What is a *type*?

A set of terms. A term `t` belongs to a type `T` if it can be determined statically that `t` evaluates to a value of the form `T` is trying to represent.

What is a *typing relation*?

The smallest binary relation between terms and types that satisfies the language's type rules.

A term in the typing relation is called ...

... *typable* or *well typed*.

An algorithm for assigning types to terms follows directly from what?

The *inversion* (or *generation*) *lemma*, which for any typing statement in the language shows how the proof could be generated, using nothing more than the typing relation.

How might you prove that a term has a given type?

Using a *typing derivation* which is a tree of applications of typing rules in which every leaf ends with a typing rule that has no premises.

How do you know when the evaluation of a term is "stuck"?

The term is not in a *canonical form* (a value) but no further evaluation rules apply.

How can we formalize the notion of a type system's soundness?

If the type system is sound a well-typed term when evaluated will not reach a "stuck state".

Given the "no stuck states" formalization of type soundness, how can type soundness be proved.

By proving *progress* and *preservation* properties.
Progress: well-typed terms are not stuck (*i.e.*, they can make progress).
Preservation: an evaluation step of a well-typed term results in another well-typed term.

Why is the preservation property often called *subject reduction/evaluation*.

The statement "`t` has type `T`" remains true even as its subject (`t`) is reduced.