Multilingual Programs

- Why are large programs often written in a combination of different programming languages?
  - To use the most suitable language for a given problem.
  - To gradually migrate existing projects from one language to another.
  - To reuse existing source code.

Multilingual Programs

- Why are large programs often written in a combination of different programming languages?
  - To use the most suitable language for a given problem.
  - To gradually migrate existing projects from one language to another.
  - To reuse existing source code.
- Examples
  - Diversity of hardware ISAs that no single low-level language can cover: a RISC ISA, a multithreaded GPU, a VLIW DSP, a FPGA.

Multilingual Programs

- Why are large programs often written in a combination of different programming languages?
  - To use the most suitable language for a given problem.
  - To gradually migrate existing projects from one language to another.
  - To reuse existing source code.
- Examples
  - Diversity of hardware ISAs that no single low-level language can cover: a RISC ISA, a multithreaded GPU, a VLIW DSP, a FPGA.
  - Domain-specific languages such as SQL make certain categories of operations impossible to implement but make common tasks within their domain possible to express in a few lines of code.

Multilingual Programs

- Why are large programs often written in a combination of different programming languages?
  - To use the most suitable language for a given problem.
  - To gradually migrate existing projects from one language to another.
  - To reuse existing source code.
- Examples
  - Diversity of hardware ISAs that no single low-level language can cover: a RISC ISA, a multithreaded GPU, a VLIW DSP, a FPGA.
  - Domain-specific languages such as SQL make certain categories of operations impossible to implement but make common tasks within their domain possible to express in a few lines of code.
  - Languages like Java sacrifice the ability to manipulate pointers directly (and unsafely) in exchange for a more abstract memory model.

Multilingual Programs

- Why are large programs often written in a combination of different programming languages?
  - To use the most suitable language for a given problem.
  - To gradually migrate existing projects from one language to another.
  - To reuse existing source code.
- Examples
  - Diversity of hardware ISAs that no single low-level language can cover: a RISC ISA, a multithreaded GPU, a VLIW DSP, a FPGA.
  - Domain-specific languages such as SQL make certain categories of operations impossible to implement but make common tasks within their domain possible to express in a few lines of code.
  - Languages like Java sacrifice the ability to manipulate pointers directly (and unsafely) in exchange for a more abstract memory model.
  - Mixed-mode programs.
    - HLLs typically call code written in lower-level languages as part of their standard libraries (e.g., GUI rendering).
    - Business logic written in JavaScript uses a database driver written in C.

Common Areas of Mismatch

- Object models.
  - Are factory objects (classes) a first-class construct? If so, are they also objects?
  - Are there zero, one, or many superclasses for an object?
  - Is method lookup tied to the static type system? Can it be modified at runtime?

Common Areas of Mismatch

- Object models.
  - Are factory objects (classes) a first-class construct? If so, are they also objects?
  - Are there zero, one, or many superclasses for an object?
  - Is method lookup tied to the static type system? Can it be modified at runtime?
- Memory models.
  - Is deallocation manual or automatic?
  - Is destruction deterministic or nondeterministic?
  - Are memory accesses safe or unsafe?

Common Areas of Mismatch

- Object models.
  - Are factory objects (classes) a first-class construct? If so, are they also objects?
  - Are there zero, one, or many superclasses for an object?
  - Is method lookup tied to the static type system? Can it be modified at runtime?
- Memory models.
  - Is deallocation manual or automatic?
  - Is destruction deterministic or nondeterministic?
  - Are memory accesses safe or unsafe?
- Exceptions and unwinding.
  - Exception models differ widely in structure and usage frequency.

Common Areas of Mismatch

- Object models.
  - Are factory objects (classes) a first-class construct? If so, are they also objects?
  - Are there zero, one, or many superclasses for an object?
  - Is method lookup tied to the static type system? Can it be modified at runtime?
- Memory models.
  - Is deallocation manual or automatic?
  - Is destruction deterministic or nondeterministic?
  - Are memory accesses safe or unsafe?
- Exceptions and unwinding.
  - Exception models differ widely in structure and usage frequency.
- Mutability and side effects.
  - An immutable reference can lead to a mutable object.
  - Shallow vs. deep copying.

Common Areas of Mismatch

- Object models.
  - Are factory objects (classes) a first-class construct? If so, are they also objects?
  - Are there zero, one, or many superclasses for an object?
  - Is method lookup tied to the static type system? Can it be modified at runtime?
- Memory models.
  - Is deallocation manual or automatic?
  - Is destruction deterministic or nondeterministic?
  - Are memory accesses safe or unsafe?
- Exceptions and unwinding.
  - Exception models differ widely in structure and usage frequency.
- Mutability and side effects.
  - An immutable reference can lead to a mutable object.
  - Shallow vs. deep copying.
- Models of parallelism.

Approaches to Interoperability

- Common language runtime (e.g., CLR)
  - Restrict languages to a subset of features that can be mapped to a common object/memory model, so that conformant languages can exchange data.

Approaches to Interoperability

- Common language runtime (e.g., CLR)
  - Restrict languages to a subset of features that can be mapped to a common object/memory model, so that conformant languages can exchange data.
- Foreign function interfaces (e.g., JNI, CXX)
  - Define a specific interface between a pair of languages that allows accessing foreign objects.

Approaches to Interoperability

- Common language runtime (e.g., CLR)
  - Restrict languages to a subset of features that can be mapped to a common object/memory model, so that conformant languages can exchange data.
- Foreign function interfaces (e.g., JNI, CXX)
  - Define a specific interface between a pair of languages that allows accessing foreign objects.
- Multi-language source code (e.g., Jeannie, PyHyp)
  - Compose languages at a very fine granularity by allowing the programmer to toggle between syntax and semantics of languages on the source code level.

Approaches to Interoperability

- Common language runtime (e.g., CLR)
  - Restrict languages to a subset of features that can be mapped to a common object/memory model, so that conformant languages can exchange data.
- Foreign function interfaces (e.g., JNI, CXX)
  - Define a specific interface between a pair of languages that allows accessing foreign objects.
- Multi-language source code (e.g., Jeannie, PyHyp)
  - Compose languages at a very fine granularity by allowing the programmer to toggle between syntax and semantics of languages on the source code level.
- Interface definition languages (e.g., XPCOM, CORBA)
  - Define message-based inter-process communication interfaces that can be mapped to multiple languages.

Approaches to Interoperability

- Common language runtime (e.g., CLR)
  - Restrict languages to a subset of features that can be mapped to a common object/memory model, so that conformant languages can exchange data.
- Foreign function interfaces (e.g., JNI, CXX)
  - Define a specific interface between a pair of languages that allows accessing foreign objects.
- Multi-language source code (e.g., Jeannie, PyHyp)
  - Compose languages at a very fine granularity by allowing the programmer to toggle between syntax and semantics of languages on the source code level.
- Interface definition languages (e.g., XPCOM, CORBA)
  - Define message-based inter-process communication interfaces that can be mapped to multiple languages.
- Multi-language semantics
  - Mostly theoretical underpinnings.