OCaml\textsubscript{light} (ESOP ’08) is a formal semantics for a substantial subset of the Objective Caml core language, suitable for writing and verifying real programs.

**OCaml\textsubscript{light} key points**

- Written in Ott
- Faithful to Objective Caml (very nearly)
- Type soundness proof mechanized in HOL (Coq and Isabelle/HOL definitions generated too)
- Operational semantics validated on test programs
- Small-step operational semantics (131 rules)
- Type system (179 rules, below)

\begin{itemize}
  \item definitions:
    \begin{itemize}
      \item variant data types (e.g., type \( t = \text{List} \)} \text{of} \text{int} \| \text{C} \text{of} \text{char} ),
      \item record types (e.g., type \( t = (\text{f} : \text{int}; \text{g} : \text{bool})$)),
      \item parametric type constructors (e.g., type \( 'a \text{t} = \text{C of} \text{'a} $)),
      \item type abbreviations (e.g., type \( 'a \text{t} = 'a \ast \text{int} $)),
      \item mutually recursive combinations of the above (excepting abbreviations),
      \item exceptions, and values;
      \item expressions for type annotations, sequencing, and primitive values (functions, lists, tuples, and records);
      \item \text{with} (record update), if, while, for, assert, try, and raise expressions;
      \item let-based polymorphism with an SML-style value restriction;
      \item mutually-recursive function definitions via \text{let rec} ;
      \item pattern matching, with nested patterns, as patterns, and \text{“or” (|)} patterns;
      \item mutable references with \text{ref}, \text{f}, and \text{:} $n$;
      \item polymorphic equality (the Objective Caml = operator);
      \item 31-bit word semantics for ints (using an existing HOL library); and
      \item IEEE-754 semantics for floats (using an existing HOL library).
    \end{itemize}
\end{itemize}