The compiler intermediate form for constraint language implementation

- The intermediate form is the compiler's internal representation of the program.

- A good intermediate form should be able to express correctness independently of termination and efficiency.

- Ideally, an intermediate form has 3 orthogonal components:

    1. Primitive constraints: the basic data manipulation instructions, which are executable and non-directional. They may include structural constraints (e.g., unification of Herbrand terms) and arithmetic constraints (e.g., equalities and inequalities).

    2. Control flow: all modifications of control flow. This includes calls, jumps, closures and continuations.

    3. Type attributes: standard data types as well as modedness and aliasing information.

- This allows expressing purely declarative execution as well as efficient operational execution.

- Compilation proceeds in two steps:

    1. Add control and type information to the constraints. This can be done through programmer annotation, compiler transformations and global analysis.

    2. Translate the annotated constraints to the target language.