improvements

```
type t =
  { a : int
  ; b : string [@defualt "b"]
  }
[@@deriving make]
```

```
Error: Uninterpreted attribute
; b : string [@defualt "b"]
              ^^^^^^^^^^^^^
```

ppxlib

Limitations