2. Aplicaciones de lenguajes

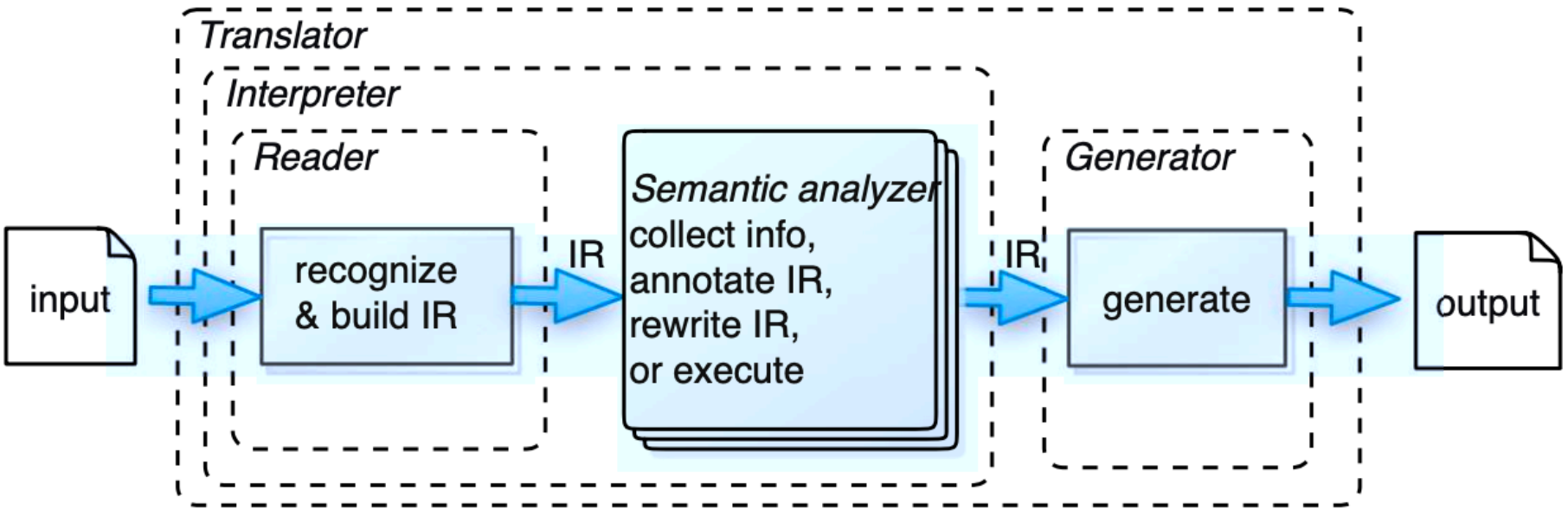
- Vista general
- Intérprete de código de byte
- Buscador de errores
- Compilador de C y C++

Vista general

- Las aplicaciones de lenguajes pueden ser muy complejas, por eso es necesario partirlas en piezas pequeñas. Los componentes encajan en una tubería multietapa que analiza o manipula un flujo de entrada. Esta tubería convierte gradualmente un enunciado de entrada en una estructura de datos interna fácil de manejar o lo traduce a un enunciado en otro lenguaje.

- La idea principal es que un lector (reader) reconoce una entrada y construye una representación intermedia (IR) que alimenta al resto de la aplicación. En el final opuesto, un generador emite una salida basada en la IR y lo que la aplicación “aprendió” en las etapas intermedias.

Vista general



Vista general

- El tipo de aplicación que se está construyendo dicta las etapas de la tubería y como las unimos. Existen cuatro grandes categorías de aplicaciones:

- **Lector.** Un lector construye una estructura de datos a partir de una o más flujos de entrada. Los flujos de entrada son texto normalmente, pero pueden ser también datos binarios (Ejemplos: lectores de archivos de configuración y herramientas de análisis de programa).

- **Generador.** Un generador camina sobre una estructura de datos interna y emite una salida (Ejemplos: herramientas de mareo de objetos a bases de datos relacionales, serializadores de objetos, generadores de código fuente y generadores de páginas web).

- **Traductor o reescritor.** Un traductor lee una entrada de texto o binaria y emite una salida de acuerdo al mismo u otro lenguaje. Es en esencia una combiandción de lector y generador. (Ejemplos: traductores de lenguajes de programación viejos a modernos, traductores de wiki a HTML, refactorizadores, generadores de bitácoras, formateadores para impresión, preprocesadores de macros, ensambladores y compiladores).

- **Intérprete.** Un intérprete lee, decodifica y ejecuta instrucciones. Los intérpretes van desde una calculadora simple hasta las implementaciones de lenguajes de programación como Java, Python o Ruby.

Ejemplos de aplicaciones

- Las aplicaciones de lenguajes son parecidas a los fractales. Conforme se hace un acercamiento se ve que las etapas en la tubería son a su vez tuberías multietapa.

- Para entender de manera general estas aplicaciones se presentan tres ejemplos: un intérprete de código de bytes, un buscador de errores y un compilador de C/C++.

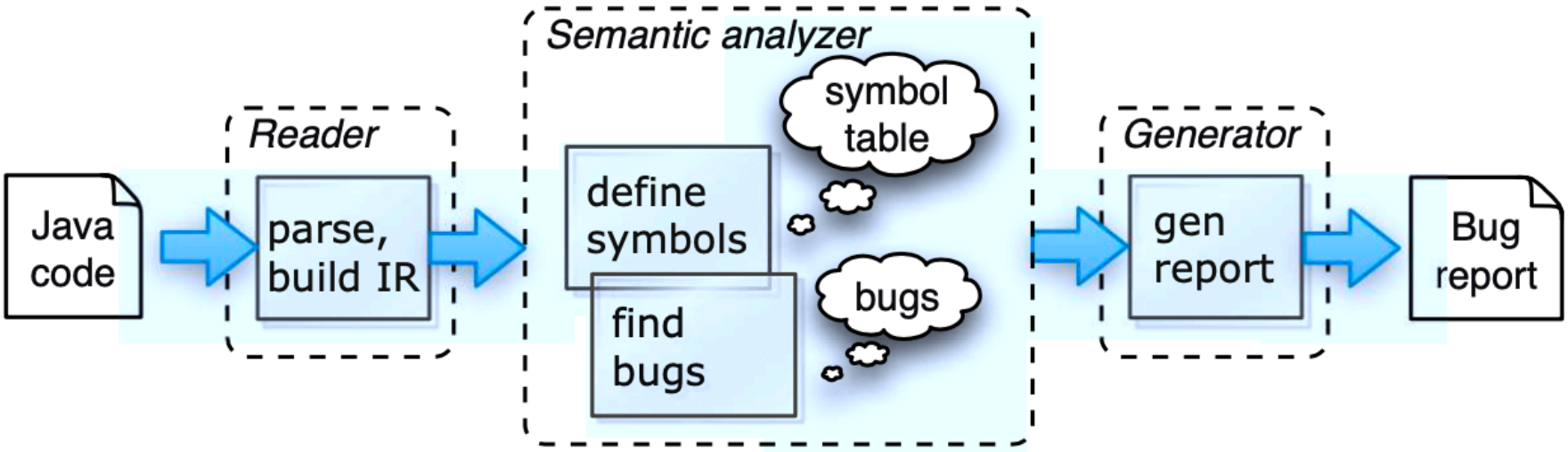
- El objetivo es enfatizar las similitudes en la arquitectura entre las diferentes aplicaciones y entre las distintas etapas dentro de una aplicación.

- Entre más se conozca sobre las aplicaciones de lenguajes existentes, más fácil será diseñar o interpretar una aplicación nueva.

Intérprete de código de byte

- Un intérprete es un programa que ejecuta otros programas. Un intérprete simula un procesador de hardware en software, por lo cual se les llama máquinas virtuales.
- El conjunto de instrucciones del intérprete es de muy bajo nivel, pero más alto que el código máquina.
- A las instrucciones se les llama código de byte porque cada instrucción se puede representar con un valor entero entre 0 y 255.
- Algunos lenguajes con intérpretes de código de byte son: Java, Lua, Python, Ruby, C# y Smalltalk.

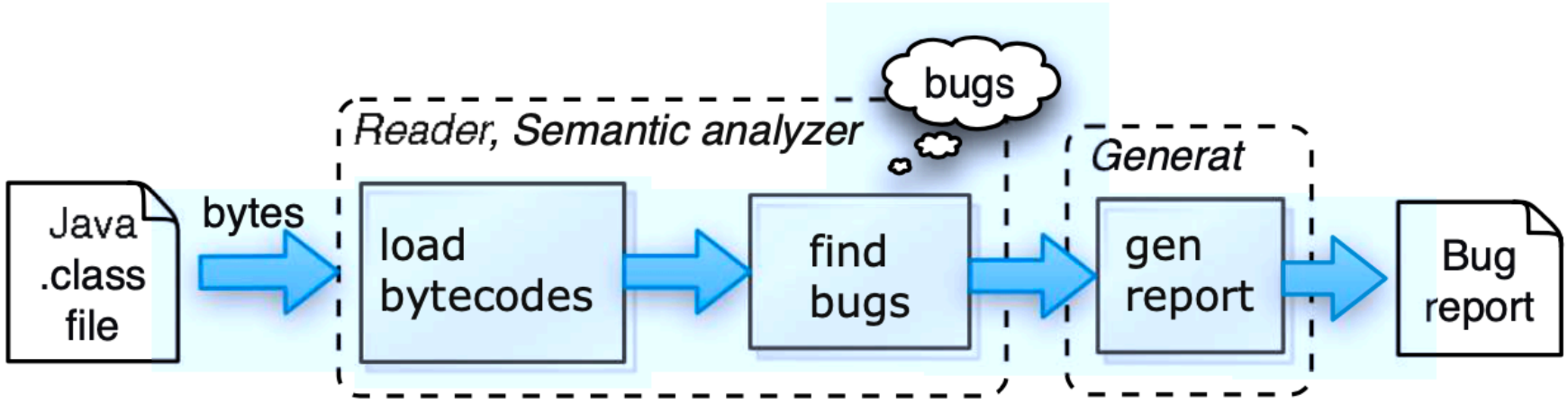
Intérprete de código de byte



Buscador de errores

- Un buscador de errores trata de encontrar errores de lógica cometidos por el programador analizando el código fuente o alguna representación intermedia.
- Como ejemplo se presenta un buscador de errores de Java.

Buscador de errores



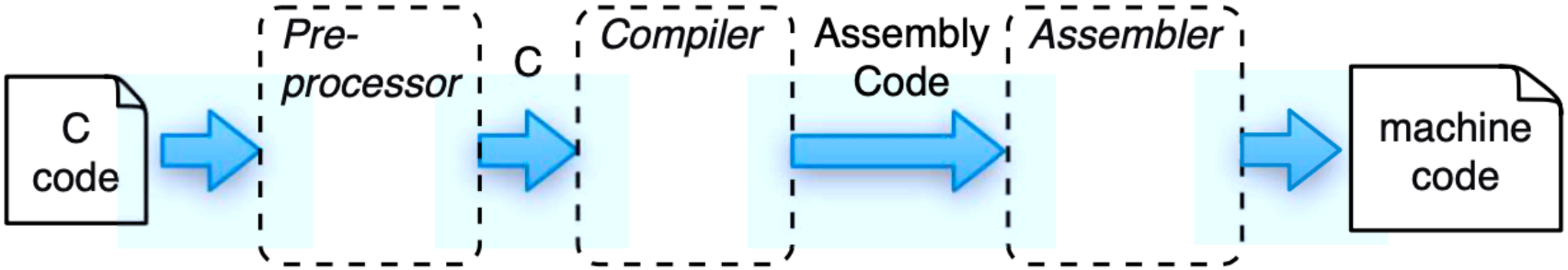
Compilador de C

- Un compilador de C luce como un programa grande ya que utilizamos un solo comando para ejecutarlo (via cc o gcc en Unix).

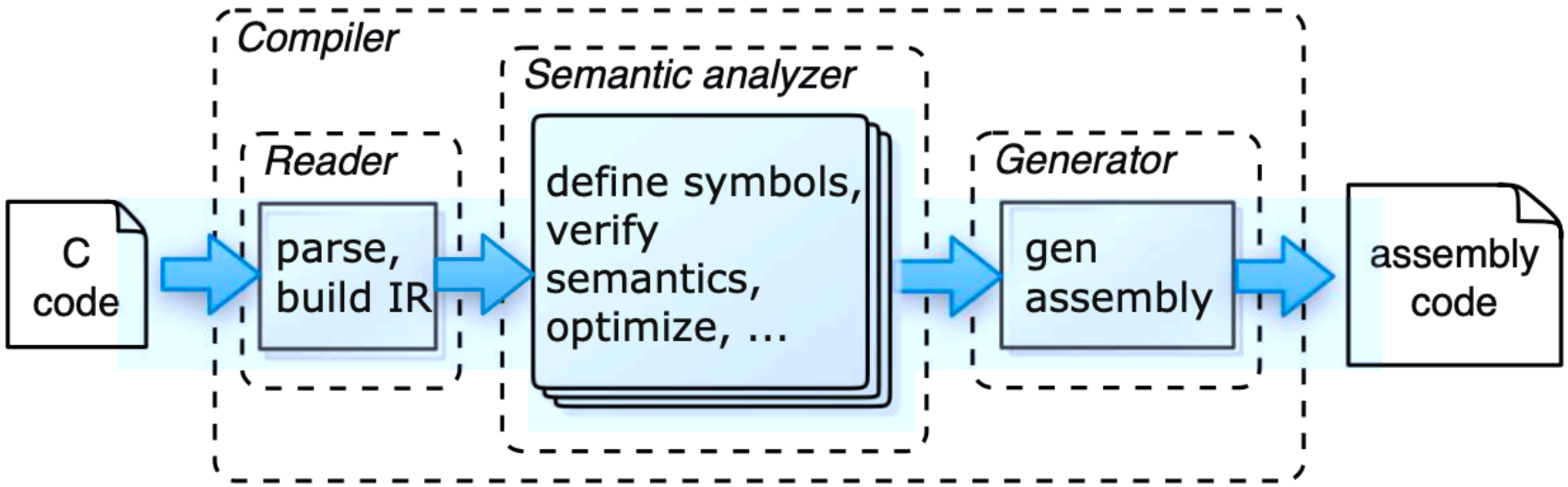
- A pesar que el compilador de C es el componente más complejo, el proceso de compilación tiene varios componentes.

- Antes de la compilación, se deben preprocesar los archivos C. El preprocesador emite código C puro con directivas que entiende el compilador. El compilador procesa este código C y emite código ensamblador. Un ensamblador separado traduce a código de máquina binario.

Compilador de C



Compilador de C



Compilador de C++

- Para construir un compilador de C++, se reduce el problema a utilizar el compilador de C, traduciendo el programa fuente C++ a C.